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

Committee Specification Draft ~~04~~05 / Public Review Draft ~~01~~02

~~17 October 2018~~

~~Specification URIs~~

04 April 2019

This version:

https://docs.oasis-open.org/openc2/oc2slpf/v1.0/csprd02/oc2slpf-v1.0-csprd02.md (Authoritative)
https://docs.oasis-open.org/openc2/oc2slpf/v1.0/csprd02/oc2slpf-v1.0-csprd02.html
https://docs.oasis-open.org/openc2/oc2slpf/v1.0/csprd02/oc2slpf-v1.0-csprd02.pdf

Previous version:

http://docs.oasis-open.org/openc2/oc2slpf/v1.0/csprd01/oc2slpf-v1.0-csprd01.md (Authoritative)
http://docs.oasis-open.org/openc2/oc2slpf/v1.0/csprd01/oc2slpf-v1.0-csprd01.html
http://docs.oasis-open.org/openc2/oc2slpf/v1.0/csprd01/oc2slpf-v1.0-csprd01.pdf

~~Previous version:~~

- ~~(Authoritative)~~

Latest version:

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

~~**Additional artifacts:**~~

~~This prose specification is one component of a Work Product that also includes:~~

- ~~SLPF schema ():~~
  - ~~(authoritative)~~
  - ~~(formatted)~~
- ~~Tailored OpenC2 schema ():~~
  - ~~(example)~~
  - ~~(formatted)~~
- ~~Merged schema example ():~~

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~~Actions, Targets, Specifiers and ~~o~~Options that are consistent with the version 1.0 of the OpenC2 Language Specification ([OpenC2-Lang-v1.0]) in the context of stateless packet filtering~~.~~ (SLPF).

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"~~"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. ~~17 October 2018.~~04 April 2019. OASIS Committee Specification Draft ~~04~~05 / Public Review Draft ~~01~~02. https://docs.oasis-open.org/openc2/oc2slpf/v1.0/csprd02/oc2slpf-v1.0-csprd02.html. Latest version: https://docs.oasis-open.org/openc2/oc2slpf/v1.0/oc2slpf-v1.0.html.

Notices

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

The content in this section is non-normative, except where it is marked normative.

OpenC2 is a suite of specifications that enables command and control of cyber defense systems and components. OpenC2 typically uses a request-response paradigm where a ~~c~~*Command* is encoded by ~~an OpenC2 producer~~a *Producer* (managing application) and transferred to ~~an OpenC2 consumer~~a *Consumer* (managed device or virtualized function) using a secure transpo~~fer~~r~~t~~ protocol, and the ~~c~~Consumer can respond with status and any requested information~~. The contents of both the command and the response are fully described in schemas, allowing both parties to recognize the syntax constraints imposed on the exchange~~.

OpenC2 allows the application producing the commands to discover the set of capabilities supported by the managed devices. These capabilities permit the managing application to adjust its behavior to take advantage of the features exposed by the managed device. The capability definitions can be easily extended in a noncentralized manner, allowing standard and non-standard capabilities to be defined with semantic and syntactic rigor.

1.1 IPR Policy

This specification is provided under the Non-Assertion Mode of the OASIS IPR Policy~~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 Terminology

This section is normative.

- **Action**: The task or activity to be performed~~.~~ (e.g., 'deny').

- **Actuator**: The ~~entity~~function performed by the Consumer that ~~performs~~executes the ~~action.~~

    - Command (e.g., 'Stateless Packet Filtering').
    - **Argument**: A ~~message~~property of a Command that provides additional information on how to perform the Command, such as date/time, periodicity, duration, etc.
    - **Command**: A Message defined by an ~~action-target~~Action-Target pair that is sent from a ~~p~~Producer and received by a ~~c~~Consumer.
    - **Consumer**: A managed device / application that receives Commands. Note that a single device / application can have both ~~c~~Consumer and ~~p~~Producer capabilities.
    - **Message**: A content- and transport-independent set of elements conveyed between Consumers and Producers
    - **Producer**: A manager application that sends Commands.
    - **Response**: A ~~m~~Message from a ~~c~~Consumer to a ~~p~~Producer acknowledging a ~~c~~Command or returning the requested resources or status to a previously received request.
    - **Specifier**: A property or field that identifies a Target or Actuator to some level of precision.
    - **Target**: The object of the ~~a~~Action, i.e., the ~~a~~Action is performed on the ~~target.~~Target (e.g., IP Address).

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] ~~[]~~ ~~[]~~ and [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.3 Normative References

[RFC1123]

Braden, R., Ed., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, DOI 10.17487/RFC1123, October 1989, https://www.rfc-editor.org/info/rfc1123.

[RFC2119]

Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, https://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, https://www.rfc-editor.org/info/rfc8174.

[RFC8259]

Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, .

[RFC1123]

Author, T., "Requirements for Internet Hosts", October 1989. .

[RFC4291]

Hinden, R., Deering S. , T., "IP Version 6 Addressing Architecture ", February 2006, .

[RFC2673]

Crawford, M., "Binary Labels in Domain Name System", August 1999, .

[RFC3339]

Kline, G., "Date and Time on the Internet: Timestamps", July 2002, .

[RFC5237]

https://www.rfc-editor.org/info/rfc8259Arkko, J., Errícsson, S. , "IANA Allocation Guidelines for the Protocol Field", February 2008, .

[OpenC2-Lang-v1.0]

Open Command and Control (OpenC2) Language Specification Version 1.0. Edited by Jason Romano and Duncan Sparrell. Latest version:November 2018, http://docs.oasis-open.org/openc2/oc2ls/v1.0/oc2ls-v1.0.html.

1.4 Non-n-Normative References

[RFC3339]

Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, https://www.rfc-editor.org/info/rfc3339.

[RFC4291]

Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, https://www.rfc-editor.org/info/rfc4291.

[RFC6891]

Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, DOI 10.17487/RFC6891, April 2013, https://www.rfc-editor.org/info/rfc6891..

[RFC5237]

Arkko, J. and S. Bradner, "IANA Allocation Guidelines for the Protocol Field", BCP 37, RFC 5237, DOI 10.17487/RFC5237, February 2008, https://www.rfc-editor.org/info/rfc5237.

[OpenC2-HTTPS-v1.0]

Specification for Transfer of OpenC2 Messages via HTTPS Version 1.0. Edited by David Lemire. ~~Latest version:~~November, 2018, http://docs.oasis-open.org/openc2/open-impl-https/v1.0/open-impl-https-v1.0.html.

[ACD]

Herring, M.J. and Willett, K.D. "Active Cyber Defense: A Vision for Real-Time Cyber Defense," Journal of Information Warfare, vol. 13, Issue 2, p. 80, April 2014.

[IACD]

Willett, Keith D., "Integrated Adaptive Cyberspace Defense: Secure Orchestration", International Command and Control Research and Technology Symposium, June 2015.

1.5 Document Conventions

1.5.1 Naming Conventions

- [RFC2119]~~RFC2119/RFC8174~~/[RFC8174] key words (see Section 1.2~~section 1.2)~~) are in all uppercase.
- All property names and literals are in lowercase, except when referencing canonical names defined in another standard (e.g., literal values from an IANA registry).

- ~~All words in structure component names are capitalized and are separated with a hyphen, e.g., ACTION, TARGET, TARGET-SPECIFIER.~~

- Words in property names are separated with an underscore (_), while words in string enumerations and type names are separated with a hyphen (-).
- The term "hyphen" used here refers to the ASCII hyphen or minus character, which in Unicode is "hyphen-minus", U+002D.

- ~~All type names, property names, object names, and vocabulary terms are between three and 40 characters long.~~

1.5.2 Font Colors and Style

The following color, font and font style conventions are used in this document:

- A fixed width font is used for all type names, property names, and literals.
- Property names are in bold style – ~~**created_a**t~~**'created_at'**.
- All examples in this document are expressed in JSON. They are in fixed width font, with straight quotes, black text and a light shaded background, and 4-space indentation. JSON examples in this document are representations of JSON Objects. They should not be interpreted as string literals. The ordering of object keys is insignificant. Whitespace before or after JSON structural characters in the examples are insignificant [RFC8259]~~[]~~.
- Parts of the example may be omitted for conciseness and clarity. These omitted parts are denoted with ~~the~~ ellipses (...).

Example:

```
{
    "action": "contain",
    "target": {
        "user_account": {
            "user_id": "fjbloggs",
            "account_type": "windows-local"
        }
    }
}
```

1.6 Overview

In general, there are two types of participants involved in the exchange of OpenC2 Messages, as depicted in Figure 1-1:

1. **Producers**: A Producer is an entity that creates Commands to provide instruction to one or more systems to act in accordance with the content of the Command. A Producer may receive and process Responses in conjunction with a Command.
2. **Consumers**: A Consumer is an entity that receives and may act upon a Command. A Consumer may create Responses that provide any information captured or necessary to send back to the Producer.

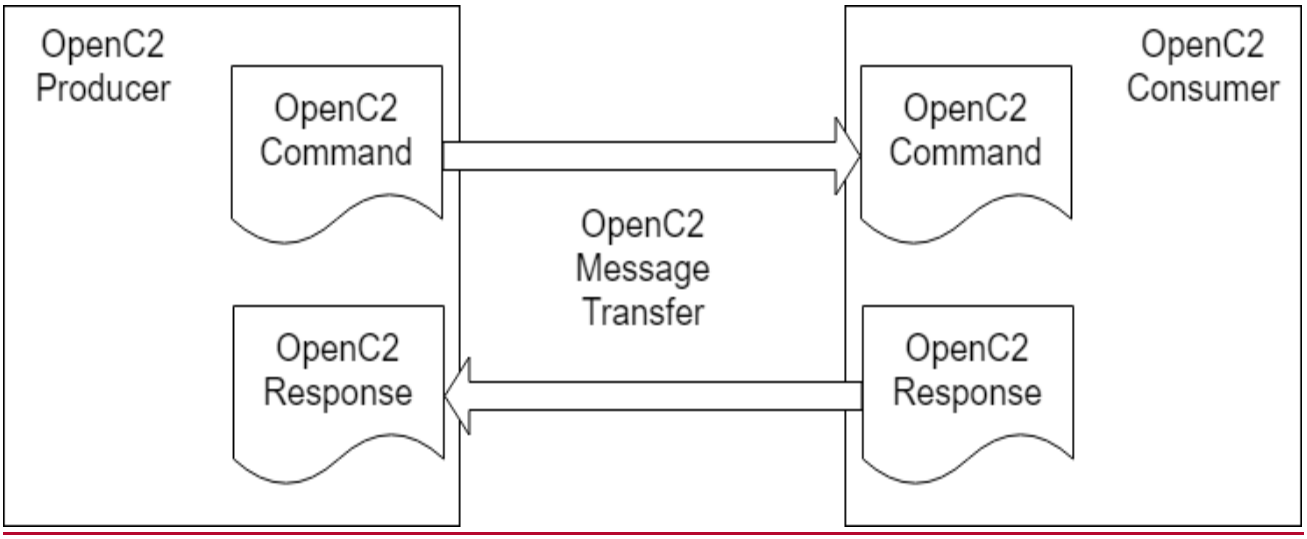


Figure 1-1. OpenC2 Message Exchange

OpenC2 is a suite of specifications for Producers and Consumers to command ~~actuators that~~and execute cyber defense functions. These specifications include the OpenC2 Language Specification, Actuator Profiles, and Transfer Specifications. The OpenC2 Language Specification and Actuator Profile~~(s)~~ specifications focus on the ~~standard~~language content and meaning at the ~~p~~Producer and ~~c~~Consumer of the ~~c~~Command and ~~r~~Response while the transfer specifications focus on the protocols for their exchange.

- The **OpenC2 Language Specification** ([OpenC2-Lang-v1.0]) provides the semantics for the essential elements of the language, the structure for ~~c~~Commands and ~~r~~Responses, and the schema that defines the proper syntax for the language elements that represents the ~~c~~Command or ~~r~~Response.
- **OpenC2 Actuator Profiles** specify the subset of the OpenC2 language relevant in the context of specific ~~a~~Actuator functions. Cyber defense components, devices, systems and/or instances may (in fact are likely~~)~~ to) implement multiple ~~a~~Actuator profiles. Actuator profiles extend the language by defining ~~s~~Specifiers that identify the ~~a~~Actuator to the required level of precision ~~and~~ . Actuator Profiles may define ~~command arguments~~Command Arguments and Targets that are relevant and/or unique to those ~~a~~Actuator functions.
- **OpenC2 Transfer Specifications** utilize existing protocols and standards to implement OpenC2 in specific environments. These standards are used for communications and security functions beyond the scope of the language, such as message transfer encoding, authentication, and end-to-end transport of OpenC2 messages.

The [OpenC2-Lang-v1.0]~~OpenC2 Language Specification~~ defines a language used to compose ~~m~~Messages for command and control of cyber defense systems and components. A ~~m~~Message consists of a header and a payload

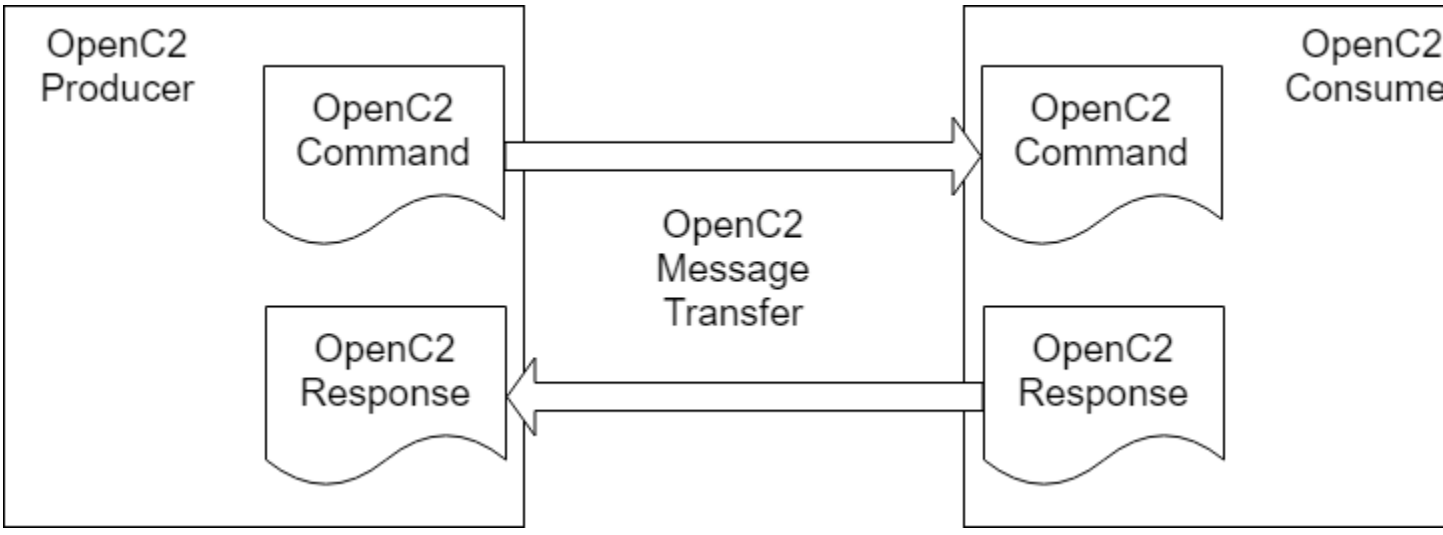
(*defined* as a ~~m~~Message body in the OpenC2 Language Specification Version 1.0 and *specified* in one or more ~~a~~Actuator profiles).

~~In general, there are two types of participants involved in the exchange of OpenC2 messages, as depicted in Figure 1-1:~~

1. ~~**OpenC2 Producers**: An OpenC2 Producer is an entity that creates commands to provide instruction to one or more systems to act in accordance with the content of the command. An OpenC2 Producer may receive and process responses in conjunction with a command.~~
2. ~~**OpenC2 Consumers**: An OpenC2 Consumer is an entity that receives and may act upon an OpenC2 command. An OpenC2 Consumer may create responses that provide any information captured or necessary to send back to the OpenC2 Producer.~~

The language defines two payload structures:

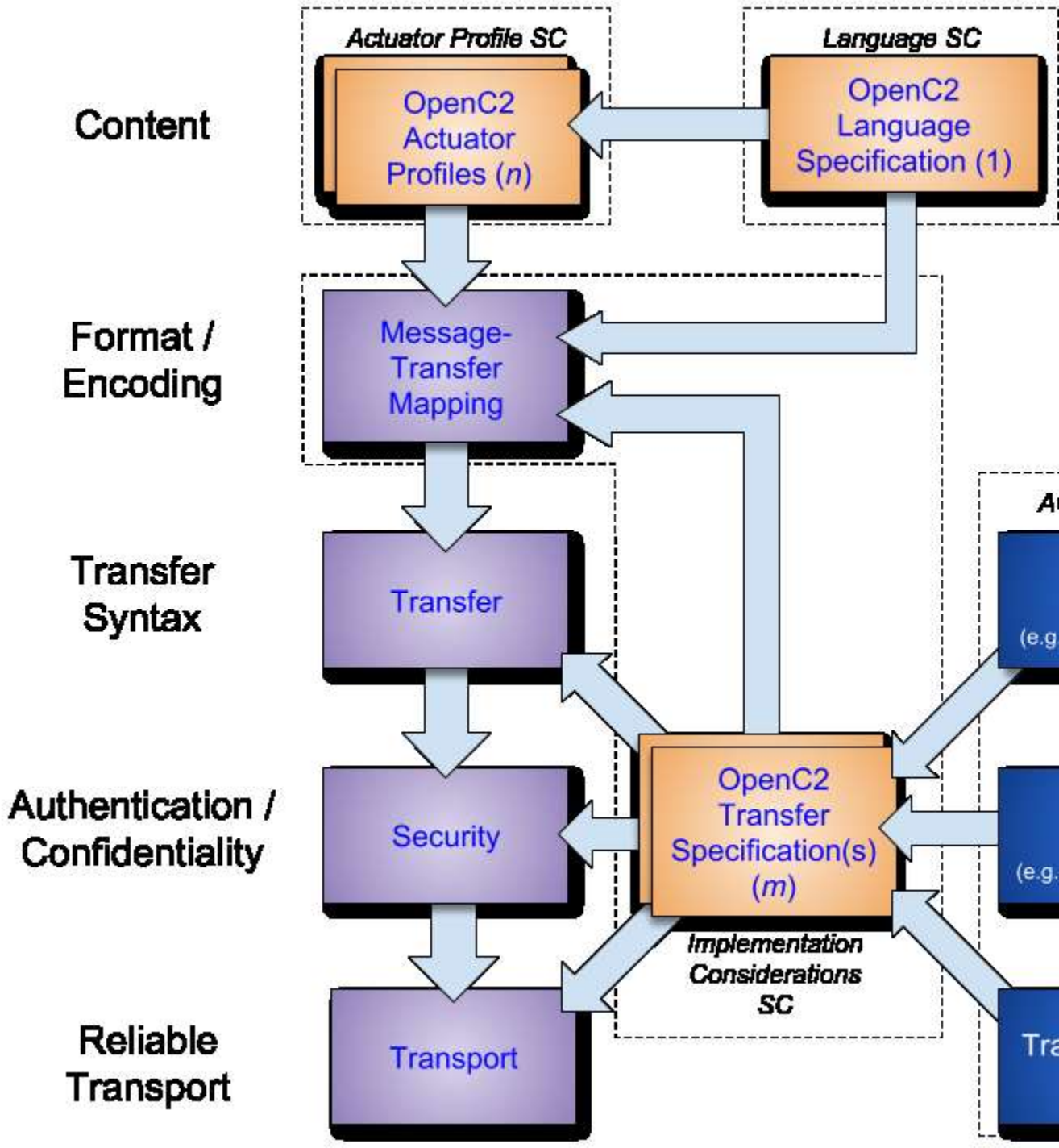
1. **Command**: An instruction from one system known as the ~~OpenC2 "~~Producer~~",~~ to one or more systems, the ~~OpenC2 "~~Consumer(s~~)",~~ to act on the content of the ~~c~~Command.
2. **Response**: Any information ~~captured or necessary to send~~sent back to the ~~OpenC2~~ Producer ~~that issued~~as a result of the Command~~, i.e., the OpenC2 Consumer's response to the OpenC2 Producer~~.



~~Figure 1-1. OpenC2 Message Exchange~~

OpenC2 implementations integrate the related OpenC2 specifications described above with related industry specifications, protocols, and standards. Figure 1-2 depicts the relationships among OpenC2 specifications, and their relationships to other industry standards and environment-specific implementations of OpenC2. Note that the layering of implementation aspects in the diagram is notional, and not intended to preclude~~, e.g.,~~ any particular

approach to implementing the needed functionality (for example, the use of an application-layer message signature function to provide message source authentication and integrity~~.~~).



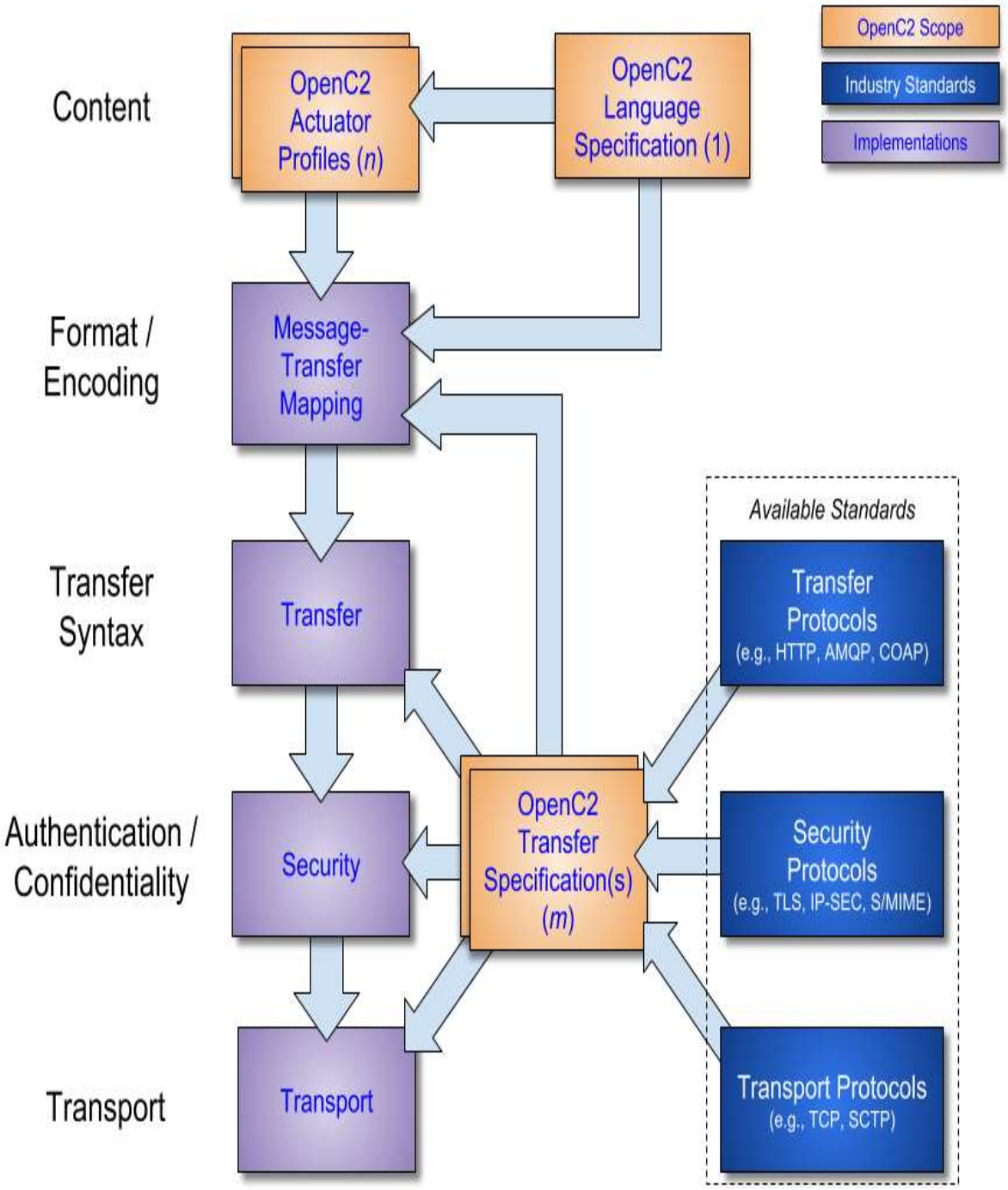


Figure 1-2. OpenC2 Documentation and Layering Model

OpenC2 is conceptually partitioned into four layers as shown in Table 1-1.

Table 1-1. OpenC2 Protocol Layers

Layer	Examples
Function-Specific Content	Actuator Profiles (standard and extensions)
Common Content	[OpenC2-Lang-v1.0]~~Language Specification (this document)~~
Message	Transfer Specifications (OpenC2-over-HTTPS, OpenC2-over-CoAP, …)
Secure Trans~~p~~fer~~t~~	HTTPS, CoAP, MQTT, OpenDXL, ...

- The **Secure Trans~~p~~fer~~t~~** layer provides a communication path between the ~~p~~Producer and the ~~c~~Consumer. OpenC2 can be layered over any standard trans~~p~~fer~~t~~ protocol.
- The **Message** layer provides a trans~~p~~fer~~t~~- and content-independent mechanism for conveying requests, responses, and notifications. A transfer specification maps trans~~p~~fer~~t~~-specific protocol elements to a trans~~p~~fer~~t~~-independent set of message elements consisting of content and associated metadata.
- The **Common Content** layer defines the structure of ~~OpenC2 commands~~Commands and ~~r~~Responses and a set of common language elements used to construct them.
- The **Function-specific Content** layer defines the language elements used to support a particular cyber defense function. An ~~a~~Actuator profile defines the implementation conformance requirements for that function. ~~OpenC2~~ Producers and Consumers will support one or more profiles.

The components of ~~an OpenC2~~a Command are an ~~a~~Action (what is to be done), a ~~t~~Target (what is being acted upon), an optional ~~a~~Actuator (what is performing the ~~c~~Command), and ~~command arguments~~Command Arguments, which influence how the ~~c~~Command is to be performed. An ~~a~~Action coupled with a ~~t~~Target is sufficient to describe a complete ~~OpenC2~~ Command. Though optional, the inclusion of an ~~a~~Actuator and/or ~~command arguments~~Command Arguments provides additional precision to a ~~command, when needed~~Command.

The components of ~~an OpenC2~~a Response are a numerical status code, an optional status text string, and optional results. The format of the results, if included, depend on the type ~~or response~~of Response being transferred.

1.7 Goal

The goal of the OpenC2 Language Specification is to provide a language for interoperating between functional elements of cyber defense systems. This language used in conjunction with OpenC2 Actuator Profiles and OpenC2 Transfer Specifications allows for vendor-agnostic cybertime response to attacks.

The Integrated Adaptive Cyber Defense (IACD) framework defines a collection of activities, based on the traditional OODA (Observe–Orient–Decide–Act) Loop [IACD[IACD]:]:

- Sensing: gathering of data regarding system activities
- Sense Making: evaluating data using analytics to understand what's happening
- Decision Making: determining a course-of-action to respond to system events
- Acting: Executing the course-of-action

The goal of OpenC2 is to enable coordinated defense in cyber-relevant time between decoupled blocks that perform cyber defense functions. OpenC2 focuses on the Acting portion of the IACD framework; the assumption that underlies the design of OpenC2 is that the sensing/analytics have been provisioned and the decision to act has been made. This goal and these assumptions guides the design of OpenC2:

- **Technology Agnostic:** The OpenC2 language defines a set of abstract atomic cyber defense actions in a platform and productimplementation agnostic manner
- **Concise:** An OpenC2 commandA Command is intended to convey only the essential information required to describe the action requiredan Action and can be represented in a very compact form for communications-constrained environments
- **Abstract:** OpenC2 commandsCommands and rResponses are defined abstractly and can be encoded and transferred via multiple schemes as dictated by the needs of different implementation environments
- **Extensible:** While OpenC2 defines a core set of aActions and tTargets for cyber defense, the language is expected to evolve with cyber defense technologies, and permits extensions to accommodate new cyber defense technologies.

1.8 Purpose and Scope

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 ~~-~~ Filtering herein referred to as SLPF.

This ~~a~~Actuator profile specifies the set of ~~actions, targets, specifiers~~Actions, Targets, Specifiers, and ~~command arguments~~Command Arguments that integrates SLPF functionality with the Open Command and Control (OpenC2) ~~c~~Command set. Through this ~~c~~Command set, cyber security orchestrators may gain visibility into and provide control over the SLPF functionality in a manner that is independent of the instance of the SLPF function.

All components, devices and systems that provide SLPF functionality will implement the OpenC2 ~~ACTIONS, TARGETS, SPECIFIERS and ARGS~~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~~Actions for ~~a~~Actuators with SLPF functionality~~.~~
- Identify the required and optional ~~TARGET~~Target types ~~and associated specifiers~~ for each ~~a~~Action in the SLPF class of ~~a~~Actuators~~.~~
- Identify ~~ACTUATOR-SPECIFIERS, ACTUATOR-ARGS and COMMAND-ARGS~~Actuator-Specifiers and Arguments for each ~~action-target~~Action/Target pair that are applicable and/or unique to the SLPF class of ~~a~~Actuators
- Annotate each Action/Target pair with a justification and example, and provide sample OpenC2 ~~c~~Commands to a SLPF with corresponding ~~r~~Responses

- ~~Provide an abstract schema that captures the specifiers and options for a SLPF~~

This SLPF profile:

- Does not define or implement ~~ACTIONS~~Actions beyond those defined in Version 1.0 of the [OpenC2-Lang-v1.0]~~Language Specification.~~
- Is consistent with ~~v~~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 ~~c~~Commands, receive ~~r~~Responses, and manage the execution of ~~c~~Commands involving one or more SLPF or other ~~a~~Actuators with SLPF capability. The OpenC2 ~~p~~Producer needs *a priori* knowledge of which ~~c~~Commands the ~~a~~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 ~~a~~Actuators that execute the cyber defense function, but could be orchestrators (i.e., a device or instance that forwards ~~c~~Commands to the ~~a~~Actuator)~~.~~

Though cyber defense components, devices, systems and/or instances may ~~may~~ implement multiple ~~a~~Actuator profiles, a particular OpenC2 ~~m~~Message may reference at most a single ~~a~~Actuator profile. The scope of this document is limited to SLPF.

This specification is organized into three major sections.

Section One (this section) provides a ~~nonnormative~~non-normative overview of the suite of specifications that realize OpenC2. This section provides references as well as defines the scope and purpose of this specification.

Section Two~~Section Two~~ (normative) binds this particular profile to the OpenC2 Language Specification. Section Two enumerates the components of the language specification that are meaningful in the context of SLPF and defines components that are applicable to this distinct profile. Section Two also defines the ~~c~~Commands (i.e., the ~~action target~~Action/Target pairs) that are permitted in the context of SLPF.

Section Three~~Section Three~~ (normative) presents definitive criteria for conformance so that cyber security stakeholders can be assured that their products, instances and/or integrations are compatible with OpenC2.

Annex A~~This specification provides three~~ (non-normative~~ Annexes. OpenC2 is intended for machine to machine interactions, therefore a schema for SLPF and the applicable portions of the OpenC2 Language schema are provided to facilitate development. There is also an Annex that~~) provides multiple examples of ~~SLPF commands~~Commands and associated Responses (JSON serialization~~).~~) to facilitate development.

---

2 OpenC2 Language Binding

This section is normative

This section defines the set of ~~ACTIONS, TARGETS, SPECIFIERS~~Actions, Targets, Specifiers, and ~~ARGS~~Arguments that are meaningful in the context of an SLPF. This section also describes the appropriate format ~~of~~for the ~~response frame's~~ status and ~~results field.~~properties of a Response frame. This section is organized into three major subsections; Command Components, Response Components and Commands.

Extensions to the Language Specification are defined in accordance with [OpenC2-Lang-v1.0], Section 3.1.5, where:

1. The unique name of the SLPF schema is `oasis-open.org/openc2/v1.0/ap-slpf`
2. The namespace identifier (nsid) referring to the SLPF schema is: `slpf`
~~1.~~3. The definitions of and conformance requirements for these types are contained in this document

2.1 OpenC2 Command Components

The components of an OpenC2 ~~c~~Command include ~~ACTIONS, TARGETS, ACTUATORS~~Actions, Targets, Actuators and associated ~~ARGS~~Arguments and ~~SPECIFIERS~~Specifiers. Appropriate aggregation of the components will define a ~~c~~Command-body that is meaningful in the context of an SLPF.

This specification identifies the applicable components of an OpenC2 ~~c~~Command. The components of an OpenC2 ~~c~~Command include:

- ~~ACTION~~Action: A subset of the ~~ACTIONS~~Actions defined in the OpenC2 Language ~~s~~Specification that are meaningful in the context of a SLPF.
  - This profile ~~does not~~SHALL NOT define ~~ACTIONS~~Actions that are external to Version 1.0 of the OpenC2 Language Specification~~OpenC2 Language Specification.~~
  - This profile MAY augment the definition of the ~~a~~Actions in the context of a SLPF~~.~~
  - This profile SHALL NOT define ~~ACTIONS~~Actions in a manner that is inconsistent with version 1.0 of the OpenC2 ~~language specification.~~Language Specification
- ~~TARGET~~Target: A subset of the ~~TARGETS~~Targets and ~~target-specifiers~~Target-Specifiers defined in Version 1.0 of the OpenC2 Language ~~s~~Specification that are meaningful in the context of SLPF and one ~~TARGET~~Target (and its associated ~~s~~Specifier) that is defined in this specification~~.~~
- ~~ARGS~~Arguments: A subset of the ~~COMMAND-ARGS~~Arguments defined in the Language Specification and a set of ~~ACTUATOR-ARGS~~Arguments defined in this specification~~.~~
- ~~ACTUATOR~~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 ~~a~~Actions defined in version 1.0 of the Language Specification which are meaningful in the context of an SLPF. The particular ~~action/target~~Action/Target pairs that are required or are optional are presented in Section 2.3~~section 2.3.~~.

Table 2.1.1-1. Actions Applicable to SLPF

Type: Action (Enumerated)

ID	Name	Description
3	**query**	Initiate a request for information. Used to communicate the supported options and determine the state or settings~~.~~
6	**deny**	Prevent traffic or access~~.~~
8	**allow**	Permit traffic or access~~.~~
16	**update**	Instructs the ~~a~~Actuator to update its configuration by retrieving and processing a configuration file and update~~.~~
20	**delete**	Remove an access rule~~.~~

2.1.2 Targets

Table 2.1.2-1 summarizes the Targets defined in Version 1.0 of the [OpenC2-Lang-v1.0] as they relate to SLPF functionality. Table 2.1.2-2 summarizes the Targets that are defined in this specification.

2.1.2.1 Common Targets

Table 2.1.2-1 lists the ~~TARGETs~~Targets defined in the OpenC2 Language ~~s~~Specification that are applicable to SLPF. The particular ~~action/target~~Action/Target pairs that are required or are optional are presented in Section 2.3~~section 2.3.~~.

Table 2.1.2-1. Targets Applicable to SLPF

Type: Target (Choice)

ID	Name	Type	Description
9	**features**	Features	A set of items such as Action/Target pairs.

ID	Name	Type	Description
			profiles versions, options that are supported by the Actuator. The Target is used with the query Action to determine an Actuator's capabilities
10	**file**	File	Properties of a file.
11 13	**ip_addr ipv4_net**	IP-Addr IPv4-Net	The representation of one or more IPv4 addresses (either version 4 or version 6) expressed using CIDR notation.
14	**ipv6_net**	IPv6-Net	The representation of one or more IPv6 addresses expressed using CIDR notation
15	**ipv4_connection**	IPv4-Connection	A network connection that originates from as specified by a source and is addressed to a destination. Source and destination addresses may be either five-tuple (IPv4 or IPv6; both should be the same version)
16	**features ipv6_connection**	IPv6-Connection Features	A set of items such as action target pairs, profiles versions, options that are supported by the actuator. The target is used with the query action to determine an actuator's capabilities. A network connection as specified by a five-tuple (IPv6)

1024	**slpf**	slpf:Target	Targets defined in the Stateless Packet Filter profile.

The semantics/ requirements as they pertain to common targets:

- ipv4_connection
  - If the protocol = ICMP, the five-tuple is: src_addr, dst_addr, icmp_type, icmp_code, protocol where the ICMP types and codes are defined in [RFC2780]
  - If the protocol = TCP, UDP or SCTP, the five-tuple is: src_addr, src_port, dst_addr, dst_port, protocol
  - For any other protocol, the five-tuple is: src_addr, unused, dst_addr, unused, protocol
- ipv6_connection
  - If the protocol = ICMP, the five-tuple is: src_addr, dst_addr, icmp_type, icmp_code, protocol where the ICMP types and codes are defined in [RFC4443]
  - If the protocol = TCP, UDP or SCTP, the five-tuple is: src_addr, src_port, dst_addr, dst_port, protocol
  - For any other protocol, the five-tuple is: src_addr, unused, dst_addr, unused, protocol

2.1.2.2 SLPF Targets

The slpf:Target type isThe list of common Targets is extended to include the additional Targets defined in this specification and is referenced underwith the slpf namespace. Implementations that choose to include this type MUST import it in accordance with the procedures defined in section

Table 2.2.6 of Version **1.**0 of the OpenC2 Language Specification:**2-2. Targets Unique to SLPF**

2.1. The unique name of the SLPF schema is `oasis-open.org/openc2/v1.0/ap-slpf`

3.1. The namespace identifier (nsid) referring to the SLPF schema is: `slpf`

1. The list of types imported from the SLPF schema is: `Target`, `Actuator`, `Args`, and `Results`.
2. The definitions of and conformance requirements for these types are contained in this document.

Type: Target (Choice)

ID	Name	Type	Description

~~1~~	~~**rule_number**~~	~~Rule-ID~~	~~Immutable identifier assigned when a rule is created, Identifies a rule to be deleted.~~
1024	**rule_number**	Rule-ID	Immutable identifier assigned when a rule is created. Identifies a rule to be deleted

Implementations that choose to ~~support~~implement the slpf:Target MUST support the **rule_number** ~~t~~Target.

2.1.3 Command Arguments

Arguments provide additional precision to a ~~c~~Command by including information such as how, when, or where a ~~c~~Command is to be executed. Table 2.1.3-1 summarizes the ~~command arguments~~Command Arguments defined in Version 1.0 of the [OpenC2-Lang-v1.0]~~OpenC2 Language Specification~~ as they relate to SLPF functionality. Table 2.1.3-2 summarizes the ~~command arguments~~Command Arguments that are defined in this specification.

2.1.3.1 Common Arguments

Table 2.1.3~~.1~~-1 lists the ~~command arguments~~Command Arguments defined in the [OpenC2-Lang-v1.0]~~OpenC2 Language specification~~ that are applicable to SLPF.

Table 2.1.3-1. Command Arguments applicable to SLPF

Type: Args (Map)

ID	Name	Type	#	Description
1	**start_time**	Date-Time	0..1	The specific date/time to initiate the ~~a~~Action
2	**stop_time**	Date-Time	0..1	The specific date/time to terminate the ~~a~~Action
3	**duration**	Duration	0..1	The length of time for an ~~a~~Action to be in effect

ID	Name	Type	#	Description
4	**response_requested**	Response-Type	0..1	The type of ~~r~~Response required for the ~~a~~Action: `none`, `ack`, `status`, `complete`~~.~~

~~1024~~	~~**slpf**~~	~~slpf:Args~~	~~0..1~~	~~Command arguments defined in the Stateless Packet Filter profile~~

2.1.3.2 SLPF Arguments

The ~~semantics/requirements as they relate to~~ list of common ~~arguments:~~

- ~~start-time/end-time/duration~~
  - ~~If none are specified then~~Command Arguments is extended to include 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~~
- ~~response_requested~~
  - ~~If absent or not explicitly set in an OpenC2 Command, then a Consumer MUST respond the same as response_type `complete`.~~

~~2.1.3.2 SLPF Args~~

~~The command arguments~~additional Command Arguments defined in this ~~document are~~section and referenced ~~under~~with the slpf namespace.

Table 2.1.3-2. Command Arguments Unique to SLPF

Type: Args (Map)

ID	Name	Type	#	Description
~~1~~1024	**drop_process**	Drop-Process	0..1	Specifies how to handle denied packets
~~2~~1025	**running**	Boolean	0..1	Normal operations assume~~s~~ any change~~s~~ to a device are to be implemented ~~as~~

ID	Name	Type	#	Description
				~~persistent changes~~persistently. Setting the running modifier to TRUE results in a change that is not persistent in the event of a reboot or restart~~.~~
~~3~~1026	**direction**	Direction	0..1	Specifies whether to apply rules to incoming or outgoing traffic. If omitted, rules are applied to both~~.~~
41027	**insert_rule**	Rule-ID	0..1	Specifies the identifier of the rule within a list, typically used in a top-down rule list~~.~~

Type: Drop-Process (Enumerated)

ID	Name	Description
1	**none**	Drop the packet and do not send a notification to the source of the packet~~.~~
2	**reject**	Drop the packet and send an ICMP host unreachable (or equivalent) to the source of the packet~~.~~
3	**false_ack**	Drop the traffic and send a false acknowledg~~e~~ment~~.~~

Type: Direction (Enumerated)

ID	Name	Description
1	**ingress**	Apply rules to incoming traffic only
2	**egress**	Apply rules to outgoing traffic only

Type: Rule-ID

Type Name	Type	Description
Rule-ID	Integer	Access rule identifier

The semantics/requirements as they relate to SLPF arguments:

- insert_rule:
  - The value MUST be immutable - i.e. the identifier assigned to an access rule at creation must not change over the lifetime of that rule~~.~~
  - The value MUST be unique within the scope of ~~a command sent to an openc2 consumer~~ an Openc2 Producer and an Openc2 Consumer- i.e. ~~a rule_number maps~~the value MUST map to exactly one deny or allow for a given instance of an SLPF
- directionality:
  - Entities that receive but do not support directionality MUST NOT reply with 200 OK and SHOULD return a 501 error code~~.~~
  - If absent, then the ~~c~~Command MUST apply to both~~.~~
- drop_process: If absent or not explicitly set, then the ~~a~~Actuator MUST NOT send any notification to the source of the packet
- running: If absent or not explicitly set, then the value is FALSE and any changes are persistent~~.~~

2.1.4 Actuator Specifiers

An ~~ACTUATOR~~Actuator is the entity that provides the functionality and performs the ~~a~~Action. The ~~ACTUATOR~~Actuator executes the ~~ACTION~~Action on the ~~TARGET~~Target. In the context of this profile, the ~~a~~Actuator is the SLPF and the presence of one or more ~~s~~Specifiers further refine which ~~a~~Actuator(s) shall execute the ~~a~~Action.

Table 2.1.4-1 lists the ~~s~~Specifiers that are applicable to the SPLF ~~actuator. Annex C~~Actuator. Annex A provides sample ~~c~~Commands with the use of ~~s~~Specifiers.

The ~~actuator specifiers~~Actuator Specifiers defined in this document are referenced under the slpf namespace.

Table 2.1.4-1. SLPF Specifiers

Type: Specifiers (Map)

ID	Name	Type	#	Description
1	**hostname**	String	0..1	[RFC1123]~~RFC 1123~~ hostname (can be a domain name or IP address) for a particular device with SLPF functionality
2	**named_group**	String	0..1	User defined collection of devices with SLPF functionality
3	**asset_id**	String	0..1	Unique identifier for a particular SLPF
4	**asset_tuple**	String	0..10	Unique tuple identifier for a particular SLPF consisting of a list of up to 10 strings

2.2 OpenC2 Response Components

Response messages originate from the ~~ACTUATOR~~Actuator as a result of a ~~c~~Command.

Responses associated with required ~~a~~Actions MUST be implemented. Implementations that include optional ~~ACTIONS~~Actions MUST implement the RESPONSE associated with the implemented ~~ACTION~~Action. Additional details regarding the ~~c~~Command and associated ~~r~~Response are captured in Section 2.3~~section 2.3.~~. Examples ~~will be~~are provided in Annex A~~Annex C.~~.

2.2.1 Common Results

Table 2.2.1-1 lists the ~~results~~Response properties defined in the [OpenC2-Lang-v1.0]~~OpenC2 Language specification~~ that are applicable to SLPF.

Table 2.2.1-1. Results Applicable to SLPF

Type: OpenC2-Response (Map)

ID	Name	Type	#	Description
1	**status**	Status-Code	0..1	An integer status code

ID	Name	Type	#	Description
2	**status_text**	String	0..1	A free-form human-readable description of the ~~r~~Response status
6	**versions**	Version	0..n	List of OpenC2 language versions supported by this ~~a~~Actuator
7	**profiles**	~~jadn:~~Uname	0..n	List of profiles supported by this ~~a~~Actuator
~~8~~	~~**schema**~~	~~jadn:Schema~~	~~0..1~~	~~Syntax of the OpenC2 language elements supported by this actuator~~
~~9~~8	**pairs**	Action-Targets	0..n	List of ~~t~~Targets applicable to each supported ~~a~~Action
~~10~~9	**rate_limit**	Number	0..1	Maximum number of requests per minute supported by design or policy
~~1024~~	~~**slpf**~~	~~slpf:Results~~	~~0..1~~	~~Response data defined in the Stateless Packet Filtering profile~~

Table 2.2.1-2 lists the Status Codes defined in the OpenC2 Language ~~s~~Specification that are applicable to SLPF.

Table 2.2.1-2. Status Codes

Type: Status-Code (Enumerated.ID)

Value	Description
102	Processing. Command received but action not necessarily complete.
200	OK.
400	Bad Request. Unable to process ~~c~~Command, parsing error.

Value	Description
500	Internal Error. For ~~r~~Response type complete, one of the following MAY apply: * Cannot access file or path * Rule number currently in use * Rule not updated
501	Not implemented. For ~~r~~Response type complete, one of the following MAY apply: * Target not supported * Option not supported * Command not supported

2.2.2 SLPF Results

The ~~results~~list of common Response properties is extended to include the additional Response properties defined in this ~~document are presented in Table 2.2-2. The results are~~section and referenced ~~under~~with the slpf namespace ~~within the OpenC2-Response type defined in the OpenC2 language specification~~.

Table 2.2-.2-1. SLPF Results

Type: ~~Results~~ OpenC2-Response (Map)

ID	~~Type~~ Name	Type	Description
1024	**rule_number**	Rule-ID	Rule identifier returned from allow or deny ~~c~~Command~~.~~

2.3 OpenC2 Commands

An OpenC2 ~~c~~Command consists of an ~~ACTION/TARGET~~Action/Target pair and associated ~~SPECIFIERS~~Specifiers and ~~ARGUMENTs~~Arguments. This section enumerates the allowed ~~commands, identify which are required or optional to implement,~~Commands and presents the associated ~~r~~Responses.

Table 2.3-1 defines the ~~commands allowed by~~ Commands that are valid in the context of the SLPF profile ~~and indicates if implementation of the command is required or optional for Openc2 Producers and/or Openc2 Consumers.~~. An ~~ACTION~~Action (the top row in Table 2.3-1) paired with a ~~TARGET~~Target (the first column in Table 2.3-1) defines ~~an allowable~~

command.a valid Command. The subsequent subsections provide the property tables applicable to each OpenC2 cCommand.

Table 2.3-1. Command Matrix

	Allow	Deny	Query	Delete	Upda
ipv4_connection	validrequired	validrequired			
ip_addripv6_connection	requiredvalid	validrequired			
ipv4_net	valid	valid			
ipv6_net	valid	valid			
features			requiredvalid		
slpf:rule_number				optionalvalid	
file					optio

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

Table 2.3-2. Command Arguments Matrix

	Allow *target*	Deny *target*	Query features	Delete slpf:rule_number	Update file
response	2.3.12.3.1	2.3.22.3.2	2.3.3.12.3.3.1	2.3.4.12.3.4.1	2.3.5.12.3.5.1
start-time	2.3.12.3.1	2.3.22.3.2		2.3.4.12.3.4.1	2.3.5.12.3.5.1

	Allow *target*	Deny *target*	Query features	Delete slpf:rule_number	Update file
end-time	2.3.12.3.1	2.3.22.3.2			
duration	2.3.12.3.1	2.3.22.3.2			
running	2.3.12.3.1	2.3.22.3.2			
direction	2.3.12.3.1	2.3.22.3.2			
insert_rule	2.3.12.3.1	2.3.22.3.2			
drop_process		2.3.22.3.2			

2.3.1 ‘Allow’

Table 2.3.1-1 summarizes the command optionsCommand Arguments that apply to all of the cCommands consisting of the ‘allow’ action'allow' Action and a valid tTarget type.

Upon receipt of an unsupported command argumentCommand Argument, SLPF cConsumers

- MUST NOT respond with a OK/200.
- SHOULD respond with the 501 status code.
- SHOULD respond with "Option not supported" in the status text.
- MAY respond with the 500 status code.

ProductsOpenC2 Producers that send ‘allow target’ commands'allow target' Commands and support the ‘'delete slpf:rule_number’ commandnumber' Command:

- MUST support the slpf:rule_number tTarget type as defined in Section 2.1.2.2section 2.1.2.2
- SHOULD populate the command optionsCommand Arguments field with "response_requested" : "complete”"
- MAY populate the command argumentsCommand Arguments field with the "insert_rule" : option.
- MUST populate the command optionsCommand Arguments field with "response_requested" : “"complete" if the insert_rule aArgument is populated.

~~Products~~OpenC2 Consumers that receive and successfully parse ~~‘allow ’ commands~~'allow ' Commands but cannot implement the ~~‘allow ’~~'allow ' :

- MUST NOT respond with a OK/200~~.~~
- SHOULD respond with the 501 status code~~.~~
- SHOULD respond with ~~‘~~'Rule not updated~~’~~' in the status text~~.~~
- MAY respond with the 500 status code~~.~~

~~Products~~OpenC2 Consumers that receive ~~‘allow ’ commands~~'allow ' Commands and support the ~~‘~~'delete slpf:rule_~~number’ command~~number' Command:

- MUST support the slpf:rule_number ~~t~~Target type as defined in Section 2.1.2.2~~section 2.1.2.2~~
- Upon successful implementation of the ~~‘allow ’~~'allow ', MUST return the rule_number associated with the rule if the "response_requested" : ~~“~~"complete" option is populated.

~~Products~~OpenC2 Consumers that receive ~~‘allow target’ commands~~'allow target' Commands and support the ~~‘insert_rule’ command argument~~'insert_rule' Command Argument:

- MUST assign the rule number provided if the "insert_rule" : option is populated.
- If the rule number is currently in use, then
  - MUST NOT respond with a OK/200.
  - SHOULD respond with the 501 status code.
  - SHOULD respond with ~~‘~~'Rule number currently in use~~’~~' in the status text.
  - MAY respond with the 500 status code.

~~- MAY respond with the 500 status code.~~

The valid ~~t~~Target types, associated ~~s~~Specifiers, and ~~o~~Options are summarized in Section 2.3.1.1~~sections 2.3.1.1~~ and Section 2.3.1.2~~2.3.1.2.~~. Sample ~~c~~Commands are presented in Annex A~~Annex C.~~.

2.3.1.1 ~~‘Allow ip_connection’~~'Allow ipv4_connection'

The ~~‘allow ip_connection’ command is required~~'allow ipv4_connection' Command is OPTIONAL for ~~openc2 producers~~Openc2 Producers implementing the SLPF.

~~If~~ The 'allow ipv4_connection' Command is OPTIONAL for Openc2 Consumers implementing the ~~‘allow ip_addr’ target is not implemented, then~~ SLPF ~~consumers MUST implement the ‘allow ip-connection’ command. Otherwise it is OPTIONAL~~.

The ~~c~~Command permits traffic that is consistent with the specified ipv4_connection. A valid ~~'allow ip_connection' command~~'allow ipv4_connection' Command has at least one property of the ipv4_connection populated and may have any combination of the five properties populated. An unpopulated property within the ipv4_connection Target MUST be treated as an 'any'.

Products that receive but do not implement the 'allow ipv4_connection' Command:

- MUST NOT respond with a OK/200
- SHOULD respond with the 501 Response code
- SHOULD respond with 'Target type not supported' in the status text
- MAY respond with the 500 status code

2.3.1.2 'Allow ipv6_connection'

The 'allow ipv6_connection' Command is OPTIONAL for Openc2 Producers implementing the SLPF. The 'allow ipv6_connection' Command is OPTIONAL for Openc2 Consumers implementing the SLPF.

The Command permits traffic that is consistent with the specified ipv6_connection. A valid 'allow ipv6_connection' Command has at least one property of the ipv6_connection populated and may have any combination of the five properties populated. An unpopulated property within the the ipv4_connection ~~t~~Target MUST be treated as an ~~'~~'any'~~'~~.

Products that receive but do not implement the ~~'allow ip_connection' command~~'allow ipv6_connection' Command:

- MUST NOT respond with a OK/200~~.~~
- SHOULD respond with the 501 ~~r~~Response code~~.~~
- SHOULD respond with ~~'~~'Target type not supported'~~'~~ in the status text~~.~~

~~o MAY respond with the 500 status code.~~

- MAY respond with the 500 status code

2.3.1.2 ~~'Allow ip_addr'~~3 'Allow ipv4_net'

The ~~'allow ip_addr' command is required~~'allow ipv4_net' Command is OPTIONAL for ~~openc2 producers~~Openc2 Producers implementing the SLPF.

~~If the 'allow ip_connection' target~~ The 'allow ipv4_net' Command is ~~not implemented, then~~ OPTIONAL for Openc2 Consumers implementing the SLPF ~~consumers MUST implement the 'allow ip_addr' command. Otherwise the 'allow ip-addr' command is OPTIONAL~~.

The ~~c~~Command permits traffic as specified by the ~~ip_addr property and may be an IPV4 or IPV6~~range of IPv4 addresses as expressed by CIDR notation. If the mask is absent (or unspecified) then it MUST be treated as a single IPv4 address~~. The ip-addr supports CIDR notation.~~ (i.e. an address range of one element). The address range specified in the ~~ip_addr~~ipv4_net MUST be treated as a source OR destination address.

Products that receive but do not implement the ~~'allow ip_addr' command~~'allow ipv4_net' Command:

- MUST NOT respond with a OK/200~~.~~
- SHOULD respond with the 501 ~~r~~Response code
- SHOULD respond with ~~'~~'Target type not supported~~'~~' in the status text~~.~~
- MAY respond with the 500 status code~~.~~

2.3.1.4 'Allow ipv6_net'

The 'allow ipv6_net' Command is OPTIONAL for Openc2 Producers implementing the SLPF. The 'allow ipv6_net' Command is OPTIONAL for Openc2 Consumers implementing the SLPF.

The Command permits traffic as specified by the range of IPv6 addresses as expressed by CIDR notation. If the mask is absent (or unspecified) then it MUST be treated as a single IPv6 address (i.e. an address range of one element). The address range specified in the ipv6_net MUST be treated as a source OR destination address.

Products that receive but do not implement the 'allow ipv6_net' Command:

- MUST NOT respond with a OK/200
- SHOULD respond with the 501 Response code
- SHOULD respond with 'Target type not supported' in the status text
- MAY respond with the 500 status code

2.3.2 ~~'~~Deny~~'~~

~~'~~Deny~~'~~ can be treated as the mathematical complement to ~~'~~allow~~'~~. With the exception of the additional ~~'drop_process' actuator-argument, the targets, specifiers, options~~'drop_process' Actuator-Argument, the Targets, Specifiers, Options and corresponding ~~r~~Responses are identical to the ~~two 'allow' commands.~~four 'allow' Commands. Table 2.3-2 summarizes the ~~command arguments~~Command Arguments that apply to all of the ~~c~~Commands consisting of the ~~'deny' action~~'deny' Action and valid ~~t~~Target type.

Upon receipt of a ~~c~~Command with an ~~ARGUMENT~~Argument that is not supported by the ~~actuator, actuators~~Actuator:

- ~~SHOULD~~MUST NOT respond with ~~the 501 status code~~OK/200

- SHOULD respond with the 501 status code
- SHOULD respond with ‘'Option not supported'’ in the status text.

- MAY respond with the 500 status code.

- ProductsMAY respond with the 500 status code

OpenC2 Producers that send ‘deny target’ commands'deny target' Commands and support the ‘'delete slpf:rule_number’ commandnumber' Command:

- MUST support the slpf:rule_number tTarget type as defined in Section 2.1.2.2section 2.1.2.1.
- SHOULD populate the command optionsCommand Arguments field with “'"response_requested" : "complete"'”
- MAY populate the command argumentsCommand Arguments field with the "insert_rule" : option.
- MUST populate the command optionsCommand Arguments field with "response_requested" : "complete" if the insert_rule aArgument is populated.

ProductsOpenC2 Consumers that receive ‘deny ’ commands'deny ' Commands and support the ‘'delete slpf:rule_number’ commandnumber' Command:

- MUST support the slpf:rule_number tTarget type as defined in Section 2.1.2.2section 2.1.2.1.
- MUST return the rule number assigned in the slpf object if the "response_requested" : “'"complete"'” aArgument is populated.

ProductsOpenC2 Consumers that receive ‘deny target’ commands'deny target' Commands and support the ‘insert_rule’ command argument'insert_rule' Command Argument:

- MUST assign the rule number provided if the "insert_rule" : aArgument is populated.
- If the rule number is currently in use, then
  - MUST NOT respond with a OK/200.
  - SHOULD respond with the 501 status code
  - SHOULD respond with ‘'Rule number currently in use'’ in the status text.
  - MAY respond with the 500 status code

- MAY respond with the 500 status code.

2.3.3 ‘Query’

The valid ttTarget type, associated ssSpecifiers, and ooOptions are summarized in Section 2.3.3.1section 2.3.3.1.. Sample ccCommands are presented in Annex AAnnex C..

2.3.3.1 ‘Query features’

The ‘query openc2’ command'query features' Command MUST be implemented in accordance with Version 1.0 of the [OpenC2-Lang-v1.0]OpenC2 language specification..

2.3.4 ‘Delete’

The slpf:rule_number is the only valid ttTarget type for the delete aaAction. The associated ssSpecifiers, and ooOptions are summarized in Section 2.3.4.1section 2.3.4.1.. Sample ccCommands are presented in Annex AAnnex C..

2.3.4.1 ‘delete slpf:rule_number’

The ‘'delete slpf:rule_number’ commandnumber' Command is used to remove a firewall rule rather than issue an allow or deny to counteract the effect of an existing rule. Implementation of the ‘'delete slpf:rule_number’ commandnumber' Command is OPTIONAL. Products that choose to implement the ‘'delete slpf:rule_number’ commandnumber' Command MUST implement the slpf:rule_number ttTarget type described in Section 2.1.2.2section 2.1.2.1..

ProductsOpenC2 Producers that send the ‘'delete slpf:rule_number’ commandnumber' Command:

- MAY populate the command argumentsCommand Arguments field with ‘'response_requested" : "complete”.."
- MUST NOT include other command arguments.Command Arguments
- MUST include exactly one rule_number.

ProductsOpenC2 Consumers that receive the ‘'delete slpf:rule_number’ commandnumber' Command:

- but cannot parse or process the ‘'delete slpf:rule_number’ command:number' Command:
    - MUST NOT respond with a OK/200.
    - SHOULD respond with status code 400.
    - MAY respond with the 500 status code
    - MAY respond with the 500 status code
- but do not support the slpf:rule_number ttTarget type:
    - MUST NOT respond with a OK/200.

- ~~SHOULD respond with the 501 status code~~

    - SHOULD respond with the 501 status code
    - SHOULD respond with ~~'~~'target not supported~~'~~' in the status text~~.~~
    - MAY respond with the 500 status code

    - ~~MAY respond with the 500 status code~~

- MUST respond with ~~r~~Response code 200 upon successful parsing of the ~~'~~'delete slpf:rule_~~number' command~~number' Command and subsequent removal of the corresponding rule~~.~~
- upon successful parsing but failure to remove the corresponding rule:
    - MUST NOT respond with OK/200
    - MUST respond with ~~r~~Response code 500
    - SHOULD respond with ~~'~~'firewall rule not removed or updated~~'~~' in the status text~~.~~

Refer to Annex A~~Annex C~~ for sample ~~c~~Commands.

2.3.5 Update

The ~~'file' target~~'file' Target as defined in Version 1.0 of the Language Specification is the only valid ~~t~~Target type for the update ~~a~~Action. The associated ~~s~~Specifiers, and ~~o~~Options are summarized in Section 2.3.5.1~~section 2.3.5.1.~~. Sample ~~c~~Commands are presented in Annex A~~Annex C~~.

2.3.5.1 Update file

The ~~'update file' command~~'update file' Command is used to replace or update files such as configuration files, rule sets, etc. Implementation of the update file ~~c~~Command is OPTIONAL. OpenC2 ~~c~~Consumers that choose to implement the ~~'update file' command~~'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.~~'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~~'update file' Command. OpenC2 Producers and Consumers that choose to implement the ~~'update file' command~~'update file' Command MUST NOT include options other than the options identified in ~~t~~Table 2.3-2.

~~Products~~OpenC2 Producers that send the ~~'update file' command~~'update file' Command:

- MAY populate the arguments field with the "response_requested" argument. ~~“~~"Complete~~”, “~~", "Ack~~”~~" and ~~“~~"None~~”~~" are valid Response-type for ~~‘update file’~~'update file'
- MUST NOT include other ~~command arguments.~~Command Arguments
- MUST populate the name ~~s~~Specifier in the ~~t~~Target~~.~~
- SHOULD populate the path ~~s~~Specifier in the ~~t~~Target~~.~~

~~Products~~OpenC2 Consumers that receive the ~~‘update file’ command~~'update file' Command:

- but cannot parse or process the ~~c~~Command
  - MUST NOT respond with a OK/200~~.~~
  - SHOULD respond with status code 400~~.~~
  - MAY respond with the 500 status code

  - ~~MAY respond with the 500 status code~~
- but do not support the ~~‘update file’ command type~~'update file' Command
  - MUST NOT respond with a OK/200~~.~~
  - SHOULD respond with status code 501
  - SHOULD respond with ~~‘c~~'Command not supported~~’~~' in the status text~~.~~
  - MAY respond with status code 500
- but cannot access the file specified in the file ~~t~~Target
  - MUST respond with status code 500
  - SHOULD respond with ~~‘~~'cannot access file~~’~~' in the status text~~.~~
- upon successful parsing and initiating the processing of the ~~‘update file’ command, products~~'update file' Command, OpenC2 Consumers MAY respond with ~~r~~Response code 102~~.~~
- upon completion of all the steps necessary to complete the update and the ~~a~~Actuator commences operations functioning with the new file, ~~actuators products~~OpenC2 Consumers SHOULD respond with ~~r~~Response code 200~~.~~

Refer to Annex A~~Annex C~~ for sample ~~c~~Commands.

---

3 Conformance statements

This section is normative This section identifies the requirements for twenty-two conformance profiles as they pertain to two conformance targets. The two conformance targets are OpenC2 Producers and OpenC2 Consumers (as defined in Section 1.8~~Definitions: The following terms apply to this section:~~

- ~~**OpenC2 SLPF Producers:** Entities that send commands to and receive responses from OpenC2 SLPF consumers.~~

~~**Basic SLPF**~~ of this specification).

3.1 Clauses Pertaining to the OpenC2 Producer Conformance Target

All OpenC2 Producers~~: OpenC2 SLPF producers~~ that are conformant to ~~all of the normative requirements identified in~~ this specification ~~as REQUIRED to implement~~MUST satisfy Conformance Clause 1 and MAY satisfy one or more of Conformance Clauses 2 through 11.

~~Complete SLPF Producers:~~**3.1.1 Conformance Clause 1: Baseline OpenC2 ~~SLPF producers~~Producer**

An OpenC2 Producer satisfies Baseline OpenC2 Producer conformance if:

- 3.1.1.1 **MUST** support JSON serialization of OpenC2 Commands that are syntactically valid in accordance with the property tables presented in Section 2.1
- 3.1.1.2 All serializations **MUST** be implemented in a manner such that the serialization validates against and provides a one-to-one mapping to the property tables in Section 2.1 of this specification
- 3.1.1.3 **MUST** support the use of a Transfer Specification that is capable of delivering authenticated, ordered, lossless and uniquely identified OpenC2 messages
- 3.1.1.4 **SHOULD** 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 as referenced in Section 1 of this specification
- 3.1.1.5 **MUST** be conformant ~~to all of~~ with Version 1.0 of the OpenC2 Language Specification
- 3.1.1.6 **MUST** implement the 'query features' Command in accordance with the normative ~~requirements identified in this specification~~text provided in Version 1.0 of the OpenC2 Language Specification
- 3.1.1.7 **MUST** implement the 'response_requested' Command Argument as a valid option for any Command
- 3.1.1.8 **MUST** conform to at least one of the following conformance clauses in this specification:
  - Conformance Clause 2
  - Conformance Clause 3
  - Conformance Clause 4
  - Conformance Clause 5

3.1.2 Conformance Clause 2: IP Version 4 Connection Producer

An OpenC2 Producer satisfies 'IP Version 4 Connection Producer' conformance if:

- 3.1.2.1 **MUST** meet all of conformance criteria identified in Conformance Clause 1 of this specification
- 3.1.2.2 **MUST** implement the 'allow ipv4_connection' Command in accordance with Section 2.3.1 of this specification
- 3.1.2.3 **MUST** implement the 'deny ipv4_connection' Command in accordance with Section 2.3.2 of this specification

3.1.3 Conformance Clause 3: IP Version 6 Connection Producer

An OpenC2 Producer satisfies 'IP Version 6 Connection Producer' conformance if:

- 3.1.3.1 **MUST** meet all of conformance criteria identified in Conformance Clause 1 of this specification
- 3.1.3.2 **MUST** implement the 'allow ipv6_connection' Command in accordance with Section 2.3.1 of this specification
- 3.1.3.3 **MUST** implement the 'deny ipv6_connection' Command in accordance with Section 2.3.2 of this specification

3.1.4 Conformance Clause 4: IP Version 4 Net Producer

An OpenC2 Producer satisfies 'IP Version 4 Net Producer' conformance if:

- 3.1.4.1 **MUST** meet all of conformance criteria identified in Conformance Clause 1 of this specification
- 3.1.4.2 **MUST** implement the 'allow ipv4_net' Command in accordance with Section 2.3.1 of this specification
- 3.1.4.3 **MUST** implement the 'deny ipv4_net' Command in accordance with Section 2.3.2 of this specification

3.1.5 Conformance Clause 5: IP Version 6 Net Producer

An OpenC2 Producer satisfies 'IP Version 6 Net Producer' conformance if:

- 3.1.5.1 **MUST** meet all of conformance criteria identified in Conformance Clause 1 of this specification
- 3.1.5.2 **MUST** implement the 'allow ipv6_net' Command in accordance with Section 2.3.1 of this specification
- 3.1.5.3 **MUST** implement the 'deny ipv6_net' Command in accordance with Section 2.3.2 of this specification

3.1.6 Conformance Clause 6: Update File Producer

An OpenC2 Producer satisfies 'Update File Producer' conformance if:

- 3.1.6.1 **MUST** meet all of the conformance criteria identified in Conformance Clause 1 of this specification
- 3.1.6.2 **MUST** implement the 'update file' Command in accordance with Section 2.3.5.1 of this specification

3.1.7 Conformance Clause 7: delete rule number Producer

An OpenC2 Producer satisfies 'delete rule Producer' conformance if:

- 3.1.7.1 **MUST** meet all of the conformance criteria identified in Conformance Clause 1 of this specification
- 3.1.7.2 **MUST** implement the 'delete slpf:rule_number' in accordance with Section 2.3.4.1 of this specification

3.1.8 Conformance Clause 8: Running Producer

An OpenC2 Producer satisfies 'Running Producer' conformance if:

- 3.1.8.1 **MUST** meet all of the conformance criteria identified in Conformance Clause 1 of this specification
- 3.1.8.2 **MUST** implement the 'running' Command Argument as a valid option for any Command associated with the 'deny' or 'allow' Actions in accordance with Section 2.3.1 and Section 2.3.2 of this specification

3.1.9 Conformance Clause 9: Direction Producer

An OpenC2 Producer satisfies 'Direction Producer' conformance if:

- 3.1.9.1 **MUST** meet all of the conformance criteria identified in Conformance Clause 1 of this specification
- 3.1.9.2 **MUST** implement the 'direction' Command Argument as a valid option for any Command associated with the 'deny' or 'allow' Actions in accordance with Section 2.3.1 and Section 2.3.2 of this specification

3.1.10 Conformance Clause 10: drop-process Producer

An OpenC2 Producer satisfies 'drop-process Producer' conformance if:

- 3.1.10.1 **MUST** meet all of the conformance criteria identified in Conformance Clause 1 of this specification

3.1.10.2 **MUST** implement the 'drop_process' Command Argument as a valid option for any Command associated with the 'deny' Actions in accordance with Section

2.3.1 and Section 2.3.2**OpenC2 SLPF Consumers:** Entities that receive commands from and send responses to OpenC2 SLPF Producers.

- **Basic SLPF Consumers:** OpenC2 SLPF consumers of this specification

3.1.11 Conformance Clause 11: Temporal Producer

An OpenC2 Producer satisfies 'Temporal Producer' conformance if:

- 3.1.11.1 **MUST** meet all of the conformance criteria identified in Conformance Clause 1 of this specification
- 3.1.11.2 **MUST** implement the 'start_time' Command Argument as a valid option for any Command other than 'query features'
- 3.1.11.3 **MUST** implement the 'stop_time' and 'duration' Command Arguments as a valid option for any Command other than 'query features' or 'update file'.

3.2 Clauses Pertaining to the OpenC2 Consumer Conformance Target

All OpenC2 Consumers that are conformant to all of the normative requirements identified in this specification as REQUIRED to implementMUST satisfy Conformance Clause 12 and MAY satisfy one or more of Conformance Clauses 13 through 22.

- **Complete SLPF Consumers:** OpenC2 SLPF consumers that are conformant to all of the normative requirements identified in this specification

A conformant OpenC2 implementation SHALL meet all the normative requirements specified in the SLPF Profile as well as applicable normative requirements specified in the Language Specification. Table 3-1 provides a overview of the applicable normative requirements. The traceability for conformance criteria involving commands (action target pairs) are ‘derived’, where derived is defined as a combination of more than a single normative statements from the language specification into a single criteria within the SLPF specification. Sections 3.1 through 3.X provide a concise summary of the corresponding conformance criteria.

Table 3-1: SLPF Traceability Matrix

Conformance Criteria	SLPF Section Reference	Language Specification (V 1.0) Reference	Conformance Criteria Reference
JSON Serialization		2.2	3.1-1.1 and 3.2-1.1
OpenC2 Transfer Specification	1.1	5	3.1-1.3, 3.2-1.3, 3.3-1.2 and 3.4-1.2
Actions	2.1.1	3.3.1.2	
Targets	2.1.1.2	3.4.1.3, 3.4.1.8, 3.4.1.9, 3.4.1.11, 3.4.1.12,	
Slpf:rule_number Target	2.1.1.2.1	SLPF-specific	
'Query features' command	2.3.3.1	4	3.1-2.1.5 and 3.2-2.1.3
'Allow ip_connection	ip_addr'	2.3.1	Derived
Deny ip_connection	ip_addr'	2.3.2	Derived
'Delete slpf:rule_number'	2.3.4.1	SLPF-specific	3.3-2.1.1 and 3.4-2.1.1
'Update file'	2.3.5.1	Derived	3.3-2.1.2 and 3.4-2.2
Command Argument: Response_requested	2.1.3	3.3.1.5	3.1-3.1, 3.2-3.1, 3.2-3.2.1 and 3.2-3.2.2

~~Conformance Criteria~~	~~SLPF Section Reference~~	~~Language Specification (V 1.0) Reference~~	~~Conformance Criteria Reference~~
~~Command Argument: start_time, end_time and/or duration.~~	~~2.1.3~~	~~3.3.1.5~~	~~3.3-3.3.1, 3.3-3.2.1, 3.3-3.2.2 3.4-3.3.1, 3.4-3.2.1, 3.4-3.2.2~~
~~Command Argument: running, direction and/or drop_process~~	~~2.1.3~~	~~SLPF-specific~~	~~3.3-3.3.1, 3.3-3.3.2, 3.3-3.4 3.4-3.3.1, 3.4-3.3.2, 3.4-3.4~~
~~Response Codes~~	~~2.2.1~~	~~3.3.2.2~~	

~~3.1 Conformance Clause 1: Basic SLPF Producers~~

~~The Actuator Profile for the basic Stateless Packet Filtering Producers specifies the minimum functionality required in order for an OpenC2 SLPF Producer implementation to be conformant.~~

~~1. General Conformance:~~

3.2.1 Conformance Clause 12: Baseline OpenC2 Consumer

An OpenC2 Consumer satisfies Baseline OpenC2 Consumer conformance if:

- 3.2.1.1 **MUST** support JSON serialization of OpenC2 ~~c~~Commands that are syntactically valid in accordance with the property tables presented in Section 2.1~~Section 2.1.~~
- 3.2.1.2 All serializations **MUST** be implemented in a manner such that the serialization validates against and provides a one-to-one mapping to the property tables in Section 2.1~~section 2.1~~ of this specification~~.~~
- 3.2.1.3 **MUST** support the use of a Transfer Specification that is capable of delivering authenticated, ordered, lossless and uniquely identified OpenC2 messages~~.~~

~~1. **MUST** be conformant with Version 1.0 (or higher) of the Language Specification~~

~~2. Base Commands (ACTION and TARGET pairs):~~

1. **MUST** implement the following action target pairs where the actions and targets are defined in version 1.0 of the Language Specification.
    1. ‘allow ip_connection’ in accordance with the normative text provided in section 2.3.1 of this specification
    2. ‘allow ip_addr’ in accordance with the normative text provided in section 2.3.1 of this specification
    3. ‘deny ip_connection’ in accordance with the normative text provided in section 2.3.2 of this specification
    4. ‘deny ip_addr’ in accordance with the normative text provided in section 2.3.2 of this specification
    5. ‘query openc2’ in accordance with the normative text provided in version 1.0 of the OpenC2 Language Specification.
3. Command Arguments:
    1. **MUST** implement the ‘response_requested’ command argument as a valid option for any command:

3.2 Conformance Clause 2: Basic SLPF Consumers

The Actuator Profile for Stateless Packet Filtering Consumers specifies the minimum functionality required in order for a basic SPLF Consumer implementation to be conformant.

1. General Conformance:
    1. **MUST** support JSON serialization of OpenC2 commands that are syntactically valid in accordance with the property tables presented in Section 2.1.
    2. All serializations **MUST** be implemented in a manner such that the serialization validates against and provides a one-to-one mapping to the property tables in section 2.1 of this specification.
    3. **MUST** support the use of a transfer specification that is capable of delivering authenticated, ordered, lossless and uniquely identified OpenC2 messages.
    4. **MUST** be conformant with Version 1.0 (or higher) of the Language Specification
2. Base Commands (ACTION and TARGET pairs):
    1. **MUST** implement the following action target pairs where the actions and targets are defined in version 1.0 of the Language specification.
        1. ‘allow ip_connection’ or ‘allow ip_addr’ in accordance with the normative text provided in section 2.3.1 of this specification
        2. ‘deny ip_connection’ or ‘deny ip_addr’ in accordance with the normative text provided in section 2.3.2 of this specification
        3. ‘query openc2’ in accordance with the normative text provided in version 1.0 of the OpenC2 Language Specification.
3. Command Arguments:
    1. **MUST** implement the ‘response_requested’ command argument as a 3.2.1.4 **SHOULD**valid option for any command:
    2. Processing response_requested command arguments

1. 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.
2. All commands received without the response argument (or response argument not set) **MUST** process the command and respond in a manner that is consistent with "response_requested" : "complete".

3.3 Conformance Clause 3: Complete SLPF Producers

OpenC2 SLPF producers that are conformant to all of the normative requirements identified in this specification.

1. General Conformance:
    1. **MUST** meet all of conformance criteria identified in Conformance Clause 1 of this specification
    2. **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 as referenced in section 1 of this specification
2. Commands (ACTION and TARGET pairs): 3. **MUST** implement the following action target pairs where: Version 1.0 of the Language Specification defines the actions, Version 1.0 of the Language Specification defines the ‘file’ target; and the ‘slpf:rule_number’ target type is defined in this specification 1. ‘delete slpf:rule_number’ in accordance with the normative text provided in section 2.3.4.1 of this specification 2. ‘update file’ in accordance with the normative text provided in section 2.3.5.1 of this specification
3. Command Arguments:
    1. **MUST** implement the start_time command argument as a valid option for any command other than ‘query ’
    2. **MUST** implement the following command arguments as a valid option for any command other than ‘query ’ and ‘update file’
        1. end_time
        2. duration
    3. **MUST** implement the following command arguments as a valid option for ‘allow ’ and/or ‘deny ’ commands
        1. running
        2. direction
    4. **MUST** implement the drop_process command argument as a valid option for the ‘deny ’ command

3.4 Conformance Clause 4: Complete SLPF Consumers

~~OpenC2 SLPF producers that are conformant to all of the normative requirements identified in this specification.~~

1. ~~General Conformance:~~
    1. ~~**MUST** meet all of conformance criteria identified in Conformance Clause 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 as referenced in Section 1~~section 1~~ of this specification

2. ~~Commands (ACTION and TARGET pairs):~~
    1. ~~**MUST** implement the following action target pairs where version 1.0 of the Language specification defines the 'file' target and actions; and the 'slpf:rule_number' target type is defined in this specification~~

    - ~~'delete slpf:rule_number'~~ 3.2.1.5 **MUST** be conformant with Version 1.0 of the OpenC2 Language Specification
    - 3.2.1.6 **MUST** implement the 'query features' Command in accordance with the normative text provided in ~~section 2.3.~~version 1.0 of the OpenC2 Language Specification
    - 3.2.1.7 **MUST** implement the 'response_requested' Command Argument as a valid option for any Command
        - 3.2.1.7.1 All Commands received with a Response argument set to 'none' **MUST** process the Command and **MUST NOT** send a Response. This criteria supersedes all other normative text as it pertains to Responses
        - 3.2.1.7.2 All Commands received without the Response argument (or Response argument not set) **MUST** process the Command and Response in a manner that is consistent with "response_requested":"complete"
    - 3.2.1.8 **MUST** conform to at least one of the following conformance clauses in this specification:
        - Conformance Clause 13
        - Conformance Clause 14
        - Conformance Clause 15
        - Conformance Clause 16

3.2.2 Conformance Clause 13: IP Version 4~~.1~~ Connection Consumer

An OpenC2 Consumer satisfies 'IP Version 4 Connection Consumer' conformance if:

- 3.2.2.1 **MUST** meet all of conformance criteria identified in Conformance Clause 12 of this specification

- ~~'update file'~~3.2.2.2 **MUST** implement the 'allow ipv4_connection' Command in accordance with Section 2.3.1~~the normative text provided in section 2.3.5.1~~ of this specification
- 3.2.2.3 **MUST** implement the 'deny ipv4_connection' Command in accordance with Section 2.3.2 of this specification

~~**'allow ip_connection' and 'allow ip_addr' in accordance with the normative text provided in section 2.3.1**~~

3.2.3 Conformance Clause 14: IP Version 6 Connection Consumer

An OpenC2 Consumer satisfies 'IP Version 6 Connection Consumer' conformance if:

- 3.2.3.1 **MUST** meet all of conformance criteria identified in Conformance Clause 12 of this specification
- ~~'deny ip_connection' and 'deny ip_addr'~~3.2.3.2 **MUST** implement the 'allow ipv6_connection' Command in accordance with Section 2.3.1~~the normative text provided in section 2.3.2~~ of this specification
- 3.2.3.3 **MUST** implement the 'deny ipv6_connection' Command in accordance with Section 2.3.2 of this specification

3.2.4 Conformance Clause 15: IP Version 4 Net Consumer

An OpenC2 Consumer satisfies 'IP Version 4 Net Consumer' conformance if:

- 3.2.4.1 **MUST** meet all of conformance criteria identified in Conformance Clause 12 of this specification
- 3.2.4.2 **MUST** implement the 'allow ipv4_net' Command in accordance with Section 2.3.1 of this specification
- 3.2.4.3 **MUST** implement the 'deny ipv4_net' Command in accordance with Section 2.3.2 of this specification

3.2.5 Conformance Clause 16: IP Version 6 Net Consumer

An OpenC2 Consumer satisfies 'IP Version 6 Net Consumer' conformance if:

- 3.2.5.1 **MUST** meet all of conformance criteria identified in Conformance Clause 12 of this specification
- 3.2.5.2 **MUST** implement the 'allow ipv6_net' Command in accordance with Section 2.3.1 of this specification
- 3.2.5.3 **MUST** implement the 'deny ipv6_net' Command in accordance with Section 2.3.2 of this specification

3.2.6 Conformance Clause 17: Update File Consumer

An OpenC2 Consumer satisfies 'Update File Consumer' conformance if:

- 3.2.6.1 **MUST** meet all of the conformance criteria identified in Conformance Clause 12 of this specification
- 3.2.6.2 **MUST** implement the 'update file' Command in accordance with Section 2.3.5.1 of this specification

3.2.7 Conformance Clause 18: delete rule number Consumer

An OpenC2 Consumer satisfies 'delete rule Consumer' conformance if:

- 3.2.7.1 **MUST** meet all of the conformance criteria identified in Conformance Clause 12 of this specification
- 3.2.7.2 **MUST** implement the 'delete slpf:rule_number' in accordance with Section 2.3.4.1 of this specification

3.2.8 Conformance Clause 19: Running Consumer

An OpenC2 Consumer satisfies 'Running Consumer' conformance if:

- 3.2.8.1 **MUST** meet all of the conformance criteria identified in Conformance Clause 12 of this specification
- 3.2.8.2 **MUST** implement the 'running' Command Argument as a valid option for any Command associated with the 'deny' or 'allow' Actions in accordance with Section 2.3.1 and Section 2.3.2 of this specification

3.2.9 Conformance Clause 20: Direction Consumer

An OpenC2 Consumer satisfies 'Direction Consumer' conformance if:

- 3.2.9.1 **MUST** meet all of the conformance criteria identified in Conformance Clause 12 of this specification
- 3.2.9.2 **MUST** implement the 'direction' Command argument as a valid option for any Command associated with the 'deny' or 'allow' Actions in accordance with Section 2.3.1 and Section 2.3.2 of this specification

3.2.10 Conformance Clause 21: drop-process Consumer

An OpenC2 Consumer satisfies 'drop-process Consumer' conformance if:

- 3.2.10.1 **MUST** meet all of the conformance criteria identified in Conformance Clause 12 of this specification
- 3.2.10.2 **MUST** implement the 'drop_process' Command Argument as a valid option for any Command associated with the 'deny' Action in accordance with Section 2.3.1 and Section 2.3.2 of this specification

3.2.11 Conformance Clause 22: Temporal Consumer

An OpenC2 Consumer satisfies 'Temporal Consumer' conformance if:

- 3.2.11.1 **MUST** meet all of the conformance criteria identified in Conformance Clause 12 of this specification.
- 3.2.11.2 **MUST** implement the 'start_time' Command Argument as a valid option for any Command other than 'query features'

3. 3.2.11.3 **MUST** implement the 'stop_time' and 'duration' Command Arguments:
   1. **MUST** implement the start_time command argument as a valid option for any command other than 'query'

   - **MUST** implement the following command arguments as a valid option for any commandCommand other than 'query' and 'update file' 'query features' or 'update file'

     1. end_time
     2. duration
   2. **MUST** implement the following command arguments as a valid option for 'allow' and/or 'deny' commands
      1. running
      2. direction
   3. **MUST** implement the drop_process command argument as a valid option for the 'deny' command

---

Annex A SLPF Schema: Sample Commands

This annex defines the data objects used by conforming SLPF implementations, as shown in Section 2. This annex is normative, however in the event of a conflict between this annex, the property tables presented in section 2, and the separate plain text file linked below, the separate plain text file is authoritative.

Schema Files:

- Links to the schema files oc2slpf-v1.0.json (authoritative) and oc2slpf-v1.0.pdf (formatted) are listed in the section on the front page of this specification.

```
{
 "meta": {
  "module": "oasis-open.org/openc2/oc2slpf/v1.0/oc2slpf-v1.0",
  "patch": "0",
  "title": "Stateless Packet Filtering",
  "description": "Data definitions for Stateless Packet Filtering
(SLPF) functions",
  "exports": ["Target", "Specifiers", "Args", "Results"]
 },
```

```
 "types": {
   ["Target", "Choice", [], "", {
     [1, "rule_number", "Rule-ID", [], ""]]
   ],
   ["Args", "Map", [], "", {
     [1, "drop_process", "Drop-Process", ["[0"], ""],
     [2, "running", "Boolean", ["[0"], ""],
     [3, "direction", "Direction", ["[0"], ""],
     [4, "insert_rule", "Rule-ID", ["[0"], ""]]
   ],
   ["Drop-Process", "Enumerated", [], "", {
     [1, "none", ""],
     [2, "reject", ""],
     [3, "false_ack", ""]]
   ],
   ["Direction", "Enumerated", [], "", {
     [1, "ingress", ""],
     [2, "egress", ""]]
   ],
   ["Rule-ID", "Integer", [], ""],
   ["Specifiers", "Map", [], "", {
     [1, "hostname", "String", ["[0"], ""],
     [2, "named_group", "String", ["[0"], ""],
     [3, "asset_id", "String", ["[0"], ""],
     [4, "asset_tuple", "String", ["[0", "]10"], ""]]
   ],
   ["Results", "Map", [], "", {
     [1, "rule_number", "Rule-ID", ["[0"], ""]]
   ]]
}
```

Annex B Tailored OpenC2 Schema

This annex is a copy of the schema from the OpenC2 Language Specification tailored 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 may support additional elements of the OpenC2 language beyond those included here.

This annex *This section* *is non-normative*.

Schema Files:

- Links to the schema files oc2ls-v1.0-slpf.json (example) and oc2ls-v1.0-slpf.pdf (formatted) are listed in the section on the front page of this specification.

```
{
 "meta": {
  "module": "oasis-open.org/openc2/oc2ls/v1.0/oc2ls-v1.0",
  "patch": "0+slpf",
  "title": "OpenC2 Language Objects",
  "description": "OpenC2 Language content used by Stateless Packet
Filters.",
  "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", [], "", [
    [1, "action", "Action", [], ""],
    [2, "target", "Target", [], ""],
    [3, "args", "Args", ["[0"], ""],
    [4, "actuator", "Actuator", ["[0"], ""]
  ]],
  ["Action", "Enumerated", [], "", [
    [3, "query", ""],
    [6, "deny", ""],
    [8, "allow", ""],
    [16, "update", ""],
    [20, "delete", ""]
  ]],
  ["Target", "Choice", [], "", [
    [16, "features", "Features", [], ""],
    [10, "file", "File", [], ""],
    [11, "ip_addr", "IP-Addr", [], ""],
    [15, "ip_connection", "IP-Connection", [], ""],
    [1024, "slpf", "slpf:Target", [], ""]
  ]],
  ["Actuator", "Choice", [], "", [
    [1024, "slpf", "slpf:Specifiers", [], ""]
  ]],
  ["Args", "Map", [], "", [
    [1, "start_time", "Date-Time", ["[0"], ""],
    [2, "stop_time", "Date-Time", ["[0"], ""],
    [3, "duration", "Duration", ["[0"], ""],
    [4, "response_requested", "Response-Type", ["[0"], ""],
    [1024, "slpf", "slpf:Args", ["[0"], ""]
  ]],
  ["OpenC2-Response", "Map", [], "", [
    [1, "status", "Status-Code", ["[0"], ""],
    [2, "status_text", "String", ["[0"], ""],
    [6, "versions", "Version", ["[0", "]0"], ""],
    [7, "profiles", "jadn:Uname", ["[0", "]0"], ""],
```

```
    [8, "schema", "jadn:Schema", ["[0"], ""],
    [9, "pairs", "Action-Targets", ["[0", "]0"], ""],
    [10, "rate_limit", "Number", ["[0"], ""],
    [1024, "slpf", "slpf:Results", ["[0"], ""]
  ]],
  ["Status-Code", "Enumerated", ["="], "", [
    [102, "Processing", ""],
    [200, "OK", ""],
    [400, "Bad Request", ""],
    [500, "Internal Error", ""],
    [501, "Not Implemented", ""]
  ]],
  ["Features", "ArrayOf", ["*Feature", "[0"], ""],
  ["File", "Map", [], "", [
    [1, "name", "String", ["[0"], ""],
    [2, "path", "String", ["[0"], ""],
    [3, "hashes", "Hashes", ["[0"], ""]
  ]],
  ["IP-Addr", "Binary", ["@ip-addr"], ""],
  ["IP-Connection", "Record", [], "", [
    [1, "src_addr", "IP-Addr", ["[0"], ""],
    [2, "src_port", "Port", ["[0"], ""],
    [3, "dst_addr", "IP-Addr", ["[0"], ""],
    [4, "dst_port", "Port", ["[0"], ""],
    [5, "protocol", "L4-Protocol", ["[0"], ""]
  ]],
  ["Request-Id", "Binary", [], ""],
  ["Date-Time", "Integer", [], ""],
  ["Duration", "Integer", [], ""],
  ["Hashes", "Map", [], "", [
    [1, "md5", "Binary", ["[0"], ""],
    [4, "sha1", "Binary", ["[0"], ""],
    [6, "sha256", "Binary", ["[0"], ""]
  ]],
  ["L4-Protocol", "Enumerated", [], "", [
    [1, "icmp", ""],
    [6, "tcp", ""],
    [17, "udp", ""],
    [132, "sctp", ""]
  ]],
  ["Port", "Integer", ["[0", "]65535"], ""],
  ["Feature", "Enumerated", [], "", [
    [1, "versions", ""],
    [2, "profiles", ""],
    [3, "schema", ""],
    [4, "pairs", ""],
    [5, "rate_limit", ""]
  ]],
  ["Response-Type", "Enumerated", [], "", [
```

```
    [0, "none", ""],
    [1, "ack", ""],
    [2, "status", ""],
    [3, "complete", ""]
  ]],
  ["Version", "String", [], ""],
  ["Action-Targets", "Array", [], "", [
    [1, "action", "Action", [], ""],
    [2, "targets", "Target", ["]0", "*"], ""]
  ]]
 ]
}
```

Annex C Sample commands (Informative)

This section will summarize and provide examples of OpenC2 cCommands as they pertain to SLPF firewalls. The sample cCommands will be encoded in verbose JSON, however other encodings are possible provided the cCommand is validated against the property tables defined in Section 2schema presented in Annex A. of this specification. Examples of corresponding responses will be Responses are 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.

The following examples include Binary fields which are serialized in Base64url format. The examples show JSON-serialized cCommands; the conversion of Base64url serialized values to Binary data and String display text is:

Base64url	Binary	Display String
AQIDBA	01020304	1.2.3.4
xgIDBA	c6020304	198.2.3.4
xjNkEQ	c6336411	198.51.100.17

The examples include Integer Date-Time fields; the conversion of Integer values to String display text is:

Integer	Display String
`1534775460000`	`Monday, August 20, 2018 2:31:00 PM GMT, 2018-08-20T10:31:00-04:00`

C=======

A.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~~Targets, Specifiers, Arguments and corresponding ~~r~~Responses are applicable to both ~~a~~Actions.

~~C~~A.1.1 Deny a particular connection

Block a particular connection within the domain and do not send a host unreachable. Note, the "slpf":{"drop_process"} argument does not apply to the allow Action.

~~Command:~~

```
{
  "action": "deny",
  "target": {
    "ip_connection": {
      "protocol": "tcp",
```

Command:

```
{
  "action": "deny",
  "target": {
    "ipv4 connection": {
      "protocol": "tcp",
      "src_addr": "AQIDBA1.2.3.4",
      "src_port": 10996,
      "dst_addr": "xgIDBA198.2.3.4",
      "dst_port": 80
    }
  },
  "args": {
```

```
        "start_time": 1534775460000,
        "duration": 500,
        "response_requested": "ack",
    "slpf": {
        ":drop_process": "none"
    }
      },
      "actuator": {
    "slpf": {
        ":asset_id": "30"
    }
      }
    }
```

Response:

```
{
    "status": 200
}
```

CA.1.2 BlockDeny all outbound ftp transfers

Block all outbound ftp data transfers, send false acknowledgement and request ack. Note that the five-tuple is incomplete. Note that the response_type field was not populated therefore will be ''complete''. Also note that the aActuator called out was SLPF with no additional sSpecifiers, therefore all endpoints that can execute the cCommand should. Note, the "slpf":{"drop_process"} argument does not apply to the allow Action.

Command:

```
{
  "action": "deny",
  "target": {
    "ip_connection": {
      "protocol": "tcp",
```

Command:

```
{
  "action": "deny",
  "target": {
    "ip_connection": {
      "protocol": "tcp",
          "src_port": 21
        }
      },
      "args": {
    "slpf": {
```

```
      ":drop_process": "false_ack"",
      +"slpf:direction": "egress"
    },
    "actuator": {
      "slpf": {}
    }
}
```

Responses:

Case One: the aActuator successfully issued the deny.

```
{"status": 200}
```

Case Two: the cCommand failed due to a syntax error in the cCommand. Optional status text canis ignored by the Producer, but may be added to provide error details for debugging or logging.

```
{
  "status": 400,
  "status_text": "Validation Error: Target: ip_conection"
}
```

Case Three: the cCommand failed because an aArgument was not supported.

```
{
  "status": 501
}
```

CA.1.3 Block all inbound traffic from a particular source.

Block all inbound traffic from 1.2.3.4the specified ipv6 network and do not respond. In this case the ip_addr targetipv6_net Target and the direction argument was used. In this case only the perimeter filters should update the rule.

Command:

```
{
  "action": "deny",
  "target": {
    "ip_addr": "AQIDBA"
    "ipv6_net": "3ffe:1900:4545:3:200:f8ff:fe21:67cf"
  },
  "args": {
    "response_requested": "none",
  "slpf": {
    ":direction": "ingress"
```

```
    }
      },
      "actuator": {
    "slpf": {
        ":named_group": "perimeter"
      }
  }
}
}
```

A.1.4 Permit ftp transfers to a particular destination.

Permit ftp data transfers to ip address 198.51.100.173ffe:1900:4545:3::f8ff:fe21:67cf from any source. (Note that an actual application would also need to allow ftp-data (port 20) in order for transfers to be permitted.)).

Command:

```
{
  "action": "allow",
  "target": {
    "ipv6_connection": {
      "protocol": "tcp",
      "dst_addr":
"xjNkEQ3ffe:1900:4545:3::f8ff:fe21:67cf",
      "src_port": 21
    }
  },
  "actuator": {
    "slpf": {}
  }
}
```

In this case the aActuator returned a rule number associated with the allow.

Response:

```
{
  "status": 200,
  "slpf": {
    ":rule_number": 1234
  }
}
}
```

A.2 Delete Rule

Used to remove a firewall rule rather than issue an allow or deny to counteract the effect of an existing rule. Implementation of the ‘'delete slpf:rule_~~number' command~~number' Command is OPTIONAL.

In this case the rule number assigned in a previous allow will be removed (refer to the final example in Annex A.1~~section C.1)~~

Command:

```
{
  "action": "delete",
  "target": {
    "slpf": {
      ":rule_number": 1234
    }
  },
  "args": {
    "response_requested": "complete"
  },
  "actuator": {
    "slpf": {}
  }
}
```

~~C~~A.3 Update file

Implementation of the Update ~~a~~Action is optional. Update is intended for the device to process new configuration files. The update ~~a~~Action is a compound ~~a~~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 ~~t~~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 ~~a~~Actuator.

Command:

```
{
  "action": "update",
  "target": {
    "file": {
      "path": "\\\\someshared-
drive\\somedirectory\\configurations",
      "name": "firewallconfiguration.txt"
    }
  },
```

```
    "actuator": {
  "slpf": {
    ":named_group": "network"
  }
    }
}
```

Responses:

Successful update of the configuration

```
{"status": 200}
```

This aActuator does not support the update file cCommand

```
{
  "status": 501,
  "status_text": "Update-File Not Implemented"
}
```

This aActuator could not access the file

```
{
  "status": 500,
  "status_text": "Server error, Cannot access file"
}
```

CA.4 Query openc2features

Implementation of query cOpenc2 is required. The query openc2 commandfeatures Command is intended to enable the openc2 producerOpenc2 Producer to determine the capabilities of the aActuator. The query openc2 commandfeatures Command can also be used to check the status of the aActuator.

CA.4.1 No query items set

This cCommand uses query openc2features with no query items to verify that the aActuator is functioning.

Command:

```
{
  "action": "query",
  "target": {
    "openc2": []
  }
}
```

Response:

The ~~a~~Actuator is alive.

```
{"status": 200}
```

~~C~~A.4.2 Version of Language specification supported

This ~~c~~Command queries the ~~a~~Actuator to determine which version(s) of the language specification are supported. The language specifications use semantic versioning ("major.minor"); for each supported major version the ~~a~~Actuator need only report the highest supported minor version.

Command:

```
{
  "action": "query",
  "target": {
    "~~openc2~~features": ["versions"]
  }
}
```

Response:

The Actuator supports language specification versions 1.0 - 1.3.

```
{
  "status": 200,
  "versions": ["1.3"]
}
```

~~C~~A.4.3 Actuator profiles supported

This ~~c~~Command queries the ~~a~~Actuator to determine both the language versions and the ~~actuator~~ profiles supported.

Command:

```
{
  "action": "query",
  "target": {
    "~~openc2~~features": ["versions", "profiles"]
  }
}
```

Response:

The ~~a~~Actuator device is apparently a smart ~~toaster~~front-door-lock for which an extension ~~actuator~~ profile has been written. The device supports both the

standard slpf functions and whatever cCommands are defined in the extension profile.

```
{
  "status": 200,
{
  "status": 200,
    "versions": ["1.3"],
  "profiles": {
    "oasis-open.org/openc2/v1.0/ap-["slpf",
      "example.com/openc2/products/ "iot-toaster"front-
door-lock"]
  }
}
}
```

A.4.4 Specific Commands Supported

This cCommand queries the aActuator to determine which action-targetAction/Target pairs are supported. Not all tTargets are meaningful in the context of a specific aAction, and although a cCommand such as "update ip_connection" may be syntactically valid, the combination does not specify an operation supported by the aActuator.

Command:

For each supported aAction list the tTargets supported by this aActuator.

```
{
  "action": "query",
  "target": {
    "openc2features": ["pairs"]
  }
}
```

Response:

The aActuator supports all action-targetAction/Target pairs shown in Table 2.3-1 - Command Matrix.

```
{
  "status": 200,
{
  "status": 200,
  "pairs": [
    ["allow", ["ip_addr", "ipipv6_net",
"ipv6_connection"]],
    ["deny", ["ip_addr", "ipipv6_net",
"ipv6_connection"]],
```

```
    ["query", ["openc2features"]],
    ["delete", ["slpf:rule_number"]],
    ["update", ["file"]]
  ]
}
```

C.4.5 Actuator Schema

Annex B: Acronyms

This command queries the actuator for the syntax definition for all supported commands.

Command:

```
{
  "action": "query",
  "target": {
    "openc2": ["schema"]
  }
}
```

Response:

The result*section* *is* a single schema defining the syntax of all commands supported by this actuator. It is constructed from:

1. the tailored OpenC2 schema module (Annex B), merged with
2. each imported module (e.g., the SLPF schema module of Annex A, schemas from other profiles supported by this actuator), and,
3. further tailored for the specific actuator product by removing any unsupported optional elements.

Schema File:

The *non-normative* merged schema example (oc2ls-v1.0-slpf-merged.json) shown in this response is provided as a separate file, listed in the section on the front page of this specification.

```
{
 "status": 200,
 "schema": {
 "meta": {
  "module": "oasis-open.org/openc2/oc2ls/v1.0/oc2ls-v1.0",
  "patch": "0+slpf.merged",
  "title": "OpenC2 Language Objects",
```

```
  "description": "OpenC2 Language content used by Stateless Packet
Filters.",
  "exports": ["OpenC2-Command", "OpenC2-Response"]
 },
 "types": [
  ["OpenC2-Command", "Record", [], "", [
    [1, "action", "Action", [], ""],
    [2, "target", "Target", [], ""],
    [3, "args", "Args", ["[0"], ""],
    [4, "actuator", "Actuator", ["[0"], ""]
  ]],
  ["Action", "Enumerated", [], "", [
    [3, "query", ""],
    [6, "deny", ""],
    [8, "allow", ""],
    [16, "update", ""],
    [20, "delete", ""]
  ]],
  ["Target", "Choice", [], "", [
    [16, "features", "Features", [], ""],
    [10, "file", "File", [], ""],
    [11, "ip_addr", "IP-Addr", [], ""],
    [15, "ip_connection", "IP-Connection", [], ""],
    [1024, "slpf", "slpf:Target", [], ""]
  ]],
  ["Actuator", "Choice", [], "", [
    [1024, "slpf", "slpf:Specifiers", [], ""]
  ]],
  ["Args", "Map", [], "", [
    [1, "start_time", "Date-Time", ["[0"], ""],
    [2, "stop_time", "Date-Time", ["[0"], ""],
    [3, "duration", "Duration", ["[0"], ""],
    [4, "response_requested", "Response-Type", ["[0"], ""],
    [1024, "slpf", "slpf:Args", ["[0"], ""]
  ]],
  ["OpenC2-Response", "Map", [], "", [
    [1, "status", "Status-Code", ["[0"], ""],
    [2, "status_text", "String", ["[0"], ""],
    [6, "versions", "Version", ["[0", "]0"], ""],
    [7, "profiles", "jadn:Uname", ["[0", "]0"], ""],
    [8, "schema", "jadn:Schema", ["[0"], ""],
    [9, "pairs", "Action-Targets", ["[0", "]0"], ""],
    [10, "rate_limit", "Number", ["[0"], ""],
    [1024, "slpf", "slpf:Results", ["[0"], ""]
  ]],
  ["Status-Code", "Enumerated", ["="], "", [
    [102, "Processing", ""],
    [200, "OK", ""],
    [301, "Moved Permanently", ""],
```

```
    [400, "Bad Request", ""],
    [401, "Unauthorized", ""],
    [403, "Forbidden", ""],
    [404, "Not Found", ""],
    [500, "Internal Error", ""],
    [501, "Not Implemented", ""],
    [503, "Service Unavailable", ""]
  ]],
  ["Features", "ArrayOf", ["*Feature", "[0"], ""],
  ["File", "Map", [], "", [
    [1, "name", "String", ["[0"], ""],
    [2, "path", "String", ["[0"], ""],
    [3, "hashes", "Hashes", ["[0"], ""]
  ]],
  ["IP-Addr", "Binary", ["@ip-addr"], ""],
  ["IP-Connection", "Record", [], "", [
    [1, "src_addr", "IP-Addr", ["[0"], ""],
    [2, "src_port", "Port", ["[0"], ""],
    [3, "dst_addr", "IP-Addr", ["[0"], ""],
    [4, "dst_port", "Port", ["[0"], ""],
    [5, "protocol", "L4-Protocol", ["[0"], ""]
  ]],
  ["Request-Id", "Binary", [], ""],
  ["Date-Time", "Integer", [], ""],
  ["Duration", "Integer", [], ""],
  ["Hashes", "Map", [], "", [
    [1, "md5", "Binary", ["[0"], ""],
    [4, "sha1", "Binary", ["[0"], ""],
    [6, "sha256", "Binary", ["[0"], ""]
  ]],
  ["L4-Protocol", "Enumerated", [], "", [
    [1, "icmp", ""],
    [6, "tcp", ""],
    [17, "udp", ""],
    [132, "sctp", ""]
  ]],
  ["Port", "Integer", ["[0", "]65535"], ""],
  ["Feature", "Enumerated", [], "", [
    [1, "versions", ""],
    [2, "profiles", ""],
    [3, "schema", ""],
    [4, "pairs", ""],
    [5, "rate_limit", ""]
  ]],
  ["Response-Type", "Enumerated", [], "", [
    [0, "none", ""],
    [1, "ack", ""],
    [2, "status", ""],
    [3, "complete", ""]
```

```
  ]],
  ["Version", "String", [], ""],
  ["Action-Targets", "Array", [], "", [
    [1, "action", "Action", [], ""],
    [2, "targets", "Target", ["]0", "*"], ""]
  ]],
  ["slpf:Target", "Choice", [], "", [
    [1, "rule_number", "slpf:Rule-ID", [], ""]
  ]],
  ["slpf:Args", "Map", [], "", [
    [1, "drop_process", "slpf:Drop-Process", ["[0"], ""],
    [2, "running", "Boolean", ["[0"], ""],
    [3, "direction", "slpf:Direction", ["[0"], ""],
    [4, "insert_rule", "slpf:Rule-ID", ["[0"], ""]
  ]],
  ["slpf:Drop-Process", "Enumerated", [], "", [
    [1, "none", ""],
    [2, "reject", ""],
    [3, "false_ack", ""]
  ]],
  ["slpf:Direction", "Enumerated", [], "", [
    [1, "ingress", ""],
    [2, "egress", ""]
  ]],
  ["slpf:Rule-ID", "Integer", [], ""],
  ["slpf:Specifiers", "Map", [], "", [
    [1, "hostname", "String", ["[0"], ""],
    [2, "named_group", "String", ["[0"], ""],
    [3, "asset_id", "String", ["[0"], ""],
    [4, "asset_tuple", "String", ["[0", "]10"], ""]
  ]],
  ["slpf:Results", "Map", [], "", [
    [1, "rule_number", "slpf:Rule-ID", ["[0"], ""]
  ]],
  ["jadn:Schema", "Record", [], "", [
    [1, "meta", "jadn:Meta", [], ""],
    [2, "types", "jadn:Type", ["]0"], ""]
  ]],
  ["jadn:Meta", "Map", [], "", [
    [1, "module", "jadn:Uname", [], ""],
    [2, "patch", "String", ["[0"], ""],
    [3, "title", "String", ["[0"], ""],
    [4, "description", "String", ["[0"], ""],
    [5, "imports", "jadn:Import", ["[0", "]0"], ""],
    [6, "exports", "jadn:Identifier", ["[0", "]0"], ""],
    [7, "bounds", "jadn:Bounds", ["[0"], ""]
  ]],
  ["jadn:Import", "Array", [], "", [
    [1, "nsid", "jadn:Nsid", [], ""],
```

```
    [2, "uname", "jadn:Uname", [], ""]
  ]],
  ["jadn:Bounds", "Array", [], "", {
    [1, "max_msg", "Integer", [], ""],
    [2, "max_str", "Integer", [], ""],
    [3, "max_bin", "Integer", [], ""],
    [4, "max_fields", "Integer", [], ""]
  ]],
  ["jadn:Type", "Array", [], "", {
    [1, "tname", "jadn:Identifier", [], ""],
    [2, "btype", "jadn:JADN-Type", ["*"], ""],
    [3, "opts", "jadn:Option", ["]0"], ""],
    [4, "desc", "String", [], ""],
    [5, "fields", "jadn:JADN-Type", ["&btype", "]0"], ""]
  ]],
  ["jadn:JADN-Type", "Choice", [], "", {
    [1, "Binary", "Null", [], ""],
    [2, "Boolean", "Null", [], ""],
    [3, "Integer", "Null", [], ""],
    [4, "Number", "Null", [], ""],
    [5, "Null", "Null", [], ""],
    [6, "String", "Null", [], ""],
    [7, "Array", "jadn:FullField", ["]0"], ""],
    [8, "ArrayOf", "Null", [], ""],
    [9, "Choice", "jadn:FullField", ["]0"], ""],
    [10, "Enumerated", "jadn:EnumField", ["]0"], ""],
    [11, "Map", "jadn:FullField", ["]0"], ""],
    [12, "Record", "jadn:FullField", ["]0"], ""]
  ]],
  ["jadn:EnumField", "Array", [], "", {
    [1, "", "Integer", [], ""],
    [2, "", "String", [], ""],
    [3, "", "String", [], ""]
  ]],
  ["jadn:FullField", "Array", [], "", {
    [1, "", "Integer", [], ""],
    [2, "", "jadn:Identifier", [], ""],
    [3, "", "jadn:Identifier", [], ""],
    [4, "", "jadn:Options", [], ""],
    [5, "", "String", [], ""]
  ]],
  ["jadn:Identifier", "String", ["$^[a-zA-Z][\\w-]*$", "[1",
"]32"], ""],
  ["jadn:Nsid", "String", ["$^[a-zA-Z][\\w-]*$", "[1", "]8"], ""],
  ["jadn:Uname", "String", ["[1", "]100"], ""],
  ["jadn:Options", "ArrayOf", ["*jadn:Option", "[0", "]10"], ""],
  ["jadn:Option", "String", ["[1", "]100"], ""]
]
}
```

```
}
```

Term	Expansion
CoAP	Constrained Application Protocol
FTP	File Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
IACD	Integrated Adaptive Cyber Defense
IPR	Intellectual Property Rights
JADN	JSON Abstract Data Notation
JSON	JavaScript Object Notation
MQTT	Message Queuing Telemetry Transport
OASIS	Organization for the Advancement of Structured Information Standards
OODA	Observe-Orient-Decide-Act
OpenDXL	Open-source Data Exchange Layer
RFC	Request for Comment
SLPF	Stateless Packet Filter
TC	Technical Committee
URI	Uniform Resource Identifier

Annex ~~D~~C: Acknowledgments

This section is non-normative

The Actuator Profile Subcommittee was tasked by the OASIS Open Command and Control Technical Committee (OpenC2 TC) which at the time of this submission, had 132 members. The editors wish to express their gratitude to the members of the OpenC2 TC.

The following individuals are acknowledged for providing comments, suggested text and/or participation in the SLPF CSD ballots:

- Barry, Michelle- AT&T
- Brule, Joseph - National Security Agency
- Darley, Trey- New Context
- Darnell, David- North American Energy Standards Board
- Everett, Alex- University of North Carolina at Chapel Hill
- Farrel, Travis- Anomali
- Gray, Anderson- ForeScout
- Gurney, John-Mark- New Context
- Hamilton, David- AT&T
- Hunt, Christian- New Context
- Jackson, April- G2, Inc.
- Kakumaru, Takahiro- NEC Corporation
- Kasavchenko, Kirill- Arbor Networks
- Kemp, Dave- National Security Agency
- Lemire, Dave- G2, Inc.
- MacGregor, Scott- McAfee
- Martinez, Danny- G2, Inc.
- Mathews, Lisa- Department of Defense
- Mavroeidis, Vasileios- University of Oslo
- Ortiz, Efrain- Symantec
- Rajarathnam, Nirmal- ForeScout
- Riedel, Daniel- New Context
- Romano, Jason- National Security Agency
- Royer, Phillip- Splunk
- Skeen, Duane- Northrup Grumman
- Sparrell, Duncan- sFractal Consulting
- Stair, Michael- AT&T
- Storms, Andrew- New Context
- Stueve, Gerald- Fornetix
- Trost, Bill- AT&T
- Voit, Eric- Cisco
- Webb, Jason- LookingGlass
- White, Chuck- Fornetix
- Yu, Sounil- Bank of America

Annex ~~E~~D: Revision History

This section is non-normative

Revision	Date	Editor	Changes Made
Committee Specification Draft 1	31 AUG 2018	Brule, Joe	Initial draft
Committee Specification Draft 2	04 OCT 2018	Brule, Joe	Added Document overview, complete rewrite of introduction, modified components section to be consistent with Language Specification and address ballot comments, added schema, added conformance section, added examples, added acknowledgements section.
Committee Specification Draft 3	16 OCT 2018	Brule, Joe	Aligned section 1 with other OpenC2 specifications; other changes to track dependencies on the language specification: 1) replace openc2 target with features target, 2) flatten response examples so that there is not a separate "results" layer.
Working Draft 06	28 MAR 2019	Brule, Joe	Addressed Public Review 01 comments.