Open Command and Control (OpenC2) Profile for Stateless Packet Filtering Version 1.0

Committee Specification Draft 02

20 July 2018

Specification URIs

This version:

http://docs.oasis-open.org/openc2/oc2slpf/v1.0/csd02/oc2slpf-v1.0-csd02.md
(Authoritative)
http://docs.oasis-open.org/openc2/oc2slpf/v1.0/csd02/oc2slpf-v1.0-csd02.html
http://docs.oasis-open.org/openc2/oc2slpf/v1.0/csd02/oc2slpf-v1.0-csd02.pdf

Previous version:

http://docs.oasis-open.org/openc2/oc2slpf/v1.0/csd01/md/oc2slpf-v1.0-wd01.md
(Authoritative)
http://docs.oasis-open.org/openc2/oc2slpf/v1.0/csd01/oc2slpf-v1.0-csd01.html
http://docs.oasis-open.org/openc2/oc2slpf/v1.0/csd01/oc2slpf-v1.0-csd01.pdf

Latest version:

http://docs.oasis-open.org/openc2/oc2slpf/v1.0/csd02/oc2slpf-v1.0-csd02.md
(Authoritative)
http://docs.oasis-open.org/openc2/oc2slpf/v1.0/csd02/oc2slpf-v1.0-csd02.html
http://docs.oasis-open.org/openc2/oc2slpf/v1.0/csd02/oc2slpf-v1.0-csd02.pdf

Technical Committee:

OASIS Open Command and Control (OpenC2) TC

Chairs

Joe Brule (jmbrule@nsa.gov), National Security Agency
Sounil Yu (sounil.yu@bankofamerica.com), Bank of America

Editors

Joe Brule (jmbrule@nsa.gov), National Security Agency

Duncan Sparrell (duncan@sfractal.com), sFractal Consulting
Alex Everett (alex.everett@unc.edu), University of North Carolina, Chapel Hill

Abstract

Open Command and Control (OpenC2) is a concise and extensible language to enable the command and control of cyber defense components, subsystems and/or systems in a manner that is agnostic of the underlying products, technologies, transport mechanisms or other aspects of the implementation. Stateless packet filtering is a cyber defense mechanism that denies or allows traffic based on static properties of the traffic (such as address, port, protocol etc). This profile defines the actions, targets, specifiers and options that are consistent with version 1.0 of the OpenC2 Language Specification in the context of stateless packet filtering.

Status

This document was last revised or approved by the OASIS Open Command and Control (OpenC2) TC on the above date. The level of approval is also listed above. Check the "Latest version" location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Technical Committee (TC) are listed at https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=openc2#technical.

TC members should send comments on this specification to the TC's email list. Others should send comments to the TC's public comment list, after subscribing to it by following the instructions at the "Send A Comment" button on the TC's web page at https://www.oasis-open.org/committees/openc2/.

This specification is provided under the Non-Assertion Mode of the OASIS IPR Policy, the mode chosen when the Technical Committee was established. For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the TC's web page (https://www.oasis-open.org/committees/openc2/ipr.php).

Note that any machine-readable content (Computer Language Definitions) declared Normative for this Work Product is provided in separate plain text files. In the event of a discrepancy between any such plain text file and display content in the Work Product's prose narrative document(s), the content in the separate plain text file prevails.

Citation format:

When referencing this specification the following citation format should be used:

[OpenC2-SLPF-v1.0]

Open Command and Control (OpenC2) Profile for Stateless Packet Filtering Version 1.0. Edited by Joe Brule, Duncan Sparrell and Alex Everett. 20 July 2018. OASIS Committee Specification Draft 01. http://docs.oasis-open.org/openc2/oc2slpf/v1.0/csd02/oc2slpf-v1.0-csd02.html. Latest version: http://docs.oasis-open.org/openc2/oc2slpf/v1.0/oc2slpf-v1.0.html.

Notices

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the “OASIS IPR Policy”). The full Policy may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an “AS IS” basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS’ procedures with respect to rights in any document or deliverable

produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

Table of Contents

1 Introduction 8
1.1 IPR Policy 8
1.2 Purpose and Scope 8
1.3 Terminology 9
1.4 Document Conventions 9
1.5 Document Overview 9
1.6 Normative References 10
RFC2119 10
RFC8174 10
1.7 Non normative References 10
1.8 Acknowledgments 10
2 OpenC2 Language Binding 11
2.1 OpenC2 Components 11
2.1.1 Actions 11
2.1.2 Targets 12
2.1.2.1 Type Name: Target 13
2.1.2.2 Type Name: Rules 13
2.1.2.3 Type Name: Rule-Target 14
2.1.3 Command Arguments 14
2.1.4 Actuator Data Model 16
2.2 OpenC2 Responses 17
2.2.1 Response Codes 17
2.2.2 Results 19
2.2.2.1 Type Name: Results 19
2.2.2.2 Type Name: Rule-Item 19
2.2.2.3 Type Name: Rule-Action 19
2.2.2.4 Type Name: Rule-Target 19
2.2.2.5 Example Response 20
2.3 OpenC2 Commands 20
2.3.1 ‘Allow’ 21
2.3.2 ‘Allow ip_connection’ 22
2.3.3 ‘Allow ip-addr’ 23
2.3.4 ‘Deny’ 23
2.3.5 ‘Query’ 24
2.3.6 ‘Query openc2’ 24
2.3.7 ‘Query slpf:access_rules’ 24
2.3.8 ‘Delete’ 25
2.3.9 ‘delete slpf:access_rules’ 25
2.3.10 Update 25
2.3.11 Update file 26

3 Conformance statements 27
3.1 Conformance Clause 1: OpenC2 SLPF Entities (both producers and consumers) 27
3.2 Conformance Clause 2: OpenC2 Stateless Packet Filter Producer: Command Arguments 28
3.3 Conformance Clause 3: OpenC2 Stateless Packet Filter Consumer: Processing and Response to Commands and command arguments 28
4 Annex 1 SLPF Schema 30
5 Annex 2 OpenC2 Schema 32
6 Annex 3 Sample commands (Informative) 36
6.1 Deny and Allow 36
6.2 Update 38
6.3 Query 39

1 Introduction

A ‘Stateless-Packet-Filter’ (SLPF) is a policy enforcement mechanism that restricts or permits traffic based on static values such as source address, destination address, and/or port numbers. A Stateless-Packet-Filter does not consider traffic patterns, connection state, data flows, applications, or payload information. The scope of this profile is limited to Stateless-Packet-Filters herein referred to as SLPF.

This actuator profile specifies the set of actions, targets, specifiers, and command arguments that integrates SLPF functionality with the Open Command and Control (OpenC2) command set. Through this command set, cyber security orchestrators may gain visibility and provide control into the SLPF functionality in a manner that is independent of the vendor or generator of the SLPF function.

1.1 IPR Policy

This specification is provided under the Non-Assertion Mode of the OASIS IPR Policy, the mode chosen when the Technical Committee was established. For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the TC’s web page (https://www.oasis-open.org/committees/openc2/ipr.php).

1.2 Purpose and Scope

All components, devices and systems that provide SLPF functionality will implement the OpenC2 ACTIONS, TARGETS, SPECIFIERS and ARGUMENTS identified as required in this document. Actions that are applicable, but not necessarily required, for SLPF will be identified as optional.

The purpose of this document is to:

- Identify the required and optional OpenC2 ACTIONS for actuators with SLPF functionality.
- Identify the required and optional TARGET types and associated specifiers for each action in the SLPF class of actuators.
- Identify ACTUATOR SPECIFIERS, ACTUATOR-ARGUMENTS and COMMAND-ARGUMENTS for each action-target pair that are applicable and/or unique to the SLPF class of actuators
- Annotate each Action/ Target pair with a justification and example and provide sample OpenC2 commands to a SLPF with corresponding responses
- Provide an abstract schema that captures the specifiers and options for a SLPF

This SLPF profile:

- Does not define or implement ACTIONS beyond those defined in Version 1.0 of the Language Specification.
- Is conformant with version 1.0 of the OpenC2 Language Specification

Cyber defense systems that are utilizing OpenC2 may require the following components to implement the SLPF profile:

- OpenC2 Producers: Devices that send commands, receive responses, and manage the execution of commands involving one or more SLPF or other actuators with SLPF capability. The OpenC2 producer needs *a priori* knowledge of which commands the actuator can process and execute, therefore must understand the profiles for any device that it intends to command.
- OpenC2 Consumers: Devices or instances that provide stateless packet filtering functions. Typically these are actuators that execute the cyber defense function, but could be orchestrators (i.e., a device or instance that forwards commands to the actuator).

1.3 Terminology

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this specification are to be interpreted as described in IETF RFC 2119 [RFC2119].

1.4 Document Conventions

The following typographical conventions are used in this document.

italics

Indicates new terms, URLs, email addresses, filenames, and file extensions.

ALL CAPS

Used for components of the abstract syntax: ACTION, TARGET, ACTUATOR, OPTIONS.

‘single quotes’

Used for single actions or commands (i.e., action target pairs)

> **Editor’s Note** - to be provided later >

1.5 Document Overview

TBD

1.6 Normative References

RFC2119

Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels”, BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, http://www.rfc-editor.org/info/rfc2119.

RFC8174

Leiba, B., “Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words”, BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, http://www.rfc-editor.org/info/rfc8174.

TBD

1.7 Non normative References

TBD

1.8 Acknowledgments

TBD

2 OpenC2 Language Binding

This section defines the set of ACTIONS, TARGETS, SPECIFIERS, and ARGUMENTS that are meaningful in the context of an SLPF. This section also describes the format of the response frame's status and results field. This section organized into three major subsections; Components, Response and Commands.

2.1 OpenC2 Components

The components of an OpenC2 command include ACTIONS, TARGETS, ACTUATORS and associated ARGUMENTS and SPECIFIERS which are defined independently. Appropriate aggregation of the components will define a command-body that is meaningful in the context of an SLPF.

The components of an OpenC2 command include:

- ACTIONS: A subset of the ACTIONS defined in the OpenC2 Language specification.
  - This profiles does not define ACTIONS that are external to Version 1.0 of the OpenC2 Language Specification.
  - This profile MAY define the actions in the context of a SLPF
  - This profile SHALL NOT define actions in a manner that is inconsistent with version 1.0 of the OpenC2 language specification
- TARGETS: A subset of the TARGETs and target-specifiers defined in the Language specification that are meaningful in the context of SLPF or a TARGET and target-specifiers that are defined in this specification.
- ARGUMENTS: A subset of the COMMAND-ARGUMENTS defined in the Language Specification and a set of ACTUATOR-ARGUMENTS defined in this specification.
- ACTUATOR: A set of specifiers defined in this specification that are meaningful in the context of SLPF.

2.1.1 Actions

Table 2.1.1-1 presents the OpenC2 actions that are meaningful in the context of an SLPF and identifies which actions are required for at least one target type. The particular action/ target pairs that are required or optional are presented in section 2.3.

Table 2.1.1-1. SLPF Actions

Action	Description
query	The query action initiates a single request for information. Used to communicate the supported options and determine the state or settings.
deny	The deny action prevents traffic or access
allow	The allow action permits traffic or access.
delete	Remove rule(s).
update	Instructs the actuator to retrieve and process a software update, reconfiguration, or some other update.

2.1.2 Targets

Table 2.1.2-1 lists the TARGETs defined in version 1.0 of the language specification that are applicable to SLPFS.

Table 2.1.2-1. Target Data Model Applicable to SLPF

Target	Description/Notes
ip_connection	Consists of the addresses (source and destination), port numbers (source and destination) and protocol identifier. An 'incomplete' five-tuple may be sent . Default for unspecified fields are "all".
openc2	Used to determine the profile(s) and imported targets supported by the actuator. SLPF must be included in the profiles attribute.
ip_addr	In the context of an SLPF, identifies IP address(es) that are to be denied (or allowed) regardless of source or destination. Equivalent to two ip-connections where the port and protocol fields are unspecified. Supports IPV4, IPV6 and CIDR notation.
file	In the context of SLPF, the path and name of the file needed for a configuration update.

Target	Description/Notes
slpf:access_rules	Used to enable an openc2 producer to determine the current elements in the rule set. This is an optional target type defined in this specification.

OpenC2 producers MUST implement the ip_connection AND the ip_addr target.

OpenC2 consumers MUST implement the ip_connection OR the ip_addr target. If a device or instance of the SLPF supports the ip_connection target, then the ip_addr target is OPTIONAL. Conversely, if the ip_addr target is implemented, then the ip_connection is OPTIONAL.

The slpf:access_rules target is defined in this specification. Implementations that choose to include the slpf:access_rules target type MUST import it in accordance with the procedures defined in section 2.2.6 of Version 1.0 of the OpenC2 Language Specification. The slpf:access_rules data profile attributes are:

1. The namespace identifier is: slpf:access_rules
2. The path for the data profile is: /docs.oasis-open.org/openc2/futurepath4profiles
3. A list of the object identifiers to be imported. Permitted objects are presented in sections 2.1.2.1 through 2.1.2.3
4. Implementations that choose to include the slpf:access_rules MUST include the objects listed as ‘required’ and MAY include the objects listed as optional.

2.1.2.1 Type Name: Target

Targets defined in this document are referenced under the slpf namespace.

Base Type: Choice

ID	Name	Type	Description
1	access_rules	Rules	Return the requested access rules in the response

2.1.2.2 Type Name: Rules

Specifiers for the access_rules target. If neither allow_rules nor deny_rules is included, return all rules.

Base Type: Map

ID	Name	Type	#	Description
1	allow_rules	Rule-Target	0…1	Return specified allow rules
2	deny_rules	Rule-Target	0…1	Return specified deny rules

2.1.2.3 Type Name: Rule-Target

Specifier for an access rule.

Base Type: Choice

ID	Name	Type	Description
1	all	Null	Any rules
2	ip_addr	…:IP-Addr	Rules affecting the specified address
3	ip_connection	…:IP-Connection	Rules affecting the specified 5-tuple

Editor's note - example:

```
{"action": "query",
 "target": {
    "slpf": {
      "access-rules": {
        "deny-rules": {"ip_addr": "1.2.3.4"}
}}}}
# the response will return all the denys associated with 1.2.3.4
```

Editor's Note - The slpf:access rules target is still being discussed and will be a topic at the pending face to face. There is general consensus for being able to query the actuator for a rule number (needed to support 'delete rule_number', but the details of this target type are TBS and subject to modification>

2.1.3 Command Arguments

Arguments provide additional precision to a command by including information such as how, when or where a command is to be executed. Table 2.1.3-1 summarizes the command arguments defined in Version 1.0 of the OpenC2 Language Specification as they relate to

SLPF functionality. Table 2.1.3-2 summarizes the command arguments that are defined in this specification.

Table 2.1.3-1. Command Arguments applicable to SLPF

Argument	Type	Description
response	string	Indicate the type of response required for the action.
start-time	datetime	The specific date/time to initiate the action. Implementations SHOULD use UTC to ensure consistency across different time zones. Unspecified start time defaults to 'now'
end-time	datetime	The specific date/time to end the action. Implementations SHOULD use Universal Time (UTC) to ensure consistency across different time zones. Unspecified end-time defaults to 'never'
duration	int	[=] seconds.

The semantics of start-time/end-time/duration are:

- If none are specified then the start time is now, the end time is never, and the duration is infinity
- Only two of the three are allowed on any given command and the third is derived from the equation end-time = start-time + duration
- If only start time is specified then end-time is never and duration is infinity
- If only end time is specified then start-time is now and duration is derived
- If only duration is specified then start-time is now and end-time is derived

Table 2.1.3-2. Command Arguments Unique to SLPF

Base Type: Map

ID	Argument	Type	#	Description
1	drop_process	enumerated	0…1	Three possible values. 'None'; Drop the packet and do not send a notification to the source of the packet. 'Reject'; Drop the packet and send an ICMP host unreachable (or equivalent) to the source of the packet 'False_ack'; Drop the packet and

ID	Argument	Type	#	Description
				send a false acknowledgement that the packet was received
2	running	Boolean	0…1	Normal operations assumes any change to a device are to be implemented as persistent changes. Setting the running modifier to TRUE results in a change that is not persistent in the event of a reboot or restart.
3	direction	string	0…1	Possible settings are ingress, egress or both. The default value is both. Ingress applies the allow or deny to incoming traffic only. Egress applies to outbound. Entities that do not support directionality MUST return a 501 error code when they receive ingress or egress as the option and MAY include 'Directionality not supported' in the error description.
4	prepend	boolean	0…1	If set to TRUE, then the new rule is placed at the beginning of the rule list. If set to FALSE, then the new rule is appended to the list. The default value is FALSE.
5	insert_rule	integer	0…1	A command option defined in this specification. An integer. Can only be sent if prepend is NOT sent. Specifies the number of the rule within a list, typically used in a top-down rule list.

2.1.4 Actuator Data Model

An ACTUATOR is the entity that provides the functionality and performs the action. The ACTUATOR executes the ACTION on the TARGET. In the context of this profile, the actuator is the SLPF and the presence of one or more specifiers further refine which actuator(s) shall execute the action.

Table 2.1.4-1 lists the specifiers that are applicable to the SPLF actuator. Section 5 provides sample commands with the use of specifiers.

Table 2.1.4-1. SLPF Specifiers

Base Type: Map

Specifier	Type	Description
hostname	String	RFC 1123 hostname (can be a domain name or IP address) for a particular device with SLPF functionality
named_group	string	User defined collection of devices with SLPF functionality
Asset_id	string	Unique identifier for a particular SLPF
Network-function-virtualization	list	Undefined list of indeterminate length
Network_function_virtualization	string	An identifier that maps to a virtualized or cloud based stateless packet filtering function

Editor's Note -Discussion regarding how to specify NFV are still underway. The three options are; List of undefined length and undefined content; Unique identifier; defined list with all optional content. Only one of the last two rows in table 2.1.4-1 will be kept in the final version >

2.2 OpenC2 Responses

Response messages originate from the actuator and are informative rather than a command or request that the recipient execute some task(s).

The formats and what is expected in the response for specific commands is summarized in Table 2.2-1 and additional details regarding the command and associated response will be captured in section 2.3. Examples will be provided in Annex 3.

Responses associated with required actions MUST be implemented. Implementations that include optional actions MUST implement the Responses associated with the implemented action.

2.2.1 Response Codes

Table 2.2-1. Response Codes

Status Code	Status Text
102	Processing. Command received but action not necessarily complete
200	OK. For response type complete, one of the following MAY apply:
	Openc2 target type (populated)
	slpf:access_rules target type (populated)
	Rule_number [ipconnection
	Rule_number
400	Unable to process command, parsing error
500	Server Error
	Rule not removed
	Cannot access file or path
501	Choice of : Not implemented
	Target not supported
	Option not supported
	Data model not supported
	Rule-number currently in use
	Rule not updated
	Command not supported

2.2.2 Results

The response to an access_rules query is a set of rule numbers matching the query:

2.2.2.1 Type Name: Results

Results defined in this document are referenced under the slpf namespace.

Base Type: Map

ID	Name	Type	#	Description
1	access_rules	Rule-Item	0…n	Return the matching access rules

2.2.2.2 Type Name: Rule-Item

Base Type: Array

ID	Name	Type	#	Description
1	id	Integer	1	Rule identifier
2	action	Rule-Action	1	Allow or Deny
2	rule	Rule-Target	1	Matching criterion

2.2.2.3 Type Name: Rule-Action

Base Type: Enumerated

ID	Name	Description
1	allow	Matching packets are permitted
2	deny	Matching packets are blocked

2.2.2.4 Type Name: Rule-Target

Base Type: Choice

ID	Name	Type	Description
1	ip_addr	IP-Addr	Rules affecting the specified address
2	ip_connection	IP-Connection	Rules affecting the specified 5-tuple

2.2.2.5 Example Response

```
{    "status": 200,
     "results": {
         "slpf": {
             "access_rules": [
                 [30, "allow", {"ip_addr": "1.2.3.4"}],
                 [52, "deny", {"ip_addr": "1.2.3.0/24"}],
                 [0, "allow", {"ip_addr": "*"}]]
}}}
```

2.3 OpenC2 Commands

An OpenC2 command consists of an ACTION/TARGET pair and associated specifiers and arguments. This section will enumerate the allowed commands, identify which are required to implement and present the associated responses.

Table 2.3-1 defines the commands allowed by the SLPF profile and indicates if implementation of the command is required or optional for openc2 producers and/or openc2 consumers. An action (the top row in Table 2.3-1) paired with a target (the first column in Table 2.3-1) defines an allowable command. The subsequent subsections provide the property tables applicable to each OpenC2 command.

Table 2.3-1. Command Matrix

	Allow	Deny	Query	Delete	Update
ip_connection	required	required			
ip_addr	required	required			
openc2			required		
slpf:access_rules			optional	optional	

	Allow	Deny	Query	Delete	Update
file					optional

**Table 2.3-2 defines the command arguments that are allowed for a particular command by the SLPF profile. A command (the top row in Table 2.3-2) paired with an argument (the first column in Table 2.3-2) defines an allowable combination. The subsection identified at the intersection of the command/ argument provides details applicable to each command as influenced by the argument. **

Table 2.3-2. Command Arguments Matrix

	Allow <target>	Deny <target>	Query openc2	Query access_rules	Delete access_rules	Update file
response	2.3.1	2.3.2	2.3.3.1	2.3.3.2	2.3.4.1	2.3.5.1
start-time	2.3.1	2.3.2			2.3.4.1	2.3.5.1
end-time	2.3.1	2.3.2				
duration	2.3.1	2.3.2				
respond-to	2.3.1	2.3.2	2.3.3.1	2.3.3.2	2.3.4.1	2.3.5.1
running	2.3.1	2.3.2				
direction	2.3.1	2.3.2				
prepend	2.3.1	2.3.2				
insert_rule	2.3.1	2.3.2				
drop_process		2.3.2				

2.3.1 ‘Allow’

Table 2.3.1-1 summarizes the command options that apply to all of the commands consisting of the ‘allow’ action and a valid target type.

Upon receipt of an unsupported command argument, SLPF consumers

- MUST NOT respond with a OK/200
- MUST respond with the 501 error code
- MUST respond with “Option not supported” in the error description.

Products that send ‘allow target’ commands and support the ‘delete slpf:access_rules’ command:

- MUST support the slpf:access_rules target type as defined in section 2.1.2
- SHOULD populate the command options field with "response-type="complete”
- MAY populate the command arguments field with the ‘prepend’ option
- MAY populate the command arguments field with the ‘insert_rule=INT’ option where INT is an integer.
- MUST populate the command options field with ‘response_type=complete’ if the insert_rule argument is populated.

Products that receive ‘allow target’ commands and support the ‘delete slpf:access_rules’ command:

- MUST support the slpf:access_rules target type as defined in section 2.3.3.2
- MUST return the rule number assigned to the rule in the slpf:access_rules object if the “response-type= “complete” option is populated.

Products that receive ‘allow target’ commands and support the ‘insert_rule’ command argument:

- MUST assign the rule number provided if the ‘insert_rule=INT’ option is populated.
- If the rule number is currently in use, then MUST respond with the 501 error code and SHOULD respond with ‘Rule number currently in use’ in the error description

The valid target types, associated specifiers, and options are summarized in sections 2.3.1.1 and 2.3.1.2. Sample commands are presented in section 3.

2.3.2 ‘Allow ip_connection’

The ‘allow ip_connection’ command is required for openc2 producers implementing the SLPF.

If the ‘allow ip_addr’ target is not implemented, then SLPF consumers MUST implement the ‘allow ip-connection’ command. Otherwise it is OPTIONAL.

The command permits traffic that is consistent with the specified ip_connection. A valid ‘allow ip-connection’ command has at least one property of the ip_connection populated and may have any combination of the five properties populated. An unpopulated property within the the ip_connection target must be treated as an ‘any’.

Products that do not implement the ‘allow ip_connection’ command MUST respond with the 501 response code and SHOULD respond with ‘Target type not supported’ in the error description

2.3.3 ‘Allow ip-addr’

The ‘allow ip_addr’ command is required for openc2 producers implementing the SLPF.

If the ‘allow ip_connection’ target is not implemented, then SLPF consumers MUST implement the ‘allow ip_addr’ command. Otherwise the ‘allow ip-addr’ command is OPTIONAL.

The command permits traffic as specified by the ip_addr property and may be an IPV4 or IPV6 address. The ip-addr supports CIDR notation. The address specified in the ip_addr MUST be treated as a source OR destination address.

Products that do not implement the ‘allow ip-addr’ command MUST respond with the 501 response code and SHOULD respond with ‘Target type not supported’ in the error description.

2.3.4 ‘Deny’

‘Deny’ can be treated as mathematical complement to ‘allow’. With the exception of the additional ‘drop_process’ actuator-argument, the targets, specifiers, options and corresponding responses are identical to the two ‘allow’ commands. Table 2.3.2-1 summarizes the command arguments that apply to all of the commands consisting of the ‘deny’ action and valid target type.

Upon receipt of a command with an ARGUMENT that is not supported by the actuator, actuators MUST respond with the 501 error code and MUST respond with ‘Option not supported’ in the error description.

Products that send ‘deny target’ commands and support the ‘delete slpf:access_rules’ command:

- MUST support the slpf:access_rules target type as defined in section 2.3.3.2
- SHOULD populate the command options field with ‘response-type="complete” ’
- MAY populate the command arguments field with the ‘prepend’ option
- MAY populate the command arguments field with the ‘insert_rule=INT’ option where INT is an integer AND the prepend option is not present.
- MUST populate the command options field with ‘response_type=complete’ if the insert_rule argument is populated.

Products that receive ‘deny target’ commands and support the ‘delete slpf:access_rules’ command:

- MUST support the slpf:access_rules target type as defined in section 2.3.3.2
- MUST return the rule number assigned in the slpf object if the ‘response-type= “complete” ’ option is populated.

Products that receive ‘deny target’ commands and support the ‘insert_rule’ command argument:

- MUST assign the rule number provided if the ‘insert_rule=INT’ option is populated.
- If the rule number is currently in use, then MUST respond with the 501 error code and SHOULD respond with ‘Rule number currently in use’ in the error description

2.3.5 ‘Query’

The valid target types, associated specifiers, and options are summarized in sections 2.2.3.1 through 2.2.3.2. Sample commands are presented in Section 5 appendix A.

2.3.6 ‘Query openc2’

The ‘query openc2’ command MUST be implemented in accordance with Version 1.0 of the OpenC2 language specification.

2.3.7 ‘Query slpf:access_rules’

The ‘query slpf:access_rules’ command is used to determine the current settings of the SLPF. Implementation of the ‘query slpf:access_rules’ command is OPTIONAL.

Products that send the ‘query slpf:access_rules’ command:

- MUST populate the command options field with ‘response-type="complete”.
- MAY populate the respond-to argument.
- MUST NOT include other command arguments.
- MAY include one or more of the slpf:access_rules specifiers identified in table 2.2.1.2.

Products that receive the ‘query slpf:access_rules’ command:

- MUST respond with response code 400 if the command cannot be parsed or processed.
- MUST respond with error code 501 and MAY respond with error description ‘target type not supported’ if the product does not support the slpf:access_rules target type
- Upon successful parsing and processing of the ‘query slpf:access_rules’ command, products MUST respond with response code 200 and populate the results field with the slpf:access_rules target type and associated specifiers identified in the command.
- If one or more ip-connection objects were explicitly listed in the allow-rules specifier, then the actuator returns the ip-connection if it is explicitly allowed. An empty string implies that the ip-connection is not allowed.
- If one or more ip-connection objects were explicitly listed in the deny-rules specifier, then the actuator returns the ip-connection if it is explicitly denied. An empty string implies that the ip-connection is not denied.
- If no ip-connection objects were explicitly listed in the allow-rules specifier, then the actuator MUST return the complete list of ip-connection objects that are explicitly allowed.

- If no ip-connection objects were explicitly listed in the deny-rules specifier, then the actuator MUST return the complete list of ip-connection objects that are explicitly denied.
- If no specifiers were identified in the query slpf:access_rules command, then the results field MUST return the entire allow and deny list.

Refer to Annex 3 for sample commands.

2.3.8 'Delete'

The slpf:access_rules is the only valid target type for the delete action. The associated specifiers, and options are summarized in section 2.2.4.1. Sample commands are presented in Annex 3.

2.3.9 'delete slpf:access_rules'

The 'delete slpf:access_rules' command is used to remove a firewall rule rather than issue another allow or deny to counteract the effect of an existing rule. Implementation of the 'delete slpf:access_rules' command is OPTIONAL. Products that choose to implement the 'delete slpf:access_rules' command MUST implement the slpf:access_rules target type described in section 2.3.3.1.

Products that send the 'delete slpf:access_rules' command:

- MAY populate the command options field with 'response-type="complete".
- MAY populate the respond-to argument.
- MUST NOT include other command arguments.
- MUST include exactly one deny-rule OR allow-rule as described in table 2.2.1.2.
- The deny-rule OR allow-rule MUST include the rule number and MAY include the corresponding ip_connection or ip_addr.

Products that receive the 'delete slpf:access_rules' command:

- but cannot parse or process the 'delete slpf' command MUST respond with response code 400.
- but do not support the slpf:access_rules target type MUST respond with error code 501 and SHOULD respond with error description 'command not supported'
- MUST respond with response code 200 upon successful parsing of the 'delete slpf:access_rules' command and subsequent removal of the corresponding rule,
- MUST respond with response code 500 upon successful parsing but failure to remove the corresponding rule
- SHOULD respond with error description 'firewall rule not removed or updated' upon successful parsing but failure to remove the corresponding rule.

Refer to Annex 3 for sample commands.

2.3.10 Update

The ‘file’ target as defined in Version 1.0 of the Language Specification is the only valid target type for the delete action. The associated specifiers, and options are summarized in section 2.3.4.1. Sample commands are presented in Annex 3.

2.3.11 Update file

The ‘update file’ command is used to replace or update files such as configuration files, rule sets, etc. Implementation of the update file command is OPTIONAL. OpenC2 consumers that choose to implement the ‘update file’ command MUST must include all steps that are required for the update file procedure such as retrieving the file(s), install the file(s), restart/ reboot the device etc. The end state shall be that the firewall operates with the new file at the conclusion of the ‘update file’ command. The atomic steps that take place are implementation specific.

Table 2.3-2 presents the valid options for the ‘update file’ command. Products that choose to implement the ‘update file’ command MUST NOT include options other than the options identified in table 2.3-2

Products that send the ‘update file’ command:

- MAY populate the command options field with with the ‘response-type’. ‘Complete’, ‘Ack’ and ‘None’ are valid response types for ‘update file’
- MAY populate the respond-to option.
- MUST NOT include other command options.
- MUST populate the name specifier in the target.
- SHOULD populate the path specifier in the target.

Products that receive the ‘update file’ command:

- but cannot parse or process the command MUST respond with response code 400.
- but do not support the ‘update file’ command type MUST respond with error code 501 and MAY respond with error description ‘command not supported’
- but cannot access the file specified in the file target MUST respond with response code 500 and SHOULD respond with error description ‘cannot access file’.
- upon successful parsing and initiating the processing of the ‘update file’ command, products MAY respond with response code 102.
- upon completion of all the steps necessary to complete the update and the actuator commences operations functioning with the new file, actuators products SHOULD respond with response code 200.

  Refer to Annex 3 for sample commands.

3 Conformance statements

Entities that send commands to and receive responses from entities with StateLess Packet Filtering capabilities are known as OpenC2 SLPF Producers.

Entities with StateLess Packet Filtering capabilities that receive commands and send responses are known as OpenC2 SLPF consumers.

3.1 Conformance Clause 1: OpenC2 SLPF Entities (both producers and consumers)

- MUST support JSON serialization of OpenC2 commands that are syntactically valid in accordance with the JADN schema presented in Annex 1.
- MAY support other serializations provided the serialization is validated against, and provides a one-to-one mapping to, the JADN schema in Annex 1 and Annex 2 of this specification
- MUST support the use of one or more published OpenC2 transfer specifications which identify underlying transport protocols such that an authenticated, ordered, lossless, delivery of uniquely identified OpenC2 messages is provided.
- SHOULD support the OpenC2 HTTPS transfer specification
- MUST be conformant with Version 1.0 of the Language Specification
- MUST implement the following action target pairs where the actions and targets are defined in version 1.0 of the Language specification.
    - OpenC2 SLPF producers:
        - ‘allow ip_connection’ AND ‘allow ip_addr’ in accordance with the normative text provided in section 2.3.1 of this specification
        - ‘deny ip_connection’ AND ‘deny ip_addr’ in accordance with the normative text provided in section 2.3.2 of this specification
    - OpenC2 SLPF consumers:
        - ‘allow ip_connection’ OR ‘allow ip_addr’ in accordance with the normative text provided in section 2.3.1 of this specification
        - ‘deny ip_connection’ OR ‘deny ip_addr’ in accordance with the normative text provided in section 2.3.2 of this specification
    - ‘query openc2’ in accordance with the normative text provided in version 1.0 of the OpenC2 Language Specification.
- MAY implement the following action target pairs where version 1.0 of the Language specification defines the ‘file’ target and actions; and the ‘slpf:access_rules’ target type is defined in this specification
    - ‘query slpf:access_rules’ in accordance with the normative text provided in section 2.3.3.2 of this specification
    - ‘delete slpf:access_rules’ in accordance with the normative text provided in section 2.3.4.1 of this specification

- ‘update file’ in accordance with the normative text provided in section 2.3.5.1 of this specification

3.2 Conformance Clause 2: OpenC2 Stateless Packet Filter Producer: Command Arguments

MUST implement the ‘response_type’ command argument that MAY be included with any command:

MAY implement the ‘respond_to’ command argument that MAY be included with any command. The ‘respond_to’ command argument MUST be used to designate any recipient other than the the command producer.

MAY implement the start_time command argument that MAY be included with any command other than ‘query <target>’

MAY implement the following command arguments that MAY be included with any command other than ‘query <target>’ and ‘update file’

- end_time
- duration

MAY implement the following command arguments that MAY be included with ‘allow <target>’ and/or ‘deny <target>’ commands

- running
- direction
- prepend

MAY implement the drop_process command arguments that MAY be included with ‘deny <target>’ command

3.3 Conformance Clause 3: OpenC2 Stateless Packet Filter Consumer: Processing and Response to Commands and command arguments

- All commands received with the response argument set to ‘none’ MUST process the command and MUST NOT send a response. This conformance clause supersedes all other normative text as it pertains to responses.
- All commands received without the response argument (or response argument not set) MUST process the command and MUST NOT send a response.

- OpenC2 consumers MAY support the ‘respond_to’ command argument and send the response as designated in the ‘respond_to’ command argument. If the respond_to command argument is not present, not supported nor populated, then the response MUST be sent to the command producer.

Editor’s Note - There is discussion within the Language SC regarding ‘default’ response type. Currently is ‘none’ but may change to ‘complete’ The second bullet will need to be modified accordingly

4 Annex 1 SLPF Schema

This annex defines the data objects used by conforming SLPF implementations, as shown in Section 2. This annex is normative.

```
{ "meta": {
    "module": "oasis-open.org/openc2/v1.0/ap-slpf",
    "title": "Stateless Packet Filtering",
    "version": "wd02",
    "description": "Data definitions for Stateless Packet Filtering
(SLPF) functions",
    "exports": ["Target", "Specifiers", "Args", "Results"]
  },

  "types": [
    ["Target", "Choice", [], "SLPF targets", [
      [1, "access_rules", "Rules", [], "Return the requested access
rules in the response"]]
    ],
    ["Rules", "Map", [], "Specifiers for the access_rules target", [
      [1, "allow_rules", "Rule-Target", ["[0"], "Return specified
allow rules"],
      [2, "deny_rules", "Rule-Target", ["[0"], "Return specified deny
rules"]]
    ],
    ["Rule-Target", "Choice", [], "Specifier for an access rule", [
      [1, "all", "Null", [], "Any rules"],
      [2, "ip_addr", "..:IP-Addr", ["[0"], "Rules affecting the
specified address"],
      [3, "ip_connection", "..:IP-Connection", ["[0"], "Rules
affecting the specified 5-tuple"]]
    ],

    ["Specifiers", "Map", [], "SLPF actuator specifiers", [
      [1, "nfv_id", "String", [], "Identifier of a virtualized packet
filter"]]
    ],

    ["Args", "Map", [], "SLPF command arguments", [
      [1, "disposition", "Disposition", [], "How to handle denied
packets"],
      [2, "insert", "Integer", [], "Position of a rule in access
list"]]
    ],
    ["Disposition", "Enumerated", [], "", [
      [1, "drop", ""],
```

```
      [2, "reject", ""],
      [3, "deceive", ""]]

    ["Results", "Map", [], "SLPF results", [
      [1, "access_rules", "Rule-Item", ["[0", "]0"], "Access rules
matching a query"]]
    ],
    ["Rule-Item", "Array", [], "", [
      [1, "id", "Integer", [], "Rule identifier within a ruleset"],
      [2, "action", "Rule-Action", [], "Allow or Deny"],
      [3, "rule", "Rule-Target", [], "Matching criterion"]]
    ],
    ["Rule-Action", "Enumerated", [], "", [
      [1, "allow", "Matching packets are permitted"],
      [2, "deny", "Matching packets are blocked"]]
    ]]
 }
```

5 Annex 2 OpenC2 Schema

This annex is a copy of the schema from the OpenC2 Language Specification trimmed to include only elements needed to support the SLPF functions defined in this document. This subset defines the elements of the Language Specification that are meaningful in the context of SLPF, however an implementation may have capabilities beyond the scope of an SLPF therefore will support additional elements of the OpenC2 language beyond those included here.

This annex is normative, however in the event of a conflict with the schema in the OpenC2 Language Specification, the Language Specification is authoritative.

```
{
 "meta": {
  "module": "/oasis-open.org/openc2/v1.0/openc2-lang",
  "title": "OpenC2 Language Objects",
  "version": "wd07-ap-slpf",
  "description": "OpenC2 datatypes used by the Stateless Packet
Filter profile.",
  "imports": [
    ["slpf", "/oasis-open.org/openc2/v1.0/ap-slpf"],
    ["jadn", "/oasis-open.org/openc2/v1.0/jadn"]
  ],
  "exports": ["OpenC2-Command", "OpenC2-Response"]
 },
 "types": [
  ["OpenC2-Command", "Record", [], "An action performed on a
target.", [
    [1, "action", "Action", [], "The task or activity to be
performed"],
    [2, "target", "Target", [], "The object of the action."],
    [3, "actuator", "Actuator", ["[0"], "The subject of the
action."],
    [4, "args", "Args", ["[0"], "Additional information that applies
to the command"],
    [5, "id", "Command-ID", ["[0"], "Identifier used to link
responses to a command"]]
  ],
  ["Action", "Enumerated", [], "", [
    [3, "query", "Initiate a request for information."],
    [6, "deny", "Prevent a certain event or action from completion"],
    [8, "allow", "Permit access to or execution of a target."],
    [16, "update", "Instruct a component to install an update."],
    [20, "delete", "Remove an entity (e.g., data, files, flows."]]
  ],
```

```
  ["Target", "Choice", [], "OpenC2 Target datatypes", [
    [10, "file", "File", [], "Properties of a file"],
    [11, "ip_addr", "IP-Addr", [], "One or more IP addresses (either
v4 or v6)"],
    [15, "ip_connection", "IP-Connection", [], "A network
connection"],
    [16, "openc2", "OpenC2", [], "Query target to determine an
actuator's capabilities"],
    [1024, "slpf", "slpf:Target", [], "SLPF Targets"]]
  ],
  ["Actuator", "Choice", [], "", [
    [1, "generic", "Specifiers", [], "Generic actuator specifiers"],
    [1024, "slpf", "slpf:Specifiers", [], "SLPF actuator
specifiers"]]
  ],
  ["Specifiers", "Map", [], "", [
    [1, "actuator_id", "URI", ["[0"], "Network locater of the
actuator"],
    [2, "asset_id", "String", ["[0"], "Hardware identifier of an
actuator device"]]
  ],
  ["Args", "Map", [], "", [
    [1, "start_time", "Date-Time", ["[0"], "The specific date/time to
initiate the action"],
    [2, "stop_time", "Date-Time", ["[0"], "The specific date/time to
terminate the action"],
    [3, "duration", "Duration", ["[0"], "The length of time for an
action to be in effect"],
    [4, "response_requested", "Response-Type", ["[0"], "The type of
response required for the action"],
    [1024, "slpf", "slpf:Args", ["[0"], "SLPF Command arguments"]]
  ],
  ["OpenC2-Response", "Record", [], "", [
    [1, "id", "Command-ID", [], "Id of the command that induced this
response"],
    [2, "status", "Status-Code", [], "An integer status code"],
    [3, "status_text", "String", ["[0"], "A free-form human-readable
description of the response status"],
    [4, "*", "Results", [], "Data or extended status information that
was requested from an OpenC2 command"]]
  ],
  ["Status-Code", "Enumerated", ["="], "", [
    [102, "Processing", "An interim response used to inform the
client that the server has accepted the request but not yet completed
it."],
    [200, "OK", "The request has succeeded."],
    [301, "Moved Permanently", "The target resource has been assigned
a new permanent URI"],
```

```
    [400, "Bad Request", "The server cannot process the request due
to something that is perceived to be a client error (e.g., malformed
request syntax.)"],
    [401, "Unauthorized", "The request lacks valid authentication
credentials for the target resources or authorization has been
refused for the submitted credentials."],
    [403, "Forbidden", "The server understood the request but refuses
to authorize it."],
    [500, "Server Error", "The server encountered an unexpected
condition that prevented it from fulfilling the request."],
    [501, "Not Implemented", "The server does not support the
functionality required to fulfill the request."]]
  ],
  ["OpenC2", "ArrayOf", ["*Query-Item", "[0", "]3"], "Query Actuator
capabilities"],
  ["Query-Item", "Enumerated", [], "Results to be included in
response to query openc2 command", [
    [1, "versions", "OpenC2 language versions supported by this
actuator"],
    [2, "profiles", "List of profiles supported by this actuator"],
    [3, "schema", "Definition of the command syntax supported by this
actuator"]]
  ],
  ["IP-Connection", "Record", [], "5-tuple that specifies a tcp/ip
connection", [
    [1, "src_addr", "IP-Addr", ["[0"], "source address"],
    [2, "src_port", "Port", ["[0"], "source TCP/UDP port number"],
    [3, "dst_addr", "IP-Addr", ["[0"], "destination address"],
    [4, "dst_port", "Port", ["[0"], "destination TCP/UDP port
number"],
    [5, "layer4-protocol", "L4-Protocol", ["[0"], "Protocol (IPv4) /
Next Header (IPv6)"]]
  ],
  ["L4-Protocol", "Enumerated", [], "protocol (IPv4) or next header
(IPv6) field - any IANA value, RFC 5237", [
    [1, "icmp", "Internet Control Message Protocol - RFC 792"],
    [6, "tcp", "Transmission Control Protocol - RFC 793"],
    [17, "udp", "User Datagram Protocol - RFC 768"],
    [132, "sctp", "Stream Control Transmission Protocol - RFC 4960"]]
  ],
  ["File", "Map", [], "", [
    [1, "name", "String", ["[0"], "The name of the file as defined in
the file system"],
    [2, "path", "String", ["[0"], "The absolute path to the location
of the file in the file system"],
    [3, "hashes", "Hashes", ["[0"], "One or more cryptographic hash
codes of the file contents"]]
   ],
  ["Response-Type", "Enumerated", [], "", [
```

```
    [0, "none", "No response"],
    [1, "ack", "Respond when command received"],
    [2, "status", "Respond with progress and when command is
complete"],
    [3, "complete", "Respond when all aspects of command completed"]]
  ],
  ["Hashes", "Map", [], "Cryptographic Hash values", [
    [1, "md5", "Binary", ["[0"], "Hex-encoded MD5 hash as defined in
RFC3121"],
    [4, "sha1", "Binary", ["[0"], "Hex-encoded SHA1 hash as defined
in RFC3174"],
    [6, "sha256", "Binary", ["[0"], "Hex-encoded SHA256 as defined in
RFC6234"]]
  ],
  ["Command-ID", "String", [], "Uniquely identifies a particular
command - TBD syntax"],
  ["Date-Time", "String", ["@date-time"], "RFC 3339 date-time"],
  ["Duration", "String", ["@duration"], "RFC 3339 / ISO 8601
duration"],
  ["Domain-Name", "String", ["@hostname"], "Domain name, RFC 1034,
section 3.5"],
  ["Email-Addr", "String", ["@email"], "Email address, RFC 5322,
section 3.4.1"],
  ["IP-Addr", "String", ["@ip"], "IPv4 or IPv6 address"],
  ["Port", "String", ["@port"], "Service Name or Transport Protocol
Port Number, RFC 6335"],
  ["Version", "String", [], "Version string - TBD syntax"],
  ["URI", "String", ["@uri"], "Uniform Resource Identifier"]]
}
```

6 Annex 3 Sample commands (Informative)

Editor's Note - PLEASE NOTE: This section is incomplete, do not code to this format. This section is a placeholder. The syntax of the sample commands all must be reworked in accordance with the changes that occured in the language specification.

This section will summarize and provide examples of OpenC2 commands as they pertain to SLPF firewalls. The sample commands will be encoded in verbose JSON, however other encodings are possible provided the command is validated against the schema presented in Appendix A. Examples of corresponding responses and/or alerts will be provided where appropriate.

The samples provided in this section are for illustrative purposes only and are not to be interpreted as operational examples for actual systems. Within the scope of this document, a # character indicates a comment, however it should be noted that OpenC2 itself does not support comments within a command.

6.1 Deny and Allow

Deny and allow are mandatory to implement and can be treated as mathematical complements of each other. Unless otherwise stated, the example targets, specifiers, modifiers and corresponding responses are applicable to both actions.

```
# Block a particular connection within the domain and do not send a
host unreachable
{
    "action": "deny",
    "target": {
        "ip_connection": {
            "layer4_protocol": "TCP",
            "src_addr": {"1.2.3.4"},
            "src_port": {10996},
            "dst_addr": {"198.2.3.4"},
            "dst_port": {80}
        }
    },
    "actuator": {
        "firewall": {
            "asset_id": "30"
            Options:{drop}
        }
```

```
    },
    "command-options": {
        ,
        "start_time": "2016-11-25T08:10:31-04:00",
        "duration": 600,
        "command_id": "fw17_8675309"
    }
}
# Block all ftp data transfers from hosts, send false acknowledgement
and request ack. Note that the five-tuple is incomplete
{
    "action": "deny",
    "target": {
        "type": ":ip_connection",
        "specifiers": {
            "Layer4Protocol": "TCP",
            "src-port": 21
        }
    }
    "actuator": {
        "type": "openc2:firewall",
        "specifiers": {endpoint},
        "options":{complete}
    },
    "command-options": {
        {"id":"UUID=123e4567-e89b-12d3-a456-426655440000"}
        {response="Ack”}
    }
}
# Note that the response was requested and all endpoints that can
execute the command should.
# In this case, one of the endpoints successfully issued the deny but
the endpoint located at 198.51.100.17 failed
{
    response
        {Source: ip-addr=198.51.100.17}
        {cmdref=123e4567-e89b-12d3-a456-426655440000 }
        {statuscode=200}
}
{
    response
        {Source: ip-addr=198.51.100.18}
        {cmdref=123e4567-e89b-12d3-a456-426655440000 }
        {statuscode=400}
}
# Allow ftp data transfers to a particular ip address from any host.
Note that the five-tuple is incomplete
{
    "action": "allow",
```

```
    "target": {
        "type": "ip-connection",
        "specifiers": {
            "Layer4Protocol": "TCP",
            "dst-addr": 198.51.100.17
            "src-port": 21
        }
    }
      "command-options": {
            "id":"UUID=123e4567-e89b-12d3-a456-426655440000"
            "response":”Ack”
        }
    }
}

#
{
    response
        {Source: openc2:ip-addr=1.2.3.4}
        {cmdref=123e4567-e89b-12d3-a456-426655440000 }
        {statuscode=200}
}
```

6.2 Update

Implementation of the Update action is optional. Update is intended for the device to process new configuration files, software updates, patches, policy updates etc. The update action is a compound action in that all of the steps required for a successful update (such as download the new file, install the file, reboot etc.) are implied. File is the only valid target type for Update.

```
# instructs the firewalls to acquire a new configuration file. Note
that all network based firewalls will install the new update because
no particular firewall was identified. Host based firewalls will not
act on this because network firewalls were identified as the
actuator.

{
    "action": "update",
    "target": {
        "file": {
            "parent_directory": {
            "path":"\\\\someshared-
drive\\somedirectory\\configurations"},
            "name": "firewallconfiguration.txt"
        }
    },
    "actuator": {
        "openc2:firewall": {
            "named-group":"network"
```

```
        }
    }
    "Command-options" {
        Command-id:123e4567-e89b-12d3-a456-426655440000
    }
}
Note the responses.  One of the devices successfully updated the
configuration file.  Another device responded with an error because
it does not support the 'update file' command

{
    response
        {Source: openc2:ip-addr=1.2.3.4}
        {cmdref=123e4567-e89b-12d3-a456-426655440000 }
        {statuscode=200}
}

{
    response
        {Source: openc2:ip-addr=1.2.3.5}
        {cmdref=123e4567-e89b-12d3-a456-426655440000 }
        {statuscode=501}
        {statustext:"command not supported"}
}
# Instructs any splf running a particular software load to install a
software upgrade
{
    "action": "update",
    "target": {
        "file": {
            "parent_directory": {
                "path": "\\\\someshared-drive\\somedirectory\\so"
            },
            "name": " version2.offirewallsoftware.exe"
        }
    },
    "actuator": {
        "openc2:slpf": {
            "x-tagID": "firewallcompanyversion2.1"
        }
    }
}
```

6.3 Query

The query action is used to gather some set of information from the SLPF. There are two valid target types for the query command; 'openc2' and 'slpf:access_rules'.

Implementation of query openc2 is required. The query openc2 command is intended to enable the openc2 producer to determine the capabilities of the actuator. The query openc2 command can also be used to check the status of the actuator.

```
# This thread illustrates the use of query openc2 to verify that the
actuator
# is functioning and/or determine the version of the language
specification.
# Note that the specifier is identified in the command but the value
is not
# populated.
{
    "Id"=12345
    "action": "query"
    "target": {
        "openc2" {
            Version:
        }
    }
    "Options" {
        Response_type=complete
    }
}

#  The corresponding response would be:
#  Case one, things are OK

    "Id":12345
    "Status":200
    Results {
        Type: openc2 {
            "Version": 1.0
        }
    }

#  Case two, error state
    "Id":12345
    "Status":400
    Results{}

# This thread illustrates the use of the query openc2 command to
determine
# the profiles supported and version of openc2.

{
    "Id"=12345
    "action": "query"
    "target": {
        "openc2" {
```

```
            Version:
            Profiles:
        }
    }
    "options" {
        Response_type=complete
    }
}

# The corresponding response would be:
# Case one, things are OK
    "Id":12345
    "Status":200
    Results {
        Type: openc2 {
            "Version": 1.0
            "profiles": stateless-packet-filtering,[
http://some.url.org](http://some.url.org)
        }
    }
#  Case two, error
    "Id":12345
    "Status":400
    Results{}

# Case two:  We know the version and profile, but want to know the
options
#  profile-options is present but not populated
{
    "Id"=12345
    "action": "query"
    "target": {
        "openc2" {
            Profile-options
        }
    }
    "options" {
        Response_type=complete}
    }

# The corresponding response would be:
# response one, things are OK
    "Id":12345
    "Status":200
    Results {
        Type: openc2 {
            Profile-options:
                stateless-packet-filtering, deny domain-name,
complete
```

```
                stateless-packet-filtering, deny domain-name,
respond-to
                stateless-packet-filtering, deny hostname, complete
                stateless-packet-filtering, deny hostname, respond-to
        }
    }

#  Response two, error
    "Id":12345
    "Status":400
    Results {}

# Case three:  In this case, we don't know anything and want a
complete report from the actuator.
#  None of the specifiers for the openc2 target type are present, so
the actuator responds with all of the specifiers populated
{
    "Id"=12345
    "action": "query"
    "target": {
        "openc2" {}
    }
    "options" {
        Response_type=complete}
    }

# The corresponding response would be:
# response one, things are OK
    "Id":12345
    "Status":200
    Results{
        Type: openc2{
            Version: 1.0
            "Profiles":
                stateless-packet-filtering,[
http://some.url.org](http://some.url.org)
                our-next-oasis-profile,[
http://someother.url.org](http://someother.url.org)
                  Profile-options:
                    stateless-packet-filtering, deny domain-name,
complete
                    stateless-packet-filtering, deny domain-name,
respond-to
                    stateless-packet-filtering, deny hostname,
complete
                    stateless-packet-filtering, deny hostname,
respond-to
                our-next-oasis-profile, scan directory, act-specific-
option
```

```
                our-next-oasis-profile, scan file, actuator-specific-
option
# note in the next entry, we had an optional action target pair, but
no options associated with it
    our-next-oasis-profile, contain username,
# note in the next entry, we have a command option for all profiles
for all actions
,,respond-to
Serialization:
Json
Cbor
XML

}
}
#  Response two, error
    "Id":12345
    "Status":400
    Results{}

# Implementation of query command is optional.  The query
slpf:access_rules is intended to enable the openc2 producer to
determine the current state of the actuator.

# Verify that the filters are blocking all outbound ftp traffic and
TELNET traffic
{
    "Id"=12345
    "action": "query"
    "target": {
        "" {
            Deny-rules: {
                specifiers {
                    src-port:21
                }
                specifiers {
                    dst-port:23
                }
            }
        }
    }
    "Options" {
        Response_type=complete
    }
}

#  The corresponding response would be:
#  Case one, things are OK and both ip-connections are denied
    "Id":12345
```

```
    "Status":200
    Results {
        Type: slpf:access_rules {
            {src-port:21}
            {dst-port:23}
        }
    }
#  Case two, things are OK.  Telnet is being denied but outbound ftp
is notboth ip-connections are denied
    "Id":12345
    "Status":200
    Results {
        Type: {
            dst-port:23
        }
    }
#  Case three, not supported
    "Id":12345
    "Status":501
    Results {}
```