OASIS OPEN

OpenC2 Actuator Profile for Packet Filtering Version 1.0

Committee Specification Draft 02

08 August 2024

This stage:

https://docs.oasis-open.org/openc2/ap-pf/v1.0/csd02/ap-pf-v1.0-csd02.md (Authoritative)
https://docs.oasis-open.org/openc2/ap-pf/v1.0/csd02/ap-pf-v1.0-csd02.html
https://docs.oasis-open.org/openc2/ap-pf/v1.0/csd02/ap-pf-v1.0-csd02.pdf

Previous stage:

https://docs.oasis-open.org/openc2/ap-pf/v1.0/csd01/ap-pf-v1.0-csd01.md (Authoritative)
https://docs.oasis-open.org/openc2/ap-pf/v1.0/csd01/ap-pf-v1.0-csd01.html
https://docs.oasis-open.org/openc2/ap-pf/v1.0/csd01/ap-pf-v1.0-csd01.pdf

Latest stage:

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

Technical Committee:

OASIS Open Command and Control (OpenC2) TC

Chairs:

Duncan Sparrell (duncan@sfractal.com), sFractal Consulting LLC
Michael Rosa (mjrosa@cyber.nsa.gov), National Security Agency

Editors:

Alex Everett (alex.everett@unc.edu), University of North Carolina, Chapel Hill
Vasileios Mavroeidis (vasileim@ifi.uio.no), University of Oslo

Additional artifacts:

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

- JADN schema: schemas/pf-ap.jadn

Abstract:

Open Command and Control (OpenC2) is a concise and extensible language to enable machine-to-machine communications for purposes of command and control of cyber defense components, subsystems, and systems in a manner that is agnostic of the underlying products, technologies, transport mechanisms, or other aspects of the implementation. This specification defines an Actuator profile for Packet Filtering (PF). Packet filtering is a cyber defense mechanism that denies or allows traffic based on static or dynamic properties. The Actuator profile collects Actions, Targets, Arguments, and Specifiers along with conformance clauses to enable the operation of OpenC2 Producers and Consumers in the context of PF.

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 stage" 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 https://groups.oasis-open.org/communities/community-home?CommunityKey=9ae0f0f9-24b5-44ea-9fe7-018dce260e09.

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.

Key words:

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 BCP 14 [RFC2119] and [RFC8174] when, and only when, they appear in all capitals, as shown here.

Citation format:

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

[OpenC2-PF-v1.0]
OpenC2 Actuator Profile for Packet Filtering Version 1.0. Edited by Alex Everett and Vasileios Mavroeidis. 08 August 2024. OASIS Committee Specification Draft 02. https://docs.oasis-open.org/openc2/ap-pf/v1.0/csd02/ap-pf-v1.0-csd02.html. Latest stage: https://docs.oasis-open.org/openc2/ap-pf/v1.0/ap-pf-v1.0.html.

Notices:

Table of Contents

1 Introduction

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

Note: This Actuator profile is consistent with Version 1.0 of the OpenC2 Language Specification ([OpenC2-Lang-v1.0]).

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 Command is encoded by a Producer (managing application) and transferred to a Consumer (managed device or virtualized function) using a secure transfer protocol, and the Consumer acts on the request and responds with status and any other requested information.

This specification defines an Actuator profile for **Packet Filtering (PF)**. In particular, the specification comprises a set of Actions, Targets and Target Specifiers, Command Arguments, and Actuator Specifiers that integrates PF functionality with the OpenC2 Command set. Through this Command set, cyber security orchestrators may gain visibility into and provide control over PF functionality in a manner that is independent of the instance of the PF function.

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

The rest of the specification is organized as follows:

The remaining of Section 1 includes information about the terminology used, and document conventions pertinent to this Actuator profile specification.

Section 2 (normative) binds this particular profile to Version 1.0 of the OpenC2 Language Specification ([OpenC2-Lang-v1.0]). It enumerates the components of the Language Specification that are meaningful in the context of PF and also defines components that are applicable to this distinct profile. In addition, Section 2 defines the Commands (i.e., the Action/Target pairs, arguments, an associated specifiers) that are permitted in the context of PF.

Section 3 (normative) presents definitive criteria for conformance so that cyber security stakeholders can be assured that their products, instances and/or integrations are compatible with this profile (OpenC2 Actuator Profile for Packet Filtering Version 1.0).

Appendix E (non-normative) provides multiple examples of Commands and associated Responses (JSON serialization).

1.1 Changes from Earlier Versions

Not applicable: initial publication.

1.2 Glossary

1.2.1 Definition of Terms

This section is normative.

- **Action**: The task or activity to be performed (e.g., 'deny').
- **Actuator**: The function performed by the Consumer that executes the Command (e.g., 'Stateless Packet Filtering').
- **Argument**: A 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 pair that is sent from a Producer and received by a Consumer.
- **Consumer**: A managed device / application that receives Commands. Note that a single device / application can have both Consumer and 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 Message from a Consumer to a Producer acknowledging a Command or returning the requested resources or status to a previously received Command.
- **Specifier**: A property or field that identifies a Target or Actuator to some level of precision.
- **Target**: The object of the Action, i.e., the Action is performed on the Target (e.g., IP Address).

1.2.2 Acronyms and Abbreviations

This section is non-normative.

Acronym	Expansion
AP	Actuator Profile
IANA	Internet Assigned Numbers Authority
ICMP	Internet Control Message Protocol
IP	Internet Protocol
JADN	JSON Abstract Data Notation
JSON	Javascript Object Notation
PF	Packet Filtering
SCTP	Stream Control Transmission Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

1.3 Document Conventions

1.3.1 Naming Conventions

- [RFC2119]/[RFC8174] key words 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).
- 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.

1.3.2 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_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 ellipses (...).

Example:

```
{
    "action": "deny",
    "target": {
        "file": {
            "hashes": {
                "sha256": "22fe72a34f006ea67d26bb7004e2b6941b5c3953d43ae7ec24d41b1a928a6973"
            }
        }
    }
}
```

1.3.3 Schema

The schema for this AP is defined using a [JSON Abstract Data Notation (JADN)] information model. The property tables in this document were generated programmatically from the JADN schema for consistency.

2 OpenC2 Language Binding

This section is normative.

This section defines the set of Actions, Targets, Arguments, and Actuator Specifiers that are meaningful in the context of PF and the appropriate status codes, status texts, and other properties of a Response message. In addition, this section defines the Commands allowed by this Actuator profile. Section 2 is organized into three major subsections; Command Components, Response Components, and Commands.

All components, devices, and systems that provide PF functionality MUST implement the identified OpenC2 Actions, Targets, Specifiers, and Arguments as specified in the Conformance section of this specification.

Extensions to the Language Specification are defined in accordance with Version 1.0 of the OpenC2 Language Specification, Section 3.1.4, where:

1. The unique name of the PF schema is: `http://oasis-open.org/openc2/ap-pf/v1.0`
2. The namespace identifier (nsid) referring to the PF schema is: `pf`
3. The conformance requirements for the OpenC2 Packet Filtering Actuator profile are defined and included in this document.

2.1 OpenC2 Command Components

The components of an OpenC2 Command include Actions, Targets, Actuators and associated Arguments and Specifiers. Appropriate aggregation of the components will define a Command-body that is meaningful in the context of PF.

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

- Action: A subset of the Actions defined in Version 1.0 of the OpenC2 Language Specification that are meaningful in the context of a packet filter.
  - This profile SHALL NOT define Actions that are external to Version 1.0 of the OpenC2 Language Specification.
  - This profile MAY augment the definition of the Actions in the context of PF.
  - This profile SHALL NOT define Actions in a manner that is inconsistent with Version 1.0 of the OpenC2 Language Specification.
- Target: A subset of the Targets and Target-Specifiers defined in Version 1.0 of the OpenC2 Language Specification that are meaningful in the context of PF and two Targets and their Specifiers that are defined in this specification.
- Arguments: A subset of the Arguments defined in Version 1.0 of the OpenC2 Language Specification and a set of Arguments defined in this specification.
- Actuator: A set of Actuator Specifiers defined in this specification that are meaningful in the context of PF.

2.1.1 Actions

Table 2.1.1-1 presents the Actions defined in Version 1.0 of the OpenC2 Language Specification which are meaningful in the context of PF. The particular Action/Target pairs that are valid combinations are presented in Section 2.3.

Table 2.1.1-1 Common Actions Applicable to PF

Type: Action (Enumerated)

Type: Actions (Enumerated)

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

2.1.2 Targets

Table 2.1.2-1 lists the Targets defined in Version 1.0 of the OpenC2 Language Specification that are applicable to PF. Table 2.1.2-2 extends the list of common Targets and includes additional Targets unique to PF. Targets that are defined in this profile (see Table 2.1.2-2) are referenced with the `pf` namespace identifier.

Table 2.1.2-1 Common Targets Applicable to PF

Type: Target (Choice)

ID	Name	Type	#	Description
9	**features**	ls:Feature	1	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.
10	**file**	ls:File	1	Properties of a file.
13	**ipv4_net**	ls:IPv4-Net	1	The representation of one or a block of IPv4 addresses expressed using CIDR notation.
14	**ipv6_net**	ls:IPv6-Net	1	The representation of one or a block of IPv6 addresses expressed using CIDR notation.
15	**ipv4_connection**	ls:IPv4-Connection	1	A network connection as specified by a five-tuple (IPv4).
16	**ipv6_connection**	ls:IPv6-Connection	1	A network connection as specified by a five-tuple (IPv6).
17	**domain_name**	ls:Domain-Name	1	A domain name as defined in [RFC1034].
1034	**pf**	String	1	

Usage Requirements:

- ipv4_connection
  - If the protocol is 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 is 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 is 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 is 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.

Table 2.1.2-2 Targets Unique to PF

Type: PF-Target (Choice)

ID	Name	Type	#	Description
1	**rule_number**	Rule-ID	1	Immutable identifier assigned when a packet filtering rule is created. Identifies the rule to be deleted or used to request information about a rule.
2	**adv_connection**	Advanced-Connection	1	Advanced connection type to support application layer firewalls

2.1.2.1 Data Type Definitions

Type Name	Type Definition	Description
Rule-ID	Integer{0..*}	

Type: Advanced-Connection (Record)

ID	Name	Type	#	Description
1	**src_addr**	Adv-Addr	1	Source address range, one of IPv4, IPv6, or network tag
2	**src_port**	ls:Port	1	Source service per [RFC6335]
3	**dst_addr**	Adv-Addr	1	Destination address range, one of IPv4, IPv6, or network tag
4	**dst_port**	ls:Port	1	Destination service per [RFC6335]
5	**protocol**	ls:L4-Protocol	1	Layer 4 protocol (e.g., TCP) - see Section 3.4.2.11 of the OpenC2 Language Specification
6	**network**	String	1	Reference to the name (also known as tag) of logical network to which the rule applies
7	**application**	String	1	Reference to the name of the application to which the rule applies

Usage Requirements:

- advanced_connection
  - The seven-tuple is: src_addr, src_port, dst_addr, dst_port, protocol, network, and application. Any component, excluding network, not specified or specified as null MUST be treated as 'any'. When defined, src_port and dst_port MUST be an integer between 0 and 65535. When defined, src_addr and dst_addr MUST specify either an IPv4 address, IPv6 address, or a tag of type string. Application, typically used by next-generation firewalls, MUST be of type string. Network MUST be of type string being the reference to the name (also known as tag) of logical network to which the rule applies.

Type: Adv-Addr (Choice)

ID	Name	Type	#	Description
1	**v4addr**	ls:IPv4-Net	1	IPv4 CIDR block address as defined in the OpenC2 LS
2	**v6addr**	ls:IPv6-Net	1	IPv6 "CIDR block" address as defined in the OpenC2 LS
3	**net_tag**	String	1	A network name, e.g., as used in cloud system network definitions

2.1.3 Command Arguments

Arguments provide additional precision to a Command by including information such as how, when, or where a Command is to be executed. Table 2.1.3-1 lists the Command Arguments defined in Version 1.0 of the OpenC2 Language Specification as they relate to PF functionality. Table 2.1.3-2 lists the Command Arguments that are defined in this profile. Command Arguments that are defined in this profile (see Table 2.1.3-2) are referenced with the `pf` namespace identifier.

Table 2.1.3-1 Common Command Arguments Applicable to PF

Type: Args (Map{1..*})

ID	Name	Type	#	Description
1	**start_time**	ls:Date-Time	1	The specific date/time to initiate the Command.
2	**stop_time**	ls:Date-Time	1	The specific date/time to terminate the Command.
3	**duration**	ls:Duration	1	
4	**response_requested**	ls:Response-Type	1	
1034	**pf**	PF-ARgs	1	

Table 2.1.3-2 Command Arguments Unique to PF

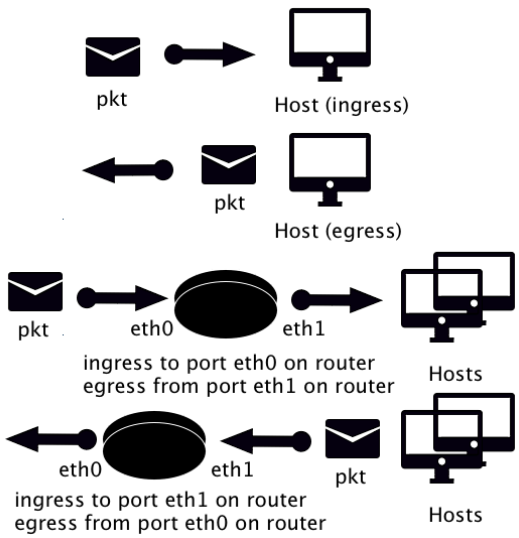
Type: PF-Args (Map{1..*})

ID	Name	Type	#	Description
1	**drop_process**	Drop-Process	1	Specifies how to handle denied packets
2	**persistent**	Boolean	1	Normal operations assume any changes to a device are to be implemented persistently. Setting the persistent modifier to FALSE results in a change that is not persistent in the event of a reboot or restart.
3	**direction**	Direction	1	Specifies whether to apply rules to incoming or outgoing traffic. If omitted, rules are applied to ingress packets.
4	**insert_rule**	Rule-ID	1	Specifies the identifier of the rule within a list, typically used in a top-down rule list.
5	**logged**	Boolean	1	Specifies if a log entry should be recorded as traffic matches the rule. The manner and mechanism for recording these entries is implementation specific and not defined by this specification.
6	**description**	String	1	A note to annotate or provide information regarding the rule.
7	**stateful**	Boolean	1	Specifies if the actuator should treat the request using state tables or connection state.
8	**priority**	Integer{0..*}	1	Specifies the priority of a specific firewall rule for firewalls that assign a numeric priority. It is used to determine which firewall rule takes precedence.

Usage Requirements:

- 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 an Openc2 Producer and an Openc2 Consumer - i.e., the value MUST map to exactly one 'deny [target]' or 'allow [target]' for a given instance of a PF.
- directionality:
  - If absent or not explicitly set, then the Command MUST apply to ingress packets.
- drop_process: If absent or not explicitly set, then the Actuator MUST NOT send any notification to the source of the packet.
- persistent: If absent or not explicitly set, then the value is TRUE and any changes are persistent.
- stateful:
  - If absent or not explicitly set, and the Actuator only operates in either stateful or stateless, the Command MUST apply as if this Argument was appropriately specified (e.g., stateful for Google Cloud Platform).
  - If the Actuator supports both mechanisms and this Argument is absent or not explicitly set, then it MUST treat the command as if the Argument was set to stateless in order to be backwards compatible with Version 1.0 of the OpenC2 Stateless Packet Filtering Actuator Profile ([OpenC2-SLPF-v1.0]).

Note that direction is required by some packet filters. For a host-based or host interface-based packet filter, ingress indicates a packet that originated from a different host. For a network-based packet filter, such as a router or a switch, ingress indicates a packet entering a physical or logical interface that your organization controls.



2.1.3.1 Data Type Definitions

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 acknowledgment.

Type: Direction (Enumerated)

ID	Name	Description
1	**both**	Apply rules to all traffic.
2	**ingress**	Apply rules to incoming traffic only.
3	**egress**	Apply rules to outgoing traffic only.

2.1.4 Actuator Specifiers

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

Table 2.1.4-1 lists the Specifiers that are applicable to the PF Actuator. Appendix E provides sample Commands with the use of Specifiers. The Actuator Specifiers defined in this profile are referenced with the `pf` namespace identifier.

Table 2.1.4-1 PF Specifiers

Type: Specifiers (Map)

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

2.2 OpenC2 Response Components

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

Responses associated with REQUIRED Actions MUST be implemented. Implementations that include OPTIONAL Actions MUST implement the Responses associated with the implemented Action. Additional details regarding Commands and associated Responses are captured in Section 2.3. Examples are provided in Appendix E. The structure of an OpenC2 Response is defined in Section 3.3.2 of the OpenC2 Language Specification.

2.2.1 Response Results

Table 2.2.1-1 lists the Response Results properties defined in Version 1.0 of the OpenC2 Language Specification that are applicable to PF and associated with the 'query features' Command. Table 2.2.1-2 extends the list of common Response Results properties and includes additional properties unique to PF. Response Results properties that are defined in this profile (see Table 2.2.1-2) are referenced with the `pf` namespace identifier.

Table 2.2.1-1 Common Response Results Applicable to PF

Type: Results (Map [1..*])

ID	Name	Type	#	Description
1	**versions**	Version	0..*	List of OpenC2 language versions supported by this Actuator.
2	**profiles**	ArrayOf(Nsid)	0..1	List of profiles supported by this Actuator.
3	**pairs**	Action-Targets	0..*	List of targets applicable to each supported Action.
4	**rate_limit**	Number	0..1	Maximum number of requests per minute supported by design or policy.

Table 2.2.1-2 Response Results Unique to PF

Type: PF-Results (Map{1..*})

ID	Name	Type	#	Description
1	**rule_number**	Rule-ID	1	Rule identifier returned from allow or deny Command.

2.2.2 Response Status Codes

Table 2.2.2-1 lists the Response Status Codes defined in Version 1.0 of the OpenC2 Language Specification that are applicable to PF.

Table 2.2.2-1 Response Status Codes

Type: Status-Code (Enumerated.ID)

Value	Description
102	**Processing** - an interim Response used to inform the Producer that the Consumer has accepted the Command but has not yet completed it.
200	**OK** - the Command has succeeded.
400	**Bad Request** - the Consumer cannot process the Command due to something that is perceived to be a Producer error (e.g., malformed Command syntax).
500	**Internal Error** - the Consumer encountered an unexpected condition that prevented it from performing the Command. For "response*requested" value "complete", one of the following MAY apply: * Cannot access file or path _ Rule number currently in use * Rule not updated
501	**Not Implemented** - the Consumer does not support the functionality required to perform the Command. For "response*requested" value "complete", one of the following MAY apply: * Target not supported _ Argument not supported * Command not supported

2.3 OpenC2 Commands

An OpenC2 Command consists of an Action/Target pair and associated Specifiers and Arguments. This section enumerates the Commands permitted by this Actuator profile and presents the associated Response behavior. Sections 2.3.1 to 2.3.5 provide details applicable to each Command, also as influenced by the Arguments.

Table 2.3-1 defines the Commands that are valid in the context of the PF profile. An Action (the top row in Table 2.3-1) paired with a Target (the first column in Table 2.3-1) defines a valid Command.

Table 2.3-1. Command Matrix

	Allow	Deny	Query	Delete	Update
ipv4_connection	valid	valid			
ipv6_connection	valid	valid			
ipv4_net	valid	valid			
ipv6_net	valid	valid			
pf:advanced_connection	valid	valid			
domain_name	valid	valid			
features			valid		
pf:rule_number			valid	valid	
file					valid

Table 2.3-2 defines the Command Arguments that are allowed for a particular Command by the PF Actuator profile. A Command (the top row in Table 2.3-2) paired with an Argument (the first column in Table 2.3-2) defines an allowable combination.

Table 2.3-2. Command Arguments Matrix

	Allow *target*	Deny *target*	Query features	Query pf:rule_number	Delete pf:rule_number	Update file
response_requested	2.3.1	2.3.2	2.3.3.1	2.3.3.2	2.3.4.1	2.3.5.1
start_time	2.3.1	2.3.2				
stop_time	2.3.1	2.3.2				
duration	2.3.1	2.3.2				
persistent	2.3.1	2.3.2				
direction	2.3.1	2.3.2				
insert_rule	2.3.1	2.3.2				
drop_process		2.3.2				
logged	2.3.1	2.3.2				
stateful	2.3.1	2.3.2				
priority	2.3.1	2.3.2				

Hereafter the specification provides details applicable to each Command, also as influenced by the Arguments.

2.3.1 Allow

Table 2.3-2, Command Arguments Matrix, summarizes the Command Arguments that apply to all Commands consisting of the allow Action and a valid Target type.

Upon receipt of a 'allow [target]' Command with an Argument that is not supported by the Actuator, PF Consumers:

- MUST NOT respond with the 200 status code.
- SHOULD respond with the 501 status code.
- SHOULD respond with "Argument not supported" in the status text.
- MAY respond with the 500 status code.

OpenC2 Producers that send 'allow [target]' Commands and support the 'delete pf:rule_number' Command or/and the 'query pf:rule_number' Command:

- MUST support the pf:rule_number Target type as defined in Table 2.1.2-2.
- SHOULD populate the Command Arguments field with '"response_requested" : "complete"'.
- MAY populate the Command Arguments field with the insert_rule Argument.
- MUST populate the Command Arguments field with '"response_requested" : "complete"' if the insert_rule Argument is populated.

OpenC2 Consumers that receive 'allow [target]' Commands:

- SHOULD respond with the Response status code 200 upon successful parsing of the 'allow [target]' Command and subsequent implementation of the corresponding rule.

OpenC2 Consumers that receive and successfully parse 'allow [target]' Commands but cannot implement the 'allow [target]':

- MUST NOT respond with the 200 status code.
- SHOULD respond with the 501 status code.
- SHOULD respond with 'Rule not implemented' in the status text.
- MAY respond with the 500 status code.

OpenC2 Consumers that receive 'allow [target]' Commands and support the 'delete pf:rule_number' Command or/and the 'query pf:rule_number' Command:

- MUST support the pf:rule_number Target type as defined in Table 2.1.2-2.
- Upon successful implementation of the 'allow [target]', MUST return the rule_number associated with the rule if the '"response_requested" : "complete"' Argument is populated.

OpenC2 Consumers that receive 'allow [target]' Commands and support the insert_rule Command Argument:

- MUST assign the rule number provided if the insert_rule Argument is populated.
- If the rule number is currently in use, then:
  - MUST NOT respond with the 200 status code.
  - SHOULD respond with the 500 status code.
  - SHOULD respond with 'Rule number currently in use' in the status text.

The valid Target types, associated Specifiers, and Arguments are summarized in the following subsections. Sample Commands are presented in Appendix E.

2.3.1.1 'Allow ipv4_connection'

The 'allow ipv4_connection' Command is OPTIONAL for Openc2 Producers implementing the PF profile.
The 'allow ipv4_connection' Command is OPTIONAL for Openc2 Consumers implementing the PF profile.

The Command permits traffic that is consistent with the specified ipv4_connection. A valid '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'.

OpenC2 Consumers that receive but do not implement the 'allow ipv4_connection' Command:

- MUST NOT respond with the 200 status code.
- 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 PF profile.
The 'allow ipv6_connection' Command is OPTIONAL for Openc2 Consumers implementing the PF profile.

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 ipv6_connection Target MUST be treated as an 'any'.

OpenC2 Consumers that receive but do not implement the 'allow ipv6_connection' Command:

- MUST NOT respond with the 200 status code.
- 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.

2.3.1.3 'Allow ipv4_net'

The 'allow ipv4_net' Command is OPTIONAL for Openc2 Producers implementing the PF profile.
The 'allow ipv4_net' Command is OPTIONAL for Openc2 Consumers implementing the PF profile.

The Command permits traffic as specified by the range of IPv4 addresses as expressed by CIDR notation. If the mask is unspecified then it MUST be treated as a single IPv4 address (i.e., an address range of one element). The address range specified in the ipv4_net MUST be treated as a source OR destination address.

OpenC2 Consumers that receive but do not implement the 'allow ipv4_net' Command:

- MUST NOT respond with the 200 status code.
- 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.

2.3.1.4 'Allow ipv6_net'

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

The Command permits traffic as specified by the range of IPv6 addresses as expressed by CIDR notation. If the mask is 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.

OpenC2 Consumers that receive but do not implement the 'allow ipv6_net' Command:

- MUST NOT respond with the 200 status code.
- 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.

2.3.1.5 'Allow domain_name'

The 'allow domain_name' Command is OPTIONAL for Openc2 Producers implementing the PF profile.
The 'allow domain_name' Command is OPTIONAL for Openc2 Consumers implementing the PF profile.

The Command permits traffic that is consistent with the specified domain name.

OpenC2 Consumers that receive but do not implement the 'allow domain_name' Command:

- MUST NOT respond with the 200 status code.
- 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.

2.3.1.6 'Allow advanced_connection'

The 'allow advanced_connection' Command is OPTIONAL for Openc2 Producers implementing the PF profile.
The 'allow advanced_connection' Command is OPTIONAL for Openc2 Consumers implementing the PF profile.

The Command permits traffic that is consistent with the specified advanced_connection. A valid 'allow advanced_connection' Command has at least one property of the advanced_connection populated and may have any combination of the seven properties populated. An unpopulated property, excluding network, within the advanced_connection Target MUST be treated as an 'any'.

OpenC2 Consumers that receive but do not implement the 'allow advanced_connection' Command:

- MUST NOT respond with the 200 status code.
- 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 Argument, the Targets, Specifiers, Arguments and corresponding Responses are identical to the six allow Commands. Table 2.3-2, Command Arguments Matrix, summarizes the Command Arguments that apply to all Commands consisting of the deny Action and valid Target types.

Upon receipt of a 'deny [target]' Command with an Argument that is not supported by the Actuator, PF Consumers:

- MUST NOT respond with the 200 status code.
- SHOULD respond with the 501 status code.
- SHOULD respond with 'Argument not supported' in the status text.
- MAY respond with the 500 status code.

OpenC2 Producers that send 'deny [target]' Commands and support the 'delete pf:rule_number' Command or/and the 'query pf:rule_number' Command:

- MUST support the pf:rule_number Target type as defined in Table 2.1.2-2.
- SHOULD populate the Command Arguments field with '"response_requested" : "complete"'.
- MAY populate the Command Arguments field with the insert_rule Argument .
- MUST populate the Command Arguments field with '"response_requested" : "complete"' if the insert_rule Argument is populated.

OpenC2 Consumers that receive 'deny [target]' Commands:

- SHOULD respond with Response status code 200 upon successful parsing of the 'deny [target]' Command and subsequent implementation of the corresponding rule.

OpenC2 Consumers that receive and successfully parse 'deny [target]' Commands but cannot implement the 'deny [target]' :

- MUST NOT respond with the 200 status code.
- SHOULD respond with the 501 status code.
- SHOULD respond with 'Rule not implemented' in the status text.
- MAY respond with the 500 status code.

OpenC2 Consumers that receive 'deny [target]' Commands and support the 'delete pf:rule_number' Command or/and the 'query pf:rule_number' Command:

- MUST support the pf:rule_number Target type as defined in Table 2.1.2-2.
- MUST return the rule number assigned in the pf object if the '"response_requested" : "complete"' Argument is populated.

OpenC2 Consumers that receive 'deny [target]' Commands and support the insert_rule Command Argument:

- MUST assign the rule number provided if the insert_rule Argument is populated.
- If the rule number is currently in use, then:
  - MUST NOT respond with the 200 status code.
  - SHOULD respond with the 500 status code.
  - SHOULD respond with 'Rule number currently in use' in the status text.

2.3.3 Query

The valid Target types and Arguments for the query Action are summarized in Table 2.3-2 Command Arguments Matrix. Sample Commands are presented in Appendix E.

Upon receipt of 'query [target]' Command with an Argument that is not supported by the Actuator, PF Consumers:

- MUST NOT respond with the 200 status code.
- SHOULD respond with the 501 status code.
- SHOULD respond with 'Argument not supported' in the status text.
- MAY respond with the 500 status code.

OpenC2 Consumers that receive 'query [target]' Commands:

- SHOULD respond with the Response status code 200 upon successful parsing and execution of the Command.

2.3.3.1 'Query features'

Implementation of the 'query features' Command is REQUIRED and MUST be implemented in accordance with Section 4.1, Implementation of 'query features' Command, of Version 1.0 of the OpenC2 Language Specification.

2.3.3.2 'Query pf:rule_number'

The 'query pf:rule_number' Command provides a mechanism to obtain similar information to that provided by creating a firewall rule. Implementation of the 'query pf:rule_number' Command is OPTIONAL. Products that choose to implement the 'query pf:rule_number' Command MUST implement the pf:rule_number Target type described in Table 2.1.2-2.

2.3.4 Delete

The pf:rule_number is the only valid Target type for the delete Action. Sample Commands are presented in Appendix E.

2.3.4.1 'Delete pf:rule_number'

The 'delete pf:rule_number' 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 pf:rule_number' Command is OPTIONAL. Products that choose to implement the 'delete pf:rule_number' Command MUST implement the pf:rule_number Target type described in Table 2.1.2-2.

OpenC2 Producers that send the 'delete pf:rule_number' Command:

- MAY populate the Command Arguments field with '"response_requested" : "complete"'.
- MUST NOT include other Command Arguments.
- MUST include exactly one rule number.

Upon receipt of a 'delete pf:rule_number' Command with an Argument that is not supported by the Actuator, PF Consumers:

- MUST NOT respond with the 200 status code.
- SHOULD respond with the 501 status code.
- SHOULD respond with 'Argument not supported' in the status text.
- MAY respond with the 500 status code.

OpenC2 Consumers that receive the 'delete pf:rule_number' Command:

- SHOULD respond with Response code 200 upon successful parsing of the 'delete pf:rule_number' Command and subsequent removal of the corresponding rule.
- upon successful parsing but failure to remove the corresponding rule:
  - MUST NOT respond with the 200 status code.
  - MUST respond with the 500 status code.
  - SHOULD respond with 'Firewall rule not removed or updated' in the status text.
- but cannot parse or process the 'delete pf:rule_number' Command:
  - MUST NOT respond with the 200 status code.
  - SHOULD respond with the 400 status code.
  - MAY respond with the 500 status code.
- but do not support the pf:rule_number Target type:
  - MUST NOT respond with the 200 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.

2.3.5 Update

The file Target as defined in Version 1.0 of the OpenC2 Language Specification is the only valid Target type for the update Action. The associated Specifiers, and Arguments are summarized in Section 2.3.5.1. Sample Commands are presented in Appendix E.

2.3.5.1 'Update file'

The 'update file' Command is used to replace or update files such as configuration files, rule sets, etc. Implementation of the update file Command is OPTIONAL. OpenC2 Consumers that choose to implement the 'update file' Command MUST 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 MUST be that the firewall operates with the new file at the conclusion of the 'update file' Command. The atomic steps that take place are implementation specific.

Table 2.3-2, Command Arguments Matrix, presents the valid Arguments for the 'update file' Command. OpenC2 Producers and Consumers that choose to implement the 'update file' Command MUST NOT include Arguments other than the one identified in Table 2.3-2.

OpenC2 Producers that send the 'update file' Command:

- MAY populate the arguments field with the response_requested Argument. Valid values for response_requested for 'update file' are "complete", "ack", and "none".
- MUST NOT include other Command Arguments.
- MUST populate the name Specifier in the Target.

- SHOULD populate the path Specifier in the Target.

Upon receipt of a 'update file' Command with an Argument that is not supported by the Actuator, PF Consumers:

- MUST NOT respond with the 200 status code.
- SHOULD respond with the 501 status code.
- SHOULD respond with 'Argument not supported' in the status text.
- MAY respond with the 500 status code.

OpenC2 Consumers that receive the 'update file' Command:

- upon successful parsing and initiating the processing of the 'update file' Command, OpenC2 Consumers MAY respond with Response status code 102.
- upon completion of all the steps necessary to complete the update and the Actuator commences operations functioning with the new file, OpenC2 Consumers SHOULD respond with Response status code 200.
- but cannot parse or process the 'update file' Command:
  - MUST NOT respond with the 200 status code.
  - SHOULD respond with the 400 status code.
  - MAY respond with the 500 status code.
- but do not support the 'update file' Command:
  - MUST NOT respond with the 200 status code.
  - SHOULD respond with the 501 status code.
  - SHOULD respond with 'Command not supported' in the status text.
  - MAY respond with the 500 status code.
- but cannot access the file specified in the file Target:
  - MUST respond with the 500 status code.
  - SHOULD respond with 'Cannot access file' in the status text.

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.

3.1 Clauses Pertaining to the OpenC2 Producer Conformance Target

All OpenC2 Producers that are conformant to this specification **MUST** satisfy Conformance Clause 1 and **MAY** satisfy one or more of Conformance Clauses 2 through 18.

3.1.1 Conformance Clause 1: Baseline OpenC2 Producer

To satisfy Baseline OpenC2 Producer conformance an OpenC2 Producer:

- 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 **MUST** implement all serializations 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.
- 3.1.1.5 **MUST** be conformant with Version 1.0 of the OpenC2 Language Specification.
- 3.1.1.6 **MUST** implement the 'query features' Command in accordance with the normative 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
  - Conformance Clause 6
  - Conformance Clause 7
  - Conformance Clause 8

3.1.2 Conformance Clause 2: IP Version 4 Connection Producer

To satisfy 'IP Version 4 Connection Producer' conformance an OpenC2 Producer:

- 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

To satisfy 'IP Version 6 Connection Producer' conformance an OpenC2 Producer:

- 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

To satisfy 'IP Version 4 Net Producer' conformance an OpenC2 Producer:

- 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

To satisfy 'IP Version 6 Net Producer' conformance an OpenC2 Producer:

- 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: Domain Name Producer

To satisfy 'Domain Name Producer' conformance an OpenC2 Producer:

- 3.1.6.1 **MUST** meet all of conformance criteria identified in Conformance Clause 1 of this specification.
- 3.1.6.2 **MUST** implement the 'allow domain_name' Command in accordance with Section 2.3.1 of this specification.
- 3.1.6.3 **MUST** implement the 'deny domain_name' Command in accordance with Section 2.3.2 of this specification.

3.1.7 Conformance Clause 7: Advanced Connection Producer

To satisfy 'Advanced Connection Producer' conformance an OpenC2 Producer:

- 3.1.7.1 **MUST** meet all of conformance criteria identified in Conformance Clause 1 of this specification.
- 3.1.7.2 **MUST** implement the 'allow pf:advanced_connection' Command in accordance with Section 2.3.1 of this specification.
- 3.1.7.3 **MUST** implement the 'deny pf:advanced_connection' Command in accordance with Section 2.3.2 of this specification.

3.1.8 Conformance Clause 8: Update File Producer

To satisfy 'Update File Producer' conformance an OpenC2 Producer:

- 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 'update file' Command in accordance with Section 2.3.5.1 of this specification.

3.1.9 Conformance Clause 9: Delete Rule Number Producer

To satisfy 'Delete Rule Number Producer' conformance an OpenC2 Producer:

- 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 'delete pf:rule_number' in accordance with Section 2.3.4.1 of this specification.

3.1.10 Conformance Clause 10: Query Rule Number Producer

To satisfy 'Query Rule Number Producer' conformance an OpenC2 Producer:

- 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 'query pf:rule_number' in accordance with Section 2.3.3.2 of this specification.

3.1.11 Conformance Clause 11: Persistent Producer

To satisfy 'Persistence Producer' conformance an OpenC2 Producer:

- 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 'persistent' 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.

3.1.12 Conformance Clause 12: Direction Producer

To satisfy 'Direction Producer' conformance an OpenC2 Producer:

- 3.1.12.1 **MUST** meet all of the conformance criteria identified in Conformance Clause 1 of this specification.
- 3.1.12.2 **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.

3.1.13 Conformance Clause 13: Drop Process Producer

To satisfy 'Drop Process Producer' conformance an OpenC2 Producer:

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

- 3.1.13.2 **MUST** implement the 'drop_process' Command Argument as a valid option for any Command associated with the 'deny' Action in accordance with Section 2.3.2 of this specification.

3.1.14 Conformance Clause 14: Temporal Producer

To satisfy 'Temporal Producer' conformance an OpenC2 Producer:

- 3.1.14.1 **MUST** meet all of the conformance criteria identified in Conformance Clause 1 of this specification.
- 3.1.14.2 **MUST** implement the 'start_time', 'stop_time', and 'duration' Command Arguments as valid options 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.

3.1.15 Conformance Clause 15: Logging Producer

To satisfy 'Logging Producer' conformance an OpenC2 Producer:

- 3.1.15.1 **MUST** meet all of the conformance criteria identified in Conformance Clause 1 of this specification.
- 3.1.15.2 **MUST** implement the 'logged' 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.

3.1.16 Conformance Clause 16: Stateful Producer

To satisfy 'Stateful Producer' conformance an OpenC2 Producer:

- 3.1.16.1 **MUST** meet all of the conformance criteria identified in Conformance Clause 1 of this specification.
- 3.1.16.2 **MUST** implement the 'stateful' 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.

3.1.17 Conformance Clause 17: Priority Producer

To satisfy 'Priority Producer' conformance an OpenC2 Producer:

- 3.1.17.1 **MUST** meet all of the conformance criteria identified in Conformance Clause 1 of this specification.
- 3.1.17.2 **MUST** implement the 'priority' 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.

3.1.18 Conformance Clause 18: Insert Rule Producer

To satisfy 'Insert Rule Number Producer' conformance an OpenC2 Producer:

- 3.1.18.1 **MUST** meet all of the conformance criteria identified in Conformance Clause 1 of this specification.
- 3.1.18.2 **MUST** implement the 'insert_rule' 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 Clauses Pertaining to the OpenC2 Consumer Conformance Target

All OpenC2 Consumers that are conformant to this specification **MUST** satisfy Conformance Clause 19 and **MAY** satisfy one or more of Conformance Clauses 20 through 36.

3.2.1 Conformance Clause 19: Baseline OpenC2 Consumer

To satisfy Baseline OpenC2 Consumer conformance an OpenC2 Consumer:

- 3.2.1.1 **MUST** support JSON serialization of OpenC2 Commands that are syntactically valid in accordance with the property tables presented in Section 2.1.
- 3.2.1.2 **MUST** implement all serializations 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.
- 3.2.1.3 **MUST** support the use of a Transfer Specification that is capable of delivering authenticated, ordered, lossless and uniquely identified OpenC2 messages.
- 3.2.1.4 **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.
- 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 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_requested' 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_requested' argument **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 20
    - Conformance Clause 21
    - Conformance Clause 22
    - Conformance Clause 23
    - Conformance Clause 24
    - Conformance Clause 25
    - Conformance Clause 26

3.2.2 Conformance Clause 20: IP Version 4 Connection Consumer

To satisfy 'IP Version 4 Connection Consumer' conformance an OpenC2 Consumer:

- 3.2.2.1 **MUST** meet all of conformance criteria identified in Conformance Clause 19 of this specification.
- 3.2.2.2 **MUST** implement the 'allow ipv4_connection' Command in accordance with Section 2.3.1 of this specification.
- 3.2.2.3 **MUST** implement the 'deny ipv4_connection' Command in accordance with Section 2.3.2 of this specification.

3.2.3 Conformance Clause 21: IP Version 6 Connection Consumer

To satisfy 'IP Version 6 Connection Consumer' conformance an OpenC2 Consumer:

- 3.2.3.1 **MUST** meet all of conformance criteria identified in Conformance Clause 19 of this specification.
- 3.2.3.2 **MUST** implement the 'allow ipv6_connection' Command in accordance with Section 2.3.1 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 22: IP Version 4 Net Consumer

To satisfy 'IP Version 4 Net Consumer' conformance an OpenC2 Consumer:

- 3.2.4.1 **MUST** meet all of conformance criteria identified in Conformance Clause 19 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 23: IP Version 6 Net Consumer

To satisfy 'IP Version 6 Net Consumer' conformance an OpenC2 Consumer:

- 3.2.5.1 **MUST** meet all of conformance criteria identified in Conformance Clause 19 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 24: Domain Name Consumer

To satisfy 'Domain Name Consumer' conformance an OpenC2 Consumer:

- 3.2.6.1 **MUST** meet all of conformance criteria identified in Conformance Clause 19 of this specification.
- 3.2.6.2 **MUST** implement the 'allow domain_name' Command in accordance with Section 2.3.1 of this specification.
- 3.2.6.3 **MUST** implement the 'deny domain_name' Command in accordance with Section 2.3.2 of this specification.

3.2.7 Conformance Clause 25: Advanced Connection Consumer

To satisfy 'Advanced Connection Consumer' conformance an OpenC2 Consumer:

- 3.2.7.1 **MUST** meet all of conformance criteria identified in Conformance Clause 19 of this specification.
- 3.2.7.2 **MUST** implement the 'allow pf:advanced_connection' Command in accordance with Section 2.3.1 of this specification.
- 3.2.7.3 **MUST** implement the 'deny pf:advanced_connection' Command in accordance with Section 2.3.2 of this specification.

3.2.8 Conformance Clause 26: Update File Consumer

To satisfy 'Update File Consumer' conformance an OpenC2 Consumer:

- 3.2.8.1 **MUST** meet all of the conformance criteria identified in Conformance Clause 19 of this specification.
- 3.2.8.2 **MUST** implement the 'update file' Command in accordance with Section 2.3.5.1 of this specification.

3.2.9 Conformance Clause 27: Delete Rule Number Consumer

To satisfy 'Delete Rule Number Consumer' conformance an OpenC2 Consumer:

- 3.2.9.1 **MUST** meet all of the conformance criteria identified in Conformance Clause 19 of this specification.
- 3.2.9.2 **MUST** implement the 'delete pf:rule_number' in accordance with Section 2.3.4.1 of this specification.

3.2.10 Conformance Clause 28: Query Rule Number Consumer

To satisfy 'Query Rule Number Consumer' conformance an OpenC2 Consumer:

- 3.2.10.1 **MUST** meet all of the conformance criteria identified in Conformance Clause 19 of this specification.
- 3.2.10.2 **MUST** implement the 'query pf:rule_number' in accordance with Section 2.3.3.2 of this specification.

3.2.11 Conformance Clause 29: Persistent Consumer

To satisfy 'Persistent Consumer' conformance an OpenC2 Consumer:

- 3.2.11.1 **MUST** meet all of the conformance criteria identified in Conformance Clause 19 of this specification.
- 3.2.11.2 **MUST** implement the 'persistent' 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.

3.2.12 Conformance Clause 30: Direction Consumer

To satisfy 'Direction Consumer' conformance an OpenC2 Consumer:

- 3.2.12.1 **MUST** meet all of the conformance criteria identified in Conformance Clause 19 of this specification.
- 3.2.12.2 **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.

3.2.13 Conformance Clause 31: Drop Process Consumer

To satisfy 'Drop Process Consumer' conformance an OpenC2 Consumer:

- 3.2.13.1 **MUST** meet all of the conformance criteria identified in Conformance Clause 19 of this specification.
- 3.2.13.2 **MUST** implement the 'drop_process' Command Argument as a valid option for any Command associated with the 'deny' Action in accordance with Section 2.3.2 of this specification of this specification.

3.2.14 Conformance Clause 32: Temporal Consumer

To satisfy 'Temporal Consumer' conformance an OpenC2 Consumer:

- 3.2.14.1 **MUST** meet all of the conformance criteria identified in Conformance Clause 19 of this specification.
- 3.2.14.2 **MUST** implement the 'start_time', 'stop_time', and 'duration' Command Arguments as valid options 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.

3.2.15 Conformance Clause 33: Logging Consumer

To satisfy 'Logging Consumer' conformance an OpenC2 Consumer:

- 3.2.15.1 **MUST** meet all of the conformance criteria identified in Conformance Clause 19 of this specification.
- 3.2.15.2 **MUST** implement the 'logged' 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.

3.2.16 Conformance Clause 34: Stateful Consumer

To satisfy 'Stateful Consumer' conformance an OpenC2 Consumer:

- 3.2.16.1 **MUST** meet all of the conformance criteria identified in Conformance Clause 19 of this specification.

- 3.2.16.2 **MUST** implement the 'stateful' 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.

3.2.17 Conformance Clause 35: Priority Consumer

To satisfy 'Priority Consumer' conformance an OpenC2 Consumer:

- 3.2.17.1 **MUST** meet all of the conformance criteria identified in Conformance Clause 19 of this specification.
- 3.2.17.2 **MUST** implement the 'priority' 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.

3.2.18 Conformance Clause 36: Insert Rule Consumer

To satisfy 'Insert Rule Consumer' conformance an OpenC2 Consumer:

- 3.2.18.1 **MUST** meet all of the conformance criteria identified in Conformance Clause 19 of this specification.
- 3.2.18.2 **MUST** implement the 'insert_rule' 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.

Appendix A. References

This appendix contains the normative and informative references that are used in this document.

While any hyperlinks included in this appendix were valid at the time of publication, OASIS cannot guarantee their long-term validity.

A.1 Normative References

The following documents are referenced in such a way that some or all of their content constitutes requirements of this document.

[JADN-v1.0]

JSON Abstract Data Notation Version 1.0. Edited by David Kemp. 17 August 2021.
OASIS Committee Specification 01. https://docs.oasis-open.org/openc2/jadn/v1.0/cs01/jadn-v1.0-cs01.html.
Latest stage: https://docs.oasis-open.org/openc2/jadn/v1.0/jadn-v1.0.html.

[RFC1034]

Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, https://www.rfc-editor.org/info/rfc1034.

[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.

[RFC2780]

Bradner, S. and V. Paxson, "IANA Allocation Guidelines For Values In the Internet Protocol and Related Headers", BCP 37, RFC 2780, DOI 10.17487/RFC2780, March 2000, https://www.rfc-editor.org/info/rfc2780.

[RFC4443]

Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, https://www.rfc-editor.org/info/rfc4443.

[RFC6335]

Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, https://www.rfc-editor.org/info/rfc6335.

[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, https://www.rfc-editor.org/info/rfc8259.

[OpenC2-Lang-v1.0]

Open Command and Control (OpenC2) Language Specification Version 1.0. Edited by Jason Romano and Duncan Sparrell. 24 November 2019. OASIS Committee Specification 02. https://docs.oasis-open.org/openc2/oc2ls/v1.0/cs02/oc2ls-v1.0-cs02.html. Latest stage: https://docs.oasis-open.org/openc2/oc2ls/v1.0/oc2ls-v1.0.html.

[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. 11 July 2019. Committee Specification 01. https://docs.oasis-open.org/openc2/oc2slpf/v1.0/cs01/oc2slpf-v1.0-cs01.html. Latest version: https://docs.oasis-open.org/openc2/oc2slpf/v1.0/oc2slpf-v1.0.html.

A.2 Informative 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-v1.1]

Specification for Transfer of OpenC2 Messages via HTTPS Version 1.1. Edited by David Lemire. 30 November 2021. OASIS Committee Specification 01. https://docs.oasis-open.org/openc2/open-impl-https/v1.1/cs01/open-impl-https-v1.1-cs01.html. Latest stage: https://docs.oasis-open.org/openc2/open-impl-https/v1.1/open-impl-https-v1.1.html.

[OpenC2-MQTT-v1.0]

Specification for Transfer of OpenC2 Messages via MQTT Version 1.0. Edited by David Lemire. 19 November 2021. OASIS Committee Specification 01. https://docs.oasis-open.org/openc2/transf-mqtt/v1.0/cs01/transf-mqtt-v1.0-cs01.html. Latest stage: https://docs.oasis-open.org/openc2/transf-mqtt/v1.0/transf-mqtt-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, https://www.semanticscholar.org/paper/Active-Cyber-Defense-%3A-A-Vision-for-Real-Time-Cyber-Herring-Willett/7c128468ae42584f282578b86439dbe9e8c904a8.

[IACD]

Willett, Keith D., "Integrated Adaptive Cyberspace Defense: Secure Orchestration", International Command and Control Research and Technology Symposium, June 2015, https://www.semanticscholar.org/paper/Integrated-Adaptive-Cyberspace-Defense-%3A-Secure-by-Willett/a22881b8a046e7eab11acf647d530c2a3b03b762.

[OPENC2-M.B-2020]

Mavroeidis, V., & Brule, J. A Nonproprietary Language for the Command and Control of Cyber Defenses – OpenC2, Computers & Security, vol. 97, 101999, October 2020, https://doi.org/10.1016/j.cose.2020.101999.

Appendix B. Safety, Security and Privacy Considerations

OpenC2, as a cyber defense automation tool, is high-value target for adversaries attempting to exploit an environment where it is used. Appendix B of the OpenC2 Architecture Specification [OpenC2-Arch-v1.0] discusses:

- Threats to OpenC2
- Applying security services to OpenC2 operations
- Network topology considerations for OpenC2 messages

Refer to that document for a review of these topics in the context of OpenC2.

Appendix C. Acknowledgments

C.1 Special Thanks

Substantial contributions to this document from the following individuals are gratefully acknowledged:

First Name	Last Name	Company
David	Lemire	National Security Agency
Vasileios	Mavroeidis	University of Oslo
Duncan	Sparrell	sFractal Consulting LLC

C.2 Participants

The following individuals were members of the OASIS OpenC2 Technical Committee during the creation of this specification and their contributions are gratefully acknowledged:

First Name	Last Name	Company
Stephen	Banghart	NIST
Michelle	Barry	AT&T
David	Bizeul	SEKOIA
Jason	Callaway	Google Inc.
Marco	Caselli	Siemens AG
Toby	Considine	University of North Carolina at Chapel Hill
Sudeep	Das	McAfee
Alexandre	Dulaunoy	CIRCL
Blake	Essing	AT&T
Martin	Evandt	University of Oslo
Alex	Everett	University of North Carolina at Chapel Hill
Jessica	Fitzgerald-McKay	National Security Agency
Jane	Ginn	Cyber Threat Intelligence Network, Inc. (CTIN)
David	Girard	Trend Micro
Zachary	Gorak	National Security Agency
John-Mark	Gurney	Copado
Daichi	Hasumi	NEC Corporation
Stephanie	Hazlewood	IBM
Shoko	Honda	Trend Micro
Tim	Hudson	Cryptsoft Pty Ltd.
Christian	Hunt	Copado
Andras	Iklody	CIRCL
Sridhar	Jayanthi	EclecticIQ
Ryan	Joyce	DarkLight, Inc.
Takahiro	Kakumaru	NEC Corporation
David	Kemp	National Security Agency
Lauri	Korts-Pärn	NEC Corporation
Kent	Landfield	McAfee
Cheolho	Lee	NSR
David	Lemire	National Security Agency
Anthony	Librera	AT&T
Jason	Liu Northrop	Grumman
Terry	MacDonald	Individual
Radu	Marian	Bank of America
Patrick	Maroney	AT&T
Vasileios	Mavroeidis	University of Oslo
Paul	Patrick	DarkLight, Inc.
Andrew	Pendergast	ThreatConnect, Inc.
Alexandre	Cabrol Perales	Sopra Steria Group

First Name	Last Name	Company
Wende	Peters	Bank of America
Dmitry	Raidman	Cybeats
Chris	Ricard	Financial Services Information Sharing and Analysis Center (FS-ISAC)
Daniel	Riedel	Copado
Christopher	Robinson	Cyber Threat Intelligence Network, Inc. (CTIN)
Michael	Rosa	National Security Agency
Omar	Santos	Cisco Systems
Aleksandra	Scalco	US Department of Defense (DoD)
Randall	Sharo	US Department of Defense (DoD)
Duane	Skeen	Northrop Grumman
Calvin	Smith	Northrop Grumman
Dan	Solero	AT&T
Ben	Sooter	Electric Power Research Institute (EPRI)
Duncan	Sparrell	sFractal Consulting LLC
Michael	Stair	AT&T
Andrew	Storms	Copado
Gerald	Stueve	Fornetix
Takayuki	Tachihara	Trend Micro
Bill	Trost	AT&T
Drew	Varner	NineFX, Inc.
Jyoti	Verma	Cisco Systems
David	Waltermire	NIST
Russ	Warren	IBM
Sean	Welsh	AT&T
Charles	White	Fornetix
Sam	Taghavi Zargar	Cisco Systems

Appendix D. Revision History

Revision	Date	Editors	Changes Made
01	2021-05-03	Alex Everett and Vasileios Mavroeidis	Populated Initial working draft.
02	--	Alex Everett and Vasileios Mavroeidis	Multiple editorial, style, and grammar fixes.
03	2022-03-15	Alex Everett and Vasileios Mavroeidis	Multiple editorial, style, and grammar fixes. Major update of the conformance section.

Appendix E. Sample Commands

This section is non-normative.

This section will summarize and provide examples of OpenC2 Commands as they pertain to packet filters. The sample Commands will be encoded in verbose JSON, however other encodings are possible provided the Command is validated against the property tables defined in Section 2.1 of this specification. Examples of corresponding 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 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`

=======

E.1 Deny and Allow

Deny and allow can be treated as mathematical complements of each other. Unless otherwise stated, the example Targets, Specifiers, Arguments and corresponding Responses are applicable to both Actions.

E.1.1 Deny a particular connection

Block a particular connection within the domain and do not send a host unreachable. Note that the drop_process argument does not apply to the allow Action.

Command:

```
{
    "action":"deny",
    "target":{
        "ipv4_connection":{
            "protocol":"tcp",
            "src_addr":"1.2.3.4",
            "src_port":10996,
            "dst_addr":"198.2.3.4",
            "dst_port":80
        }
    },
    "args":{
        "start_time":1534775460000,
        "duration":500,
        "response_requested":"ack",
        "pf":{
            "drop_process":"none"
        }
    },
    "actuator":{
        "pf":{
            "asset_id":"30"
        }
    }
}
```

Response:

```
{
    "status":102
}
```

E.1.2 Deny all outbound ftp transfers

Block all outbound ftp data transfers, send false acknowledgment. Note that the five-tuple is incomplete. Note that the response_requested field was not populated therefore will be 'complete'. Also note that the Actuator called out was PF with no additional Specifiers, therefore all endpoints that can execute the Command should. Note that the drop_process argument does not apply to the allow Action.

Command:

```
{
    "action":"deny",
    "target":{
        "ipv4_connection":{
            "protocol":"tcp",
            "src_port":21
        }
    },
    "args":{
        "pf":{
            "drop_process":"false_ack",
            "direction":"egress"
        }
    },
    "actuator":{
        "pf":{

        }
    }
}
```

Responses:

Case 1: the Actuator successfully issued the deny.

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

Case 2: the Command failed due to a syntax error in the Command. Optional status text is 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 3: the Command failed because an Argument was not supported.

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

E.1.3 Block all inbound traffic from a particular source

Block all inbound traffic from the specified ipv6 network and do not send any response back to the producer. In this case the ipv6_net Target and the direction argument was used. In this case only the perimeter filters should update the rule.

Command:

```
{
    "action":"deny",
    "target":{
        "ipv6_net":"3ffe:1900:4545:3:200:f8ff:fe21:67cf"
    },
    "args":{
        "response_requested":"none",
        "pf":{
            "direction":"ingress"
        }
    },
    "actuator":{
        "pf":{
            "named_group":"perimeter"
        }
    }
}
```

E.1.4 Statefully permit ftp transfers to a particular destination

Permit ftp data transfers to 3ffe:1900:4545:3::f8ff:fe21:67cf from any initiating source and allow needed return traffic. (Note that an actual application may also need to allow ftp-data (port 20) in order for transfers to be permitted depending on the ftp connection type and the firewall technology).

Command:

```
{
    "action":"allow",
    "target":{
        "ipv6_connection":{
            "protocol":"tcp",
            "dst_addr":"3ffe:1900:4545:3::f8ff:fe21:67cf",
            "src_port":21
        }
    },
    "args":{
        "pf":{
            "stateful":true
        }
    },
    "actuator":{
        "pf":{

        }
    }
}
```

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

Response:

```
{
    "status":200,
    "results":{
        "pf":{
            "rule_number":1234
        }
    }
}
```

E.1.5 Deny outbound Network Time Protocol (NTP)

From a tagged set of webservers in the default virtual network, traffic requests from these servers to timekeeping services will be denied.

Command:

```
{
    "action":"deny",
    "target":{
        "pf":{
            "advanced_connection":{
                "src_addr":"webservers",
                "network":"default",
                "protocol":"udp",
                "dst_port":123
            }
        }
    },
    "args":{
        "pf":{
            "direction":"egress",
            "priority":500
        }
    },
    "actuator":{
        "pf":{

        }
    }
}
```

E.2 Delete Rule

Used to remove a firewall rule rather than issue an allow or deny to counteract the effect of an existing rule. In this case the rule number assigned by the actuator in a previous allow will be removed (refer to the example E.1.4.

Command:

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

        }
    }
}
```

E.3 Update file

Update is intended for the device to process new configuration files. The update Action is a compound Action in that all of the steps required for a successful update (such as download the new file, install the file, reboot etc.) are implied. File is the only valid Target type for Update.

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

Command:

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

Responses:

Case 1: successful update of the configuration.

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

Case 2: this Actuator does not support the update file Command.

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

Case 3: this Actuator could not access the file.

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

E.4 Query features

The 'query features' Command is intended to enable the Openc2 Producer to determine the capabilities of the Actuator. The 'query features' Command can also be used to check the status of the Actuator.

E.4.1 No query items set

This Command uses 'query features' with no query items to verify that the Actuator is functioning.

Command:

```
{
    "action":"query",
    "target":{
        "features":[

        ]
    }
}
```

Response:

The Actuator is alive.

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

E.4.2 Version of Language specification supported

This Command queries the 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 Actuator needs only to report the highest supported minor version.

Command:

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

Response:

The Actuator supports Language Specification Version 1.0.

```
{
    "status":200,
    "results":{
        "versions":[
            "1.0"
        ]
    }
}
```

E.4.3 Actuator profiles supported

This Command queries the Actuator to determine both the language versions and the profiles supported.

Command:

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

Response:

The Actuator device is apparently a smart front-door-lock for which an extension profile has been written. The device supports both the standard pf functions and whatever Commands are defined in the extension profile.

```
{
    "status":200,
    "results":{
        "versions":[
            "1.3"
        ],
        "profiles":[
            "pf",
            "iot-front-door-lock"
        ]
    }
}
```

E.4.4 Specific Commands Supported

This Command queries the Actuator to determine which Action/Target pairs are supported. Not all Targets are meaningful in the context of a specific Action, and although a Command such as "update ipv4_connection" may be syntactically valid, the combination does not specify an operation supported by the Actuator profile.

Command:

For each supported Action list the Targets supported by this Actuator.

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

Response:

The Actuator supports the following Action/Target pairs.

```
{
    "status":200,
    "results":{
        "pairs":{
            "allow":[
                "ipv6_net",
                "ipv6_connection"
            ],
            "deny":[
                "ipv6_net",
                "ipv6_connection"
            ],
            "query":[
                "features"
            ],
            "delete":[
                "pf:rule_number"
            ],
            "update":[
                "file"
            ]
        }
    }
}
```

E.4.5 Rule Details

This Command queries the Actuator to determine the Action, Target, and Argument values for a particular rule.

Command:

For each supported Action list the Targets supported by this Actuator.

```
{
    "action":"query",
    "target":{
        "pf:rule_number":20
    },
    "actuator":{
        "pf":{
            "asset_id":"30"
        }
    }
}
```

Response:

The Actuator returns information that could be used to reconstruct the rule.


```
{
    "status":200,
    "results":{
        "pf":{
            "rule_number":20,
            "action":"deny",
            "ipv4_connection":{
                "protocol":"tcp",
                "src_addr":"1.2.3.4",
                "src_port":10996,
                "dst_addr":"198.2.3.4",
                "dst_port":80
            },
            "args":{
                "drop_process":"false_ack",
                "direction":"egress"
            }
        }
    }
}
```

Appendix F. Notices

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

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

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

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

As stated in the OASIS IPR Policy, the following three paragraphs in brackets apply to OASIS Standards Final Deliverable documents (Committee Specification, Candidate OASIS Standard, OASIS Standard, or Approved Errata).

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by i

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that w

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain

The name "OASIS" is a trademark of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see https://www.oasis-open.org/policies-guidelines/trademark for above guidance.