



Open Command and Control (OpenC2) Language Specification Version 2.0

Committee Specification Draft 02

15 May 2024

This stage:

<https://docs.oasis-open.org/openc2/oc2ls/v2.0/csd02/oc2ls-v2.0-csd02.md> (Authoritative)
<https://docs.oasis-open.org/openc2/oc2ls/v2.0/csd02/oc2ls-v2.0-csd02.html>
<https://docs.oasis-open.org/openc2/oc2ls/v2.0/csd02/oc2ls-v2.0-csd02.pdf>

Previous stage:

<https://docs.oasis-open.org/openc2/oc2ls/v2.0/csd01/oc2ls-v2.0-csd01.md> (Authoritative)
<https://docs.oasis-open.org/openc2/oc2ls/v2.0/csd01/oc2ls-v2.0-csd01.html>
<https://docs.oasis-open.org/openc2/oc2ls/v2.0/csd01/oc2ls-v2.0-csd01.pdf>

Latest stage:

<https://docs.oasis-open.org/openc2/oc2ls/v2.0/oc2ls-v2.0.md> (Authoritative)
<https://docs.oasis-open.org/openc2/oc2ls/v2.0/oc2ls-v2.0.html>
<https://docs.oasis-open.org/openc2/oc2ls/v2.0/oc2ls-v2.0.pdf>

Technical Committee:

[OASIS Open Command and Control \(OpenC2\) TC](#)

Chairs:

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

Editors:

Duncan Sparrell (duncan@sfractal.com), [sFractal Consulting LLC](#)
Toby Considine (toby.considine@unc.edu), [University of North Carolina at Chapel Hill](#)
David Lemire (david.lemire@hii-tsd.com), [National Security Agency](#)

Abstract:

Cyber attacks are increasingly sophisticated, less expensive to execute, dynamic and automated. The provision of cyber defense via statically configured products operating in isolation is untenable. Standardized interfaces, protocols and data models will facilitate the integration of the functional blocks within a system and between systems. 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/or systems in a manner that is agnostic of the underlying products, technologies, transport mechanisms or other aspects of the implementation. It should be understood that a language such as OpenC2 is necessary but insufficient to enable coordinated cyber responses that occur within cyber relevant time. Other aspects of coordinated cyber response such as sensing, analytics, and selecting appropriate courses of action are beyond the scope of OpenC2.

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://groups.oasis-open.org/communities/tc-community-home2?CommunityKey=a34c9baf-48b2-44c5-a567-018dc7d32296>

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, technical-committee-comments@oasis-open.org

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-Lang-v2.0]

Open Command and Control (OpenC2) Language Specification Version 2.0. Edited by Duncan Sparrell, Toby Considine, and David Lemire. 15 May 2024. OASIS Committee Specification Draft 02. <https://docs.oasis-open.org/openc2/oc2ls/v2.0/csd02/oc2ls-v2.0-csd02.html>. Latest stage: <https://docs.oasis-open.org/openc2/oc2ls/v2.0/oc2ls-v2.0.html>.

Notices

Copyright © OASIS Open 2024. All Rights Reserved.

Distributed under the terms of the OASIS [IPR Policy](#).

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.

For complete copyright information please see the full Notices section in an Appendix below.

Table of Contents

[1 Introduction](#)

[1.1 Changes From Earlier Versions](#)

[1.2 Glossary](#)

[1.2.1 Definitions of Terms](#)

[1.2.2 Acronyms and abbreviations](#)

[1.2.3 Document Conventions](#)

[1.2.3.1 Naming Conventions](#)

[1.2.3.2 Font Colors and Style](#)

[1.4 Purpose and Scope](#)

[2 OpenC2 Language Description](#)

[2.1 OpenC2 Command](#)

[2.2 OpenC2 Response](#)

[3 OpenC2 Language Definition](#)

[3.1 Base Components and Structures](#)

[3.1.1 Data Types](#)

[3.1.2 Semantic Value Constraints](#)

[3.1.3 Multiplicity](#)

[3.1.4 Extensions](#)

[3.1.5 Serialization](#)

[3.2 Message](#)

[3.3 Content](#)

[3.3.1 OpenC2 Command](#)

[3.3.1.1 Action](#)

[3.3.1.2 Target](#)

[3.3.1.3 Profile](#)

[3.3.1.4 Command Arguments](#)

[3.3.2 OpenC2 Response](#)

[3.3.2.1 Response Status Code](#)

[3.3.2.2 Response Results](#)

[3.3.3 OpenC2 Event](#)

[3.3.4 Message Signatures](#)

[3.4 Type Definitions](#)

[3.4.1 Target Types](#)

[3.4.1.1 Artifact](#)

[3.4.1.2 Device](#)

[3.4.1.3 Domain Name](#)

[3.4.1.4 Email Address](#)

[3.4.1.5 Features](#)

[3.4.1.6 File](#)

[3.4.1.7 Internationalized Domain Name](#)

[3.4.1.8 Internationalized Email Address](#)

[3.4.1.9 IPv4 Address Range](#)

[3.4.1.10 IPv4 Connection](#)

[3.4.1.11 IPv6 Address Range](#)

[3.4.1.12 IPv6 Connection](#)

[3.4.1.13 IRI](#)

[3.4.1.14 MAC Address](#)

[3.4.1.15 Process](#)

[3.4.1.16 URI](#)

[3.4.2 Data Types](#)

[3.4.2.1 Action-Targets](#)

[3.4.2.2 Command-ID](#)

[3.4.2.3 Date-Time](#)

[3.4.2.4 Duration](#)

[3.4.2.5 Feature](#)

[3.4.2.6 Hashes](#)

[3.4.2.7 Hostname](#)

[3.4.2.8 Internationalized Hostname](#)

[3.4.2.9 IPv4 Address](#)

[3.4.2.10 IPv6 Address](#)

[3.4.2.11 L4 Protocol](#)

[3.4.2.12 Message-Type](#)

[3.4.2.13 Namespace Identifier](#)

[3.4.2.14 Payload](#)

[3.4.2.15 Port](#)

[3.4.2.16 Response-Type](#)

[3.4.2.17 Targets](#)

[3.4.2.18 Version](#)

[4 Mandatory Commands/Responses](#)

[4.1 Implementation of 'query features' Command](#)

[4.2 Examples of 'query features' Commands and Responses](#)

[4.2.1 Example 1: Bare "query : features"](#)

[4.2.2 Example 2: Query for Specific Features](#)

[4.2.3 Example 3: Query Features Detailed Response with Multiple Profiles](#)

[5 Conformance](#)

[5.1 Conformance Clause 1: Command](#)

[5.2 Conformance Clause 2: Response](#)

[5.3 Conformance Clause 3: Producer](#)

[5.4 Conformance Clause 4: Consumer](#)

[Appendix A. References](#)

[A.1 Normative References](#)

[A.2 Informative References](#)

[Appendix B. Safety, Security and Privacy Considerations](#)

[Appendix C. Acknowledgments](#)

[C.1 Special Thanks](#)

[C.2 Participants](#)

[Appendix D. Revision History](#)

[Appendix E. Examples](#)

[E.1 Example 1: Device Quarantine](#)

[E.2 Example 2: Block Connection](#)

[Appendix F. Notices](#)

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 *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 can respond with status and any requested information. An overview of the concepts that underlie OpenC2 and the structure of the suite of specifications can be found in the OpenC2 Architecture Specification [[OpenC2-Arch-v1.0](#)].

The goal of OpenC2 is to provide a language for interoperating between functional elements of cyber defense systems. This language, used in conjunction with OpenC2 Actuator Profiles (e.g., the [[Stateless Packet Filtering AP](#)]) and OpenC2 Transfer Specifications (e.g., for [[MQTT](#)] and [[HTTPS](#)]), allows for vendor-agnostic cybertime response to attacks.

This **OpenC2 Language Specification** provides the semantics for the essential elements of the language, the structure for Commands and Responses, and the schema that defines the proper syntax for the language elements that represents the Command or Response. It also describes the mechanisms for extending the capabilities of the language.

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 non-centralized manner, allowing standard and non-standard capabilities to be defined with semantic and syntactic rigor.

1.1 Changes From Earlier Versions

To Be Supplied.

1.2 Glossary

1.2.1 Definitions of Terms

This section is normative.

- **Action:** The task or activity to be performed (e.g., 'deny').
- **Actuator:** The Consumer that executes a Command.
- **Actuator Profile (AP):** A defined subset of the OpenC2 language (i.e., actions, targets, command arguments, results) plus any extensions required to specify the use of OpenC2 to command a particular function. Actuator Profiles are defined in JADN schemas associated with published actuator profile specification documents.

- **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. A Consumer will implement one or more Actuator Profiles.
- **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 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

Acronym	Description
API	Application Programming Interface
AP	Actuator Profile
BCP	Best Current Practice
CACAO	Collaborative Automated Course of Action Operations
CBOR	Concise Binary Object Representation
CIDR	Classless Inter-Domain Routing
DOI	Digital Object Identifier
EUI	Extended Unique Identifier
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
IACD	Integrated Adaptive Cyber Defense

Acronym	Description
IANA	Internet Assigned Numbers Authority
ICMP	Internet Control Message Protocol
ID	Identifier
IP	Internet Protocol
IPR	Intellectual Property Rights
ITU-T	International Telecommunications Union - Telecommunications
JADN	JSON Abstract Data Notation
JSON	JavaScript Object Notation
JSS	JSON Signature Scheme
MAC	Media Access Control
MD5	Message Digest
MQTT	Message Queuing Telemetry Transfer
OASIS	Organization for the Advancement of Structured Information Standards
OpenC2	Open Command and Control
RFC	Request for Comment
SCTP	Stream Control Transmission Protocol
SHA	Secure Hash Algorithm
SLPF	StateLess Packet Filtering
STD	Standard
STIX	Structured Threat Information Expression
TC	Technical Committee
TCP	Transmission Control Protocol
UDP	User Datagram Control Protocol

Acronym	Description
UML	Unified Modeling Language
URI	Uniform Resource Identifier
UTC	Coordinated Universal Time
UUID	Universally Unique IDentifier
XML	eXtensibel Markup Language

1.2.3 Document Conventions

1.2.3.1 Naming Conventions

The OpenC2 language is formally defined by a schema written in JSON Abstract Data Notation (JADN) [[JADN-v1.0](#)]. The naming conventions used in this specification are consistent with JADN's default conventions.

- 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 type names begin with an uppercase character.
- Property names and type names are between 1 and 32 characters long.
- Words in property names are separated with an underscore (`_`), while words in type names are separated with a hyphen (`-`).
- "Underscore" refers to Unicode "low line", U+005F; "hyphen" refers to Unicode "hyphen-minus", U+002D.

1.2.3.2 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"
      }
    }
  }
}
```

1.4 Purpose and Scope

This OpenC2 Language Specification defines the set of components to assemble a complete command and control Message and provides a framework so that the language can be extended. To achieve this purpose, the scope of this specification includes:

1. the set of Actions and options that may be used in Commands;
2. the set of Targets and Target Specifiers;
3. a syntax that defines the structure of Commands and Responses;
4. a JSON serialization of Commands and Responses;
5. the procedures for extending the language.

The OpenC2 language assumes that the event has been detected, a decision to act has been made, the act is warranted, and the initiator and recipient of the Commands are authenticated and authorized. The OpenC2 language was designed to be agnostic of the other aspects of cyber defense implementations that realize these assumptions. The following items are beyond the scope of this specification:

1. Language elements applicable to some Actuator functions, which may be defined in individual Actuator profiles.
2. Alternate serializations of Commands and Responses.
3. The enumeration of the protocols required for transport, information assurance, sensing, analytics and other external dependencies.

2 OpenC2 Language Description

The content in this section is non-normative.

The OpenC2 language has two distinct content types: Command and Response. The Command is sent from a Producer to a Consumer and describes an Action to be performed by the Consumer on a Target. The Response is sent from a Consumer, usually back to the Producer, and is a means to provide information (such as acknowledgment, status, etc.) as a result of a Command.

2.1 OpenC2 Command

A command has four main components, two required and two optional. The required components are the Action and the Target. The optional components are command Arguments and the Profile identifier. A command can also contain an optional Command identifier, if necessary. [Section 3.3.1](#) defines the syntax of an OpenC2 Command.

The following list summarizes the main four components of a command.

- **Action** (required): The task or activity to be performed.
- **Target** (required): The object of the action. The Action is performed on the Target. Properties of the Target, called Target Specifiers, further identify the Target to some level of precision, such as a specific Target, a list of Targets, or a class of Targets.
- **Arguments** (optional): Provide additional information on how the command is to be performed, such as date/time, periodicity, duration, etc.
- **Profile** (optional): Specifies the Actuator Profile that defines the function to be performed by the command.

The Action and Target components are required and are populated by one of the Actions in [Section 3.3.1.1](#) and either a Target from [Section 3.3.1.2](#) or an extension Target defined in an AP. A particular Target may be further refined by the Target type definitions in [Section 3.4.1](#). Procedures to extend the Targets are described in [Section 3.1.4](#).

Command Arguments, if present, influence the command by providing information such as timing, periodicity, duration, or other details on what is to be executed. They can also be used to convey the need for acknowledgment or additional status information about the execution of a command. The valid Arguments defined in this specification are in [Section 3.3.1.4](#). Procedures to extend Arguments are described in [Section 3.1.4](#).

The Profile field, if present, specifies the profile that defines the function to be performed. A Consumer executes the command if it supports the specified profile, otherwise the command is ignored. The Profile field may be omitted and typically will not be included in implementations where the functions of the recipients are unambiguous or when a high-level effects-based command is desired and tactical decisions on how the effect is achieved is left

to the recipient. If Profile is omitted and the recipient supports multiple profiles, the command will be executed in the context of each profile that supports the command's combination of action and target.

2.2 OpenC2 Response

The Response is a Message sent from the recipient of a Command. Response messages provide acknowledgment, status, results from a query, or other information. At a minimum, a Response will contain a status code to indicate the result of performing the Command. Additional status text and response fields optionally provide more detailed information that is specific to or requested by the Command. [Section 3.3.2](#) defines the syntax of an OpenC2 Response.

3 OpenC2 Language Definition

The content in this section is normative.

3.1 Base Components and Structures

OpenC2 data types are defined using the types and options available in [\[JADN\]](#). JADN is a UML-based *information modeling* language that defines data structure independently of data format. [RFC 3444](#), "Information Models and Data Models", describes the main purpose of an *information model* as modeling objects at a conceptual level, independent of specific implementations or protocols used to transport the data. This concept of an information model is consistent with the goal of defining OpenC2 commands and responses independent of their representation in any specific implementation. JADN provides a tool for developing information models, which can be used to define and generate physical data models, validate information instances, and enable lossless translation across data formats. Information modeling concepts are discussed in more detail in *Information Modeling with JADN* [\[IM-JADN-v1.0\]](#).

This section provides a concise summary of JADN type definitions to facilitate readability; for more complete explanations refer to the JADN Committee Specification and the *Information Modeling with JADN* Committee Note.

3.1.1 Data Types

The use of JADN enables type definitions that are independent of both their representation within applications ("**API**" values) and their format for transmission between applications ("**serialized**" values). The data types used in OpenC2 Messages are:

Type	Description
Primitive Types	
Binary	A sequence of octets. Length is the number of octets.
Boolean	An element with one of two values: <code>true</code> and <code>false</code> .
Integer	A positive or negative whole number.
Number	A real number.
String	A sequence of characters, each of which has a Unicode codepoint. Length is the number of characters.
Structures	

Type	Description
Array	An ordered list of unnamed fields with positionally-defined semantics. Each field has a position, label, and type.
ArrayOf(<i>vtype</i>)	An ordered list of fields with the same semantics. Each field has a position and type <i>vtype</i> .
Choice	One field selected from a set of named fields. The API value has a name and a type.
Choice.ID	One field selected from a set of fields. The API value has an id and a type.
Enumerated	A set of named integral constants. The API value is a name.
Enumerated.ID	A set of unnamed integral constants. The API value is an id.
Map	An unordered map from a set of specified keys to values with semantics bound to each key. Each field has an id, name and type.
Map.ID	An unordered set of fields. The API value of each field has an id, label, and type.
MapOf(<i>ktype</i> , <i>vtype</i>)	An unordered set of keys to values with the same semantics. Each key has key type <i>ktype</i> and is mapped to value type <i>vtype</i> .
Record	An ordered map from a list of keys with positions to values with positionally-defined semantics. Each key has a position and name, and is mapped to a type. Represents a row in a spreadsheet or database table.

- **API** values do not affect interoperability, and although they must exhibit the characteristics specified above, their representation within applications is unspecified. A Python application might represent the Map type as a dict variable, a javascript application might represent it as an object literal or an ES6 Map type, and a C# application might represent it as a Dictionary or a Hashtable.
- **Serialized** values are critical to interoperability, and this document defines a set of **serialization rules** that unambiguously define how each of the above types are serialized using a human-friendly JSON format. Other serialization rules, such as for XML, machine-optimized JSON, and Concise Binary Object Representation (CBOR) formats, exist but are out of scope for this document.

Usage Requirement:

- OpenC2 messages in JSON format SHALL be serialized in accordance with the rules defined in the JADN Specification [[JADN](#)], section 4.

OpenC2 type definitions are presented in this specification in table format. All table columns except Description are Normative, however any conflict between the table presentation and the JADN schema must be resolved by applying the schema definition. The Description column is always Non-normative.

Usage Requirement:

- The format-agnostic type definitions in [Section 3.4](#) are Normative.

For types without individual field definitions (Primitive types and ArrayOf), the type definition includes the name of the type being defined and the definition of that type. This table defines a type called *Email-Addr* that is a *String* that has a semantic value constraint of *email*:

Type Name	Type Definition	Description
Email-Addr	String (email)	Email address

For Structured types, the definition includes the name of the type being defined, the built-in type on which it is based, and options applicable to the type as a whole. This is followed by a table defining each of the fields in the structure. This table defines a type called *Args* that is a *Map* containing at least one field. Each of the fields has an integer Tag/ID, a Name, and a Type. Each field in this example type definition is optional (Multiplicity = 0..1), but per the type definition (Multiplicity = 1..*) at least one field must be present.

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

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

The field columns present in a structure definition depends on the base type:

Base Type	Field Definition Columns
Enumerated.ID	ID, Description
Enumerated	ID, Name, Description
Array, Choice.ID, Map.ID	ID, Type, Multiplicity (#), Description
Choice, Map, Record	ID, Name, Type, Multiplicity (#), Description

The ID column of Array and Record types contains the ordinal position of the field, numbered sequentially starting at 1. The ID column of Choice, Enumerated, and Map types contains tags with arbitrary integer values. IDs and Names are unique within each type definition.

3.1.2 Semantic Value Constraints

Structural validation alone may be insufficient to validate that an instance meets all the requirements of an application. Semantic validation keywords specify value constraints for which an authoritative definition exists.

Keyword	Applies to Type	Constraint
email	String	Value must be an email address as defined in [RFC5322] , Section 3.4.1
eui	Binary	Value must be an EUI-48 or EUI-64 as defined in [EUI]
hostname	String	Value must be a hostname as defined in [RFC1034] , Section 3.1
idn-email	String	Value must be an internationalized email address as defined in [RFC6531]
idn-hostname	String	Value must be an internationalized hostname as defined in [RFC5890] , Section 2.3.2.3
iri	String	Value must be an Internationalized Resource Identifier (IRI) as defined in [RFC3987]
uri	String	Value must be a Uniform Resource Identifier (URI) as defined in [RFC3986]

Usage Requirements:

- Properties identified as conforming to **eui** SHOULD be interpreted according to the values documented in the [\[IEEE Registration Authority registry\]](#).

3.1.3 Multiplicity

Property tables for types based on Array, Choice, Map and Record include a multiplicity column (#) that specifies the minimum and maximum cardinality (number of elements) of a field. As used in the Unified Modeling Language ([\[UML\]](#)), typical examples of multiplicity are:

Multiplicity	Description	Keywords
--------------	-------------	----------

Multiplicity	Description	Keywords
1	Exactly one instance	Required
0..1	No instances or one instance	Optional
1..*	At least one instance	Required, Repeatable
0..*	Zero or more instances	Optional, Repeatable
m..n	At least m but no more than n instances	Required, Repeatable

When a repeatable field type is converted to a separate `ArrayOf()` Type, multiplicity is converted to the array size, enclosed in curly brackets, e.g.,:

Type Name	Type Definition	Description
Features	<code>ArrayOf(Feature) {0..10}</code>	An array of zero to ten names used to query an actuator for its supported capabilities.

A multiplicity of 0..1 denotes a single optional value of the specified type. A multiplicity of 0..n denotes a field that is either omitted or is an array containing one or more values of the specified type.

An array containing zero or more values of a specified type cannot be created implicitly using multiplicity, it must be defined explicitly as a named `ArrayOf` type. The named type can then be used as the type of a required field (multiplicity 1). Results are unspecified if an optional field (multiplicity 0..1) is a named `ArrayOf` type with a minimum length of zero.

3.1.4 Extensions

One of the main design goals of OpenC2 is extensibility. Actuator Profiles define the language extensions that are meaningful and possibly unique to the Actuator.

Each Profile has a unique name used to identify the profile document and a short reference called a namespace identifier (NSID). The NSID is a prefix used to separate types defined in one profile document from types defined in other profiles or this specification. OpenC2 conventions for managing namespaces are described in Appendix F of the OpenC2 *Architecture Specification* [[Openc2-Arch-v1.0](#)].

Example: the OASIS standard Stateless Packet Filtering profile has:

- **Namespace:** <http://docs.oasis-open.org/openc2/ns/ap/slpf/v1.0>
- **NSID:** slpf
- **Language-defined type:** IPv4-Net

- **Profile-defined type:** slpf/Rule-ID

Example: the fictional, non-standard Superwidget Profile has:

- **Namespace:** <http://www.acme.com/openc2/superwidget-v1.0.html>
- **NSID:** acmesw
- **Language-defined type:** Device
- **Profile-defined type:** acmesw/Device

Usage Requirements:

- The list of Actions in [Section 3.3.1.1](#) SHALL NOT be extended.
- Targets, defined in [Section 3.3.1.2](#), MAY be extended. Extended Target type names MUST be prefixed with a namespace identifier followed by a slash ("/"). Extended target properties appear beneath (nested within) a profile property name.

Example: The Stateless Packet Filtering Profile supports both common and profile-specific targets:

Targets used in Consumers that support the SLPF actuator profile:

Type: Target (Choice)

ID	Name	Type	Description
13	ipv4_net	IPv4-Net	Targets defined in the LS
1024	slpf	slpf/AP-Target	Targets defined in the SLPF AP

Targets defined in the SLPF actuator profile:

Type: slpf/AP-Target (Choice)

ID	Name	Type	Description
1	rule_number	slpf/Rule-ID	

In this example Command, the extended Target `rule_number` of type `slpf/Rule-ID` appears within the SLPF profile property name `slpf`:

```
{
  "action": "delete",
  "target": {
    "slpf": {
      "rule_number": 1234
    }
  }
}
```

```

    }
  }
}

```

Usage Requirement:

- Command Arguments, defined in [Section 3.3.1.4](#), MAY also be extended using profile-defined types appearing within the profile property name.

Type: Args (Map)

ID	Name	Type	Description
1	start_time	Date-Time	Args defined in the LS
1024	slpf	slpf/AP-Args	Args defined in the SLPF AP

Args defined in the SLPF actuator profile:

Type: slpf/AP-Args (Map)

ID	Name	Type	Description
3	direction	slpf/Direction	

Example: In this example Command, the extended Argument, `direction` of type `slpf:Direction` contained in type `slpf:AP-Args`, appears in the Stateless Packet Filtering property name `slpf`:

```

{
  "action": "deny",
  "target": {
    "ipv6_net": {...}
  },
  "args": {
    "slpf": {
      "direction": "ingress"
    }
  }
}

```

The Profile property of a Command, defined in [Section 3.3.1.3](#), specifies the property name of the Actuator Profile that defines the function to be performed.

Example: In this example Command, the profile name `slpf` indicates that the `deny`

`ipv4_connection` command is to be performed as defined by the Stateless Packet Filtering Profile.

```
{
  "action": "deny",
  "target": {
    "ipv4_connection": {...}
  },
  "profile": "slpf"
}
```

Usage Requirement:

- Response results, defined in [Section 3.3.2.2](#), MAY be extended using the namespace identifier as the results name, called an extended results namespace. Extended results MUST be defined within the extended results namespace.

Example: In this example Response, the Response results property, `rule_number`, is defined within the Stateless Packet Filtering Profile namespace, `slpf`.

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

3.1.5 Serialization

OpenC2 is agnostic of any particular serialization; however, implementations MUST support JSON serialization in accordance with

- [\[RFC7493\]](#) and
- [\[JADN\]](#).

Types defined with the ".ID" appended to the base type (Enumerated, Choice, Map) are serialized in accordance with [\[JADN\]](#).

3.2 Message

This language specification and one or more Actuator Profiles define the content of Commands and Responses, while transfer specifications define the on-the-wire format of a Message over specific secure transport protocols. Transfer specifications are agnostic with regard to content, and content is agnostic with regard to transfer protocol. This decoupling is

accomplished by defining a standard message interface used to transfer any type of content over any transfer protocol.

A message is a content- and transport-independent set of elements conveyed between Producers and Consumers. To ensure interoperability all transfer specifications must unambiguously define how the Message elements in [Table 3-1](#) are represented within the secure transport protocol. This does not imply that all Message elements must be used in all Messages. Content, content_type, and msg_type are required in all Messages. Other Message elements are not required by this specification but may be required by other specifications. The internal representation of a Message does not affect interoperability and is therefore beyond the scope of OpenC2.

Table 3-1. Common Message Elements

Name	Type	Description
content		Message body as specified by content_type and msg_type.
content_type	String	Media Type that identifies the format of the content, including major version. Incompatible content formats must have different content_types. Content_type application/openc2 identifies content defined by OpenC2 language specification versions 1.x, i.e., all versions that are compatible with version 1.1.
msg_type	Message-Type	The type of OpenC2 Message.
status	Status-Code	Populated with a numeric status code in Responses.
request_id	Command-ID	A unique identifier created by the Producer and copied by Consumer into all Responses, in order to support reference to a particular Command, transaction, or event chain.
created	Date-Time	Creation date/time of the content.
from	String	Authenticated identifier of the creator of or authority for execution of a message.
to	ArrayOf(String)	Authenticated identifier(s) of the authorized recipient(s) of a message.

Usage Requirement:

- As an alternative to using protocol-specific mechanisms to convey message elements, transfer specifications MAY collect all message elements into a single Message

structure used as a protocol payload.

The media type "application/openc2" is reserved with IANA to designate content in OpenC2 Message format. The Message structure and its media type are intended to remain stable across future versions of this specification.

Type: Message (Record)

ID	Name	Type	#	Description
1	headers	Headers	0..1	
2	body	Body	1	
3	signature	String	0..1	

Headers contains optional common message elements. Additional constraints on common header values may be defined. Additional headers may be defined. The "signature" field is used to contain an optional digital signature to provide source authentication and integrity protections of the OpenC2 message.

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

ID	Name	Type	#	Description
1	request_id	Command-ID	0..1	
2	created	Date-Time	0..1	
3	from	String	0..1	
4	to	String	0..*	

Body indicates the Message content format and is intended to support new types of OpenC2 Content such as command lists or bundle objects, but OpenC2 may also assign Body types for non-OpenC2 content such as STIX or CACAO objects.

Type: Body (Choice)

ID	Name	Type	#	Description
1	openc2	OpenC2-Content	1	

Type: OpenC2-Content (Choice)

ID	Name	Type	#	Description
1	request	OpenC2-Command	1	
2	response	OpenC2-Response	1	
3	notification	OpenC2-Event	1	

Example JSON-serialized Message payload (without signature):

```
{
  "headers": {
    "request_id": "95ad511c-3339-4111-9c47-9156c47d37d3",
    "created": 1595268027000,
    "from": "Producer1@example.com",
    "to": ["consumer1@example.com", "consumer2@example.com",
"consumer3@example.com"]
  },
  "body": {
    "openc2": {
      "request": {
        "action": "deny",
        "target": {
          "uri": "http://www.example.com" }}}}
}
```

Usage Requirements:

- A Producer **MUST** include a `request_id` in the Message header of a Command if it requests a Response.
- The `request_id` of a Message **SHOULD** be a Version 4 UUID as specified in [\[RFC4122\]](#), Section 4.4.
- A Consumer **MUST** copy the `request_id` from the Message header of a Command into each Response to that Command.

3.3 Content

The purpose of this specification is to define the Action and Target portions of a Command and the common portions of a Response. The properties of the Command are defined in [Section 3.3.1](#) and the properties of the Response are defined in [Section 3.3.2](#).

In addition to the Action and Target, a Command has an optional Profile field. The semantics associated with Command and Response content are defined in the specified Actuator Profile.

3.3.1 OpenC2 Command

The Command defines an Action to be performed on a Target.

Type: OpenC2-Command (Record)

ID	Name	Type	#	Description
1	action	Action	1	The task or activity to be performed (i.e., the 'verb').
2	target	Target	1	The object of the Action. The Action is performed on the Target.
3	args	Args	0..1	Additional information that applies to the Command.
4	profile	Profile	0..1	The actuator profile defining the function to be performed by the Command.
5	command_id	Command-ID	0..1	An identifier of this Command.

Usage Requirements:

- A Consumer receiving a Command with `command_id` absent and `request_id` present in the header of the Message MUST use the value of `request_id` as the `command_id`.
- If present, the `args` property MUST contain at least one element defined in [Section 3.3.1.4](#).
- If a Consumer that implements multiple actuator profiles receives a Command with no `Profile` specified, the command will be executed in the context of each profile that supports the command's combination of action and target.

3.3.1.1 Action

Type: Action (Enumerated)

ID	Name	Description
1	scan	Systematic examination of some aspect of the entity or its environment.
2	locate	Find an object physically, logically, functionally, or by organization.
3	query	Initiate a request for information.

ID	Name	Description
6	deny	Prevent a certain event or action from completion, such as preventing a flow from reaching a destination or preventing access.
7	contain	Isolate a file, process, or entity so that it cannot modify or access assets or processes.
8	allow	Permit access to or execution of a Target.
9	start	Initiate a process, application, system, or activity.
10	stop	Halt a system or end an activity.
11	restart	Stop then start a system or an activity.
14	cancel	Invalidate a previously issued Action.
15	set	Change a value, configuration, or state of a managed entity.
16	update	Instruct a component to retrieve, install, process, and operate in accordance with a software update, reconfiguration, or other update.
18	redirect	Change the flow of traffic to a destination other than its original destination.
19	create	Add a new entity of a known type (e.g., data, files, directories).
20	delete	Remove an entity (e.g., data, files, flows).
22	detonate	Execute and observe the behavior of a Target (e.g., file, hyperlink) in an isolated environment.
23	restore	Return a system to a previously known state.
28	copy	Duplicate an object, file, data flow, or artifact.
30	investigate	Task the recipient to aggregate and report information as it pertains to a security event or incident.
32	remediate	Task the recipient to eliminate a vulnerability or attack point.

Usage Requirements:

- Each Command MUST contain exactly one Action defined in [Section 3.3.1.1](#).

3.3.1.2 Target

Type: Target (Choice)

ID	Name	Type	#	Description
1	artifact	Artifact	1	An array of bytes representing a file-like object or a link to that object.
2	command	Command-ID	1	A reference to a previously issued Command.
3	device	Device	1	The properties of a hardware device.
7	domain_name	Domain-Name	1	A network domain name.
8	email_addr	Email-Addr	1	A single email address.
9	features	Features	1	A set of items used with the query Action to determine an Actuator's capabilities.
10	file	File	1	Properties of a file.
11	idn_domain_name	IDN-Domain-Name	1	An internationalized domain name.
12	idn_email_addr	IDN-Email-Addr	1	A single internationalized email address.
13	ipv4_net	IPv4-Net	1	An IPv4 address range including CIDR prefix length.
14	ipv6_net	IPv6-Net	1	An IPv6 address range including an address and a prefix length.
15	ipv4_connection	IPv4-Connection	1	A 5-tuple of source and destination IPv4 address ranges, source and destination ports, and protocol.
16	ipv6_connection	IPv6-Connection	1	A 5-tuple of source and destination IPv6 address ranges, source and destination ports, and protocol.
20	iri	IRI	1	An internationalized resource identifier (IRI).
17	mac_addr	MAC-Addr	1	A Media Access Control (MAC) address - EUI-48 or EUI-64 as defined in [EUI] .

ID	Name	Type	#	Description
18	process	Process	1	Common properties of an instance of a computer program as executed on an operating system.
19	uri	URI	1	A uniform resource identifier (URI).

Usage Requirements:

- The `target` field in a Command MUST contain exactly one type of Target (e.g., `ipv4_net`).

3.3.1.3 Profile

OpenC2 maintains an [administrative document](#) listing current, planned, and extension actuator profile information.

Type: Profile (Enumerated)

ID	Name	Description
1024	slpf	Stateless Packet Filtering
1025	sfpf	Stateful Packet Filtering
1026	sbom	Software Bill of Materials
1027	er	Endpoint Response
1028	hop	Honeypot Control
1029	av	Anti-Virus
1030	ids	Intrusion Detection System
1031	log	Logging Control
1032	swup	Software Update
1034	pf	Packet Filtering
1035	pac	Security Posture Attribute Collection

3.3.1.4 Command Arguments

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

ID	Name	Type	#	Description
1	start_time	Date-Time	0..1	The specific date/time to initiate the Command
2	stop_time	Date-Time	0..1	The specific date/time to terminate the Command
3	duration	Duration	0..1	The length of time for a Command to be in effect
4	response_requested	Response-Type	0..1	The type of Response required for the Command: <code>none</code> , <code>ack</code> , <code>status</code> , <code>complete</code> .
5	comment	String	0..1	A human-readable note to annotate or provide information regarding the action.

Usage Requirements:

- `start_time`, `stop_time`, `duration`:
 - If none are specified, then `start_time` is `now`, `stop_time` is `never`, and `duration` is `infinity`.
 - Only two of the three are allowed on any given Command and the third is derived from the equation `stop_time = start_time + duration`.
 - If only `start_time` is specified then `stop_time` is `never` and `duration` is `infinity`.
 - If only `stop_time` is specified then `start_time` is `now` and `duration` is derived.
 - If only `duration` is specified then `start_time` is `now` and `stop_time` is derived.
- `response_requested`:
 - If `response_requested` is specified as `none` and the Consumer successfully executes the Command then the Consumer SHOULD NOT send a Response.
 - If `response_requested` is specified as `none` and the Consumer encounters an error then the Consumer SHOULD send a Response with a `status` consistent with the error detected.
 - If `response_requested` is specified as `ack` then the Consumer SHOULD send a

Response acknowledging receipt of the Command: {"status": 102}.

- If `response_requested` is specified as `status` then the Consumer SHOULD send a Response containing the current status of Command execution.
- If `response_requested` is specified as `complete` then the Consumer SHOULD send a Response containing the status or results upon completion of Command execution.
- If `response_requested` is not explicitly specified then the Consumer SHOULD respond as if `complete` was specified.

3.3.2 OpenC2 Response

OpenC2-Response defines the structure of a response to OpenC2-Command.

Type: OpenC2-Response (Record)

ID	Name	Type	#	Description
1	status	Status-Code	1	An integer status code.
2	status_text	String	0..1	A free-form human-readable description of the Response status.
3	results	Results	0..1	Map of key:value pairs that contain additional results based on the invoking Command.

Example:

```
{
  "status": 200,
  "results": {
    "versions": ["1.1"]
  }
}
```

Usage Requirements:

- All Responses MUST contain a status.

3.3.2.1 Response Status Code

Type: Status-Code (Enumerated.ID)

ID	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).
401	Unauthorized - the Command Message lacks valid authentication credentials for the target resource or authorization has been refused for the submitted credentials.
403	Forbidden - the Consumer understood the Command but refuses to authorize it.
404	Not Found - the Consumer has not found anything matching the Command.
500	Internal Error - the Consumer encountered an unexpected condition that prevented it from performing the Command.
501	Not Implemented - the Consumer does not support the functionality required to perform the Command.
503	Service Unavailable - the Consumer is currently unable to perform the Command due to a temporary overloading or maintenance of the Consumer.

3.3.2.2 Response Results

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

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

Usage Requirements:

NOTE: Confirm whether any language schema changes are necessary to

support this.

- `pairs` results SHOULD be grouped by profile such that a Producer processing the Response message can identify the commands available for each supported profile. See [Example 3](#) in [Section 4.2](#).

3.3.3 OpenC2 Event

OpenC2-Event defines the content of a one-way notification. This structure defines no common event fields, but is the point at which profile-defined event content may be added.

Type: OpenC2-Event (Map{1..*})

ID	Name	Type	#	Description
----	------	------	---	-------------

3.3.4 Message Signatures

To-Do: develop a replacement JSS-based example and add to Appendix E.
Need to find tooling to assist.

Command and control mechanisms need to provide appropriate security controls protecting message content (especially authentication of command origin and protection of command integrity) so that Consumers receiving commands can proceed to execute them with confidence and Producers can have confidence that the feedback in response messages is meaningful. Digital signatures can provide both of those security properties. OpenC2 messages can be protected with digital signatures using standard mechanisms. ITU-T Recommendation X.590 [\[ITU-T X.590\]](#), *JSON Signature Scheme (JSS)* provides a signature mechanism for JSON content that aligns with the needs of OpenC2.

Usage Requirements:

- OpenC2 messages SHOULD be digitally signed, unless message integrity and source authentication are provided by other mechanisms.
- OpenC2 messages serialized in JSON MUST conform to the requirements of [RFC 7493](#) to support canonicalization.
- Digitally-signed OpenC2 messages serialized in JSON MUST be signed using JSON Signature Scheme in accordance with ITU-T X.590.

The method for message recipients to identify and validate the appropriate public key to validate a message signature is beyond the scope of this specification. Alternative, appropriate signature mechanisms will need to be specified for serializations other than JSON.

3.4 Type Definitions

3.4.1 Target Types

3.4.1.1 Artifact

Type: Artifact (Record{1..*})

ID	Name	Type	#	Description
1	media_type	String	0..1	Media type description formatted as specified in [RFC6838]
2	payload	Payload	0..1	Choice of literal content or URL
3	hashes	Hashes	0..1	Hashes of the payload content

Usage Requirements:

- An "Artifact" Target MUST contain at least one property.
- `media_type` "Artifact" property values SHOULD be interpreted according to the values documented in the [IANA Media Types registry](#).

3.4.1.2 Device

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

ID	Name	Type	#	Description
1	hostname	Hostname	0..1	A hostname that can be used to connect to this device over a network
2	idn_hostname	IDN-Hostname	0..1	An internationalized hostname that can be used to connect to this device over a network
3	device_id	String	0..1	An identifier that refers to this device within an inventory or management system

Usage Requirement:

- A "Device" Target MUST contain at least one property.

3.4.1.3 Domain Name

Type Name	Type Definition	Description
Domain-Name	String /hostname	[RFC1034] , Section 3.5

3.4.1.4 Email Address

Type Name	Type Definition	Description
Email-Addr	String /email	Email address, RFC5322 , Section 3.4.1

3.4.1.5 Features

Type Name	Type Definition	#	Description
Features	Feature unique	0..*	An array of feature names used to query a Consumer for its supported capabilities.

Usage Requirements:

- A Producer **MUST NOT** send a list containing more than one instance of any Feature.
- A Consumer receiving a list containing more than one instance of any Feature **SHOULD** behave as if the duplicate(s) were not present.
- A Producer **MAY** send a 'query' Command containing an empty list of features. A Producer could do this to determine if a Consumer is responding to Commands (a heartbeat command) or to generate idle traffic to keep a connection to a Consumer from being closed due to inactivity (a keep-alive command). An active Consumer could return an empty response to this command, minimizing the amount of traffic used to perform heartbeat / keep-alive functions.

3.4.1.6 File

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

ID	Name	Type	#	Description
1	name	String	0..1	The name of the file as defined in the file system
2	path	String	0..1	The absolute path to the location of the file in the file system
3	hashes	Hashes	0..1	One or more cryptographic hash codes of the file contents

Usage Requirement:

- A "File" Target **MUST** contain at least one property.

3.4.1.7 Internationalized Domain Name

Type Name	Type Definition	Description
-----------	-----------------	-------------

Type Name	Type Definition	Description
IDN-Domain-Name	String /idn-hostname	Internationalized Domain Name, [RFC5890] , Section 2.3.2.3.

3.4.1.8 Internationalized Email Address

Type Name	Type Definition	Description
IDN-Email-Addr	String /idn-email	Internationalized email address, [RFC6531]

3.4.1.9 IPv4 Address Range

An IPv4 address range is a CIDR block per "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan" [\[RFC4632\]](#) and consists of two values, an IPv4 address and a prefix.

For example, "192.168.17.0/24" is range of IP addresses with a prefix of 24 (i.e. 192.168.17.0 - 192.168.17.255).

Type: IPv4-Net (Array /ipv4-net)

ID	Type	#	Description
1	IPv4-Addr	1	IPv4 address as defined in [RFC0791]
2	Integer	0..1	CIDR prefix-length. If omitted, refers to a single host address.

Usage Requirements:

- JSON serialization of an IPv4 address range SHALL use the 'dotted/slash' textual representation of [\[RFC4632\]](#).
- CBOR serialization of an IPv4 address range SHALL use a binary representation of the IP address and the prefix, each in their own field.

3.4.1.10 IPv4 Connection

Type: IPv4-Connection (Record{1..*})

ID	Name	Type	#	Description
1	src_addr	IPv4-Net	0..1	IPv4 source address range
2	src_port	Port	0..1	Source service per [RFC6335]

ID	Name	Type	#	Description
3	dst_addr	IPv4-Net	0..1	IPv4 destination address range
4	dst_port	Port	0..1	Destination service per RFC6335
5	protocol	L4-Protocol	0..1	Layer 4 protocol (e.g., TCP) - see Section 3.4.2.11

Usage Requirement:

- An "IPv4-Connection" MUST contain at least one property.

3.4.1.11 IPv6 Address Range

An IPv6 address range is a block of contiguous IPv6 addresses, per "IP Version 6 Addressing Architecture" [RFC4291](#) and consists of two values, an IPv6 address and a prefix.

For example, "2001:0DB8:0000:CD30:0000:0000:0000/60" is range of IPv6 addresses with a 60-bit prefix. Per Section 2.3 of RFC4291, the address portion may completely specify a node address (i.e., all 128-bits are meaningful) while the prefix value identifies the node's subnet.

Type: IPv6-Net (Array /ipv6-net)

ID	Type	#	Description
1	IPv6-Addr	1	IPv6 address as defined in RFC8200
2	Integer	0..1	Prefix-length

Usage Requirements:

- JSON serialization of an IPv6 address range SHALL use the textual representation of IPv6 addresses and prefix lengths defined in [RFC4291](#).
- CBOR serialization of an IPv6 address range SHALL use a binary representation of the IP address and the prefix, each in their own field.

3.4.1.12 IPv6 Connection**Type: IPv6-Connection (Record{1..*})**

ID	Name	Type	#	Description
1	src_addr	IPv6-Net	0..1	IPv6 source address range

ID	Name	Type	#	Description
2	src_port	Port	0..1	Source service per [RFC6335]
3	dst_addr	IPv6-Net	0..1	IPv6 destination address range
4	dst_port	Port	0..1	Destination service per [RFC6335]
5	protocol	L4-Protocol	0..1	Layer 4 protocol (e.g., TCP) - Section 3.4.2.11

Usage Requirement:

- An "IPv6-Connection" Target MUST contain at least one property.

3.4.1.13 IRI

Type Name	Type Definition	Description
IRI	String /iri	Internationalized Resource Identifier, [RFC3987] .

3.4.1.14 MAC Address

Type Name	Type Definition	Description
MAC-Addr	Binary /eui	Media Access Control / Extended Unique Identifier address - EUI-48 or EUI-64 as defined in [EUI] .

3.4.1.15 Process**Type: Process (Map{1..*})**

ID	Name	Type	#	Description
1	pid	Key(Integer{0..*})	0..1	Process ID of the process
2	name	String	0..1	Name of the process
3	cwd	String	0..1	Current working directory of the process
4	executable	File	0..1	Executable that was executed to start the process
5	parent	Link(Process)	0..1	Process that spawned this one

ID	Name	Type	#	Description
6	command_line	String	0..1	The full command line invocation used to start this process, including all arguments

Usage Requirement:

- A "Process" Target **MUST** contain at least one property.

3.4.1.16 URI

Type Name	Type Definition	Description
URI	String (uri)	Uniform Resource Identifier, [RFC3986] .

3.4.2 Data Types**3.4.2.1 Action-Targets**

Type Name	Type Definition	Description
Action-Targets	MapOf(Action, Targets){1..*}	Map of each action supported by this actuator function to the list of targets applicable to that action.

3.4.2.2 Command-ID

Included in a Command message by an OpenC2 Producer to enable future references to the specific command. OpenC2 Consumers can include the `command_id` from the Command message in Responses to specify the Command to which the Response corresponds.

Type Name	Type Definition	Description
Command-ID	String (%^\S{0,36}\$%)	Command Identifier

Usage Requirement:

- Value **SHOULD** be a string representation of a UUIDv4 as defined in [\[RFC4122\]](#).

3.4.2.3 Date-Time

Type Name	Type Definition	Description
Date-Time	Integer{0..*}	Date and Time

Usage Requirement:

- Value is the number of milliseconds since 00:00:00 UTC, 1 January 1970

3.4.2.4 Duration

Type Name	Type Definition	Description
Duration	Integer{0..*}	A length of time

Usage Requirement:

- Value is a number of milliseconds

3.4.2.5 Feature

Specifies the results to be returned from a query features Command.

Type: Feature (Enumerated)

ID	Name	Description
1	versions	List of OpenC2 Language versions supported by this Consumer
2	profiles	List of Actuator profiles supported by this Consumer
3	pairs	List of supported Actions and applicable Targets
4	rate_limit	Maximum number of Commands per minute supported by design or policy

3.4.2.6 Hashes

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

ID	Name	Type	#	Description
1	md5	Binary /x	0..1	MD5 hash as defined in [RFC1321]
2	sha1	Binary /x	0..1	SHA1 hash as defined in [RFC6234]
3	sha256	Binary /x	0..1	SHA256 hash as defined in [RFC6234]

Usage Requirement:

- A "Hashes" data type MUST contain at least one property.

3.4.2.7 Hostname

Type Name	Type Definition	Description
Hostname	String /hostname	Internet host name as specified in [RFC1123]

3.4.2.8 Internationalized Hostname

Type Name	Type Definition	Description
IDN-Hostname	String /idn-hostname	Internationalized Internet host name as specified in [RFC5890] , Section 2.3.2.3.

3.4.2.9 IPv4 Address

Type Name	Type Definition	Description
IPv4-Addr	Binary /ipv4-addr	32 bit IPv4 address as defined in [RFC0791]

3.4.2.10 IPv6 Address

Type Name	Type Definition	Description
IPv6-Addr	Binary /ipv6-addr	128 bit IPv6 address as defined in [RFC8200]

3.4.2.11 L4 Protocol

Value of the IPv4 "protocol" or IPv6 "next header" field in an IP packet. Recognized values for these fields are registered with IANA, according to the process defined in [\[RFC5237\]](#). The table below identifies a non-exhaustive set of commonly used values.

Type: L4-Protocol (Enumerated)

ID	Name	Description
1	icmp	Internet Control Message Protocol - [RFC0792]
6	tcp	Transmission Control Protocol - [RFC9293]
17	udp	User Datagram Protocol - [RFC0768]
28	ipv6_icmp	ICMP for IPv6 - [RFC8200]
132	sctp	Stream Control Transmission Protocol - [RFC4960]

Usage Requirements:

- The IPv4 Protocol field and IPv6 Next Header field are 8-bit values, therefore L4-Protocol is an enumeration from 0..255.
- Values of L4-Protocol SHOULD be interpreted as documented in [\[IANA_Protocols\]](#).

3.4.2.12 Message-Type

Identifies the type of Message.

Type: Message-Type (Enumerated)

ID	Name	Description
1	command	The Message content is an OpenC2 Command
2	response	The Message content is an OpenC2 Response

3.4.2.13 Namespace Identifier

Type Name	Type Definition	Description
Nsid	String{1..16}	A short identifier that refers to a namespace.

3.4.2.14 Payload

Type: Payload (Choice)

ID	Name	Type	#	Description
1	bin	Binary	1	Specifies the data contained in the artifact
2	url	URI	1	MUST be a valid URL that resolves to the un-encoded content

3.4.2.15 Port

Value of the port specified by a Layer 4 protocol, such as the `source port` in a TCP protocol header. Recognized values for these fields are registered with IANA, according to the process defined in [\[RFC6335\]](#).

Type Name	Type Definition	Description
Port	Integer{0..65535}	Transport Protocol Port Number, [RFC6335]

Usage Requirements:

- The Port type is a 16-bit value, therefore provides an enumeration from 0..65535.
- Values of Port SHOULD be interpreted as documented in [\[IANA_Ports\]](#).

3.4.2.16 Response-Type

Type: Response-Type (Enumerated)

ID	Name	Description
0	none	No response
1	ack	Respond when Command received
2	status	Respond with progress toward Command completion
3	complete	Respond when all aspects of Command completed

3.4.2.17 Targets

Type Name	Type Definition	Description
Targets	ArrayOf(Enum(Target)){1..*} unique	List of Target fields

3.4.2.18 Version

Used to report the version(s) of OpenC2 supported by Consumers.

Type Name	Type Definition	Description
Version	String	OpenC2 version in "Major.Minor" format

Usage Requirement:

- A Version string **MUST** contain the major and minor version numbers, represented as integers separated by a period (e.g., "1.1").

4 Mandatory Commands/Responses

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

A Command consists of an Action/Target pair and associated Specifiers and Arguments. This section enumerates the allowed Commands, identifies which are required or optional to implement, and presents the associated responses.

4.1 Implementation of 'query features' Command

The 'query features' Command is REQUIRED for all Producers and Consumers implementing OpenC2. This section defines the REQUIRED and OPTIONAL aspects of the 'query features' Command and associated response for Producers and Consumers.

The 'query features' Command is REQUIRED for all Producers. The 'query features' Command MAY include one or more Features as defined in [Section 3.4.2.5](#). The 'query features' Command MAY include the "response_requested": "complete" Argument. The 'query features' Command MUST NOT include any other Argument.

The 'query features' Command is REQUIRED for all Consumers. Consumers that receive and parse the 'query features':

- With any Argument other than "response_requested": "complete"
 - MUST NOT respond with OK/200.
 - SHOULD respond with Bad Request/400.
 - MAY respond with the 500 status code.
- With no Target Specifiers MUST respond with response code 200.
- With the "versions" Target Specifier MUST respond with status 200 and populate the versions field with a list of the OpenC2 Language Versions supported by the consumer.
- With the "profiles" Target Specifier MUST respond with status 200 and populate the profiles field with a list of profiles supported by the consumer.
- With the "pairs" Target Specifier MUST respond with status 200 and populate the pairs field with a list of action target pairs that define valid commands supported by the consumer.
- With the "rate_limit" Target Specifier populated:
 - SHOULD respond with status 200 and populate the rate_limit field with the maximum number of Commands per minute that the Consumer may support.
 - MAY respond with status 200 and with the rate_limit field unpopulated.

4.2 Examples of 'query features' Commands and Responses

This section is non-normative.

This sub-section provides examples of `query features` Commands and Responses. The examples provided in this section are for illustrative purposes only and are not to be interpreted as operational examples for actual systems.

4.2.1 Example 1: Bare "query : features"

There are no features specified in the `query features` Command. A simple "OK" Response Message is returned.

Command:

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

Response:

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

4.2.2 Example 2: Query for Specific Features

There are several features requested in the `query features` Command. All requested features can be returned in a single Response Message.

Command:

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

Response:

```
{
  "status": 200,
```

```

    "results": {
      "versions": ["1.1"],
      "profiles": ["slpf", "x-lock"],
      "rate_limit": 30
    }
  }
}

```

4.2.3 Example 3: Query Features Detailed Response with Multiple Profiles

This example illustrates a response to the `query features` command from a Consumer that implements two Profiles: stateless packet filtering (`slpf`) and a notional `blinky` profile. The Command exercises the full range of `query features` options. Both profiles define custom targets:

- `slpf` defines the `rule_number` target
- `blinky` defines the `device` and `display` targets

This example illustrates the response to this mandatory command is split among aspects defined at the Consumer level and aspects defined at the individual Profile level. In a sense, the requirements of [Section 4.1](#) of this specification constitute a base "profile" for the Consumer as a whole, and the results for `versions`, `profiles`, `rate_limit` and the basic `"query": ["features"]` appear in that portion of the response. The remaining content in the response is grouped by the two supported profiles.

Command:

```

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

```

Response:

```

{
  "status": 200,
  "results": {
    "versions": ["1.1"],
    "profiles": ["slpf", "blinky"],
    "pairs": {
      "query": ["features"]
    },
    "rate_limit": 30,
    "slpf": {
      "pairs": {

```

```
    "allow": ["ipv6_net", "ipv6_connection"],
    "deny": ["ipv6_net", "ipv6_connection"],
    "delete": ["slpf/rule_number"],
    "update": ["file"]
  }
},
"blinkky": {
  "pairs": {
    "query": ["blinkky/device"],
    "set": ["blinkky/display"]
  }
}
}
```

5 Conformance

This content in this section is normative.

5.1 Conformance Clause 1: Command

A conformant Command

- 5.1-1 MUST be structured in accordance with [Section 3.3.1](#).
- 5.1-2 MUST include exactly one `action` property defined in accordance with [Section 3.3.1.1](#).
- 5.1-3 MUST include exactly one `target` property defined in accordance with [Section 3.3.1.2](#) or exactly one imported `target` property defined in accordance with [Section 3.1.4](#).
- 5.1-4 MUST include zero or one `profile` property defined in accordance with [Section 3.3.1.3](#).
- 5.1-5 MUST include zero or one `args` property defined in accordance with [Section 3.3.1.4](#) or zero or one imported `args` property defined in accordance with [Section 3.1.4](#).

5.2 Conformance Clause 2: Response

A conformant Response

- 5.2-1 MUST be structured in accordance with [Section 3.3.2](#).
- 5.2-2 MUST include exactly one `status` property defined in accordance with [Section 3.3.2.1](#).

5.3 Conformance Clause 3: Producer

A conformant Producer

- 5.3-1 MUST issue Commands and process Responses in accordance with [Section 4](#).
- 5.3-2 MUST implement JSON serialization of generated Commands in accordance with [\[RFC7493\]](#).
- 5.3-3 MUST implement JSON deserialization of received Responses in accordance with [\[RFC7493\]](#).

5.4 Conformance Clause 4: Consumer

A conformant Consumer

- 5.4-1 MUST process Commands and issue Responses in accordance with [Section 4](#).
- 5.4-2 MUST implement JSON serialization of generated Responses in accordance with [\[RFC7493\]](#).
- 5.4-3 MUST implement JSON deserialization of received Commands in accordance with [\[RFC7493\]](#).

Appendix A. References

This appendix contains the normative and informative references that are used in this document. Normative references are specific (identified by date of publication and/or edition number or version number) and Informative references are either specific or non-specific.

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>.

[OpenC2-Arch-v1.0]

Open Command and Control (OpenC2) Architecture Specification Version 1.0. Edited by Duncan Sparrell. 30 September 2022. OASIS Committee Specification 01. <https://docs.oasis-open.org/openc2/oc2arch/v1.0/cs01/oc2arch-v1.0-cs01.html>. Latest stage: <https://docs.oasis-open.org/openc2/oc2arch/v1.0/oc2arch-v1.0.html>.

[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>

[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. Latest version: <http://docs.oasis-open.org/openc2/oc2slpf/v1.0/oc2slpf-v1.0.html>

[RFC0768]

Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <https://www.rfc-editor.org/info/rfc768>.

[RFC0791]

Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <https://www.rfc-editor.org/info/rfc791>.

[RFC0792]

Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, DOI 10.17487/RFC0792, September 1981, <https://www.rfc-editor.org/info/rfc792>.

[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>.

[RFC1321]

Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, DOI 10.17487/RFC1321, April 1992, <https://www.rfc-editor.org/info/rfc1321>.

[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>.

[RFC3444]

Pras, A., Schoenwaelder, J., "On the Difference between Information Models and Data Models", RFC 3444, January 2003, <https://tools.ietf.org/html/rfc3444>.

[RFC3986]

Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC3987]

Duerst, M. and M. Suignard, "Internationalized Resource Identifiers (IRIs)", RFC 3987, DOI 10.17487/RFC3987, January 2005, <https://www.rfc-editor.org/info/rfc3987>.

[RFC4122]

Leach, P., Mealling, M., and R. Salz, "A Universally Unique Identifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <https://www.rfc-editor.org/info/rfc4122>.

[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>.

[RFC4632]

Fuller, V. and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", BCP 122, RFC 4632, DOI 10.17487/RFC4632, August 2006, <https://www.rfc-editor.org/info/rfc4632>.

[RFC4960]

Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <https://www.rfc-editor.org/info/rfc4960>.

[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>.

[RFC5322]

Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <https://www.rfc-editor.org/info/rfc5322>.

[RFC5890]

Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <https://www.rfc-editor.org/info/rfc5890>.

[RFC6234]

Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <https://www.rfc-editor.org/info/rfc6234>.

[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>.

[RFC6531]

Yao, J. and W. Mao, "SMTP Extension for Internationalized Email", RFC 6531, DOI 10.17487/RFC6531, February 2012, <https://www.rfc-editor.org/info/rfc6531>.

[RFC6838]

Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <https://www.rfc-editor.org/info/rfc6838>.

[RFC7493]

Bray, T., Ed., "The IJSON Message Format", RFC 7493, DOI 10.17487/RFC7493, March 2015, <https://www.rfc-editor.org/info/rfc7493>.

[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>.

[RFC8200]

Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[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>.

[RFC9293]

Eddy, W., "Transmission Control Protocol (TCP)", RFC 9293, DOI: 10.17487/RFC9293, August 2022, <https://www.rfc-editor.org/info/rfc9293>.

[EUI]

"IEEE Registration Authority Guidelines for use of EUI, OUI, and CID", IEEE, August 2017, <https://standards.ieee.org/content/dam/ieee-standards/standards/web/documents/tutorials/eui.pdf>

[IEEE_RA]

"IEEE Registration Authority: Assignments", IEEE, August 2022, <https://regauth.standards.ieee.org/standards-ra-web/pub/view.html#registries>

[IANA_Ports]

"Internet Assigned Numbers Authority Service Name and Transport Protocol Port Number Registry", IANA, April 2024, <https://www.iana.org/assignments/service-names-port-numbers/>

[IANA_Protocols]

"Internet Assigned Numbers Authority Protocol Numbers", IANA, January 2024,
<https://www.iana.org/assignments/protocol-numbers>

[IANA_Media]

"Internet Assigned Numbers Authority Media Types", IANA, April 2024,
<https://www.iana.org/assignments/media-types>

[ITU-T X.590]

Recommendation ITU-T X.509 | ISO/IEC 9594-8 (2019), *Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks. JSON Signature Scheme (JSS)*, <https://www.itu.int/rec/T-REC-X.590-202310-I>

A.2 Informative References

[RFC3552]

Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <https://www.rfc-editor.org/info/rfc3552>.

[IACD]

"What is IACD?", **IACD**, Integrated Adaptive Cyber Defense, 3/17/2018,
<https://www.iacdautomate.org/>

[IM-JADN-v1.0]

Information Modeling with JADN Version 1.0. Edited by David Kemp. 19 April 2023. OASIS Committee Note 01. <https://docs.oasis-open.org/openc2/imjadr/v1.0/cn01/imjadr-v1.0-cn01.html>. Latest stage: <https://docs.oasis-open.org/openc2/imjadr/v1.0/imjadr-v1.0.html>.

[JSON-Schema]

"JSON Schema, a vocabulary that allows you to annotate and validate JSON documents.", retrieved 9/26/2022, <https://json-schema.org/>

[UML]

"Unified Modeling Language", Version 2.5.1, December 2017,
<https://www.omg.org/spec/UML/2.5.1/About-UML/>

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

The content in this section is non-normative.

C.1 Special Thanks

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

- Jason Romano, National Security Agency

C.2 Participants

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

OpenC2 TC Members:

First Name	Last Name	Company
Stephen	Banghart	NIST
Michelle	Barry	AT&T
David	Bizeul	SEKOIA.IO
Jason	Callaway	Google Inc.
Marco	Caselli	Siemens AG
Toby	Considine	University of North Carolina at Chapel Hill
Shiva	Dasari	Hewlett Packard Enterprise (HPE)
Alexandre	Dulaunoy	CIRCL
Alex	Everett	University of North Carolina at Chapel Hill
Jessica	Fitzgerald-McKay	National Security Agency
Jane	Ginn	Cyber Threat Intelligence Network, Inc. (CTIN)
Zachary	Gorak	National Security Agency

First Name	Last Name	Company
Stephanie	Hazlewood	IBM
Tim	Hudson	Cryptsoft Pty Ltd.
Christian	Hunt	Cyber Threat Intelligence Network, Inc. (CTIN)
Andras	Ikody	CIRCL
Ryan	Joyce	DarkLight
Takahiro	Kakumaru	NEC Corporation
Jason	Keirstead	Cyware Labs
David	Kemp	National Security Agency
Lauri	Korts-Pärn	NEC Corporation
Cheolho	Lee	NSR
David	Lemire	National Security Agency
Anthony	Librera	AT&T
Jason	Liu	Northrop Grumman
Terry	MacDonald	Terry MacDonald (Personal)
Patrick	Maroney	AT&T Services, Inc.
Vasileios	Mavroeidis	University of Oslo
Luca	Morgese Zangrandi	TNO
Ben	Ottoman	Cyber Threat Intelligence Network, Inc. (CTIN)
Paul	Patrick	DarkLight
Chris	Ricard	Financial Services Information Sharing and Analysis Center (FS-ISAC)
Daniel	Riedel	Daniel Riedel/Next Level Assurance LLC (Sole Member LLC)

First Name	Last Name	Company
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)
Michael	Simonson	Cisco Systems
Duane	Skeen	Northrop Grumman
Calvin	Smith	Northrop Grumman
Dan	Solero	AT&T Services, Inc.
Ben	Sooter	Electric Power Research Institute (EPRI)
Duncan	Sparrell	sFractal Consulting LLC
Michael	Stair	AT&T
Sam	Taghavi Zargar	Cisco Systems
Bill	Trost	AT&T
Drew	Varner	NineFX, Inc.
Jyoti	Verma	Cisco Systems
David	Waltermire	NIST
Russ	Warren	IBM
Sean	Welsh	AT&T

Appendix D. Revision History

Revision	Date	Editor	Changes Made
v1.1-wd01	10/31/2017	Sparrell, Considine	Initial working draft
Issue #388, item 1, #390	08/xx/2022	Lemire	Add guidance in 3.1.2, 3.4.1.1, 3.4.2.10 regarding types that depend on external registries, and add associated references; update RFC reference for TCP
Issue #388, item 4	08/xx/2022	Lemire	Add usage requirement for <code>Version</code> format in 3.4.2.17
Issue #386, #387	08/xx/2022	Lemire	Adjust <code>response_requested</code> handling (3.3.1.4) to consider Consumer error situations
Issues #389, #392	8/24/2022	Lemire	Remove Properties target type, per 8/10/2022 working meeting discussion
Issue #369	7/27/2022	Lemire	* Add "comment" as command argument
Issue #393	8/2/2022	Lemire	* Change <code>ArrayOf()</code> to multiplicity where possible
Issue #396	8/xx/2022	Lemire	* Fixed malformed table in 3.4.2.1 * Reordered data types alphabetically
Administrative	9/07/2022	Lemire	Changes for version update, v1.1 to v2.0
Issue #361	9/xx/2022	Lemire	Add explanatory JADN appendix
Create WD01	11/11/2022	Lemire	Create first WD package for v2.0
Issues #350, 365	02/14/2024	Lemire	Address <code>query features</code> response for multiple profiles
Document Quality Updates	05/08/2024	Lemire	Address numerous small fixes throughout

Appendix E. Examples

The content in this section is non-normative.

E.1 Example 1: Device Quarantine

This Command would be used to quarantine a device on the network.

```
{
  "action": "contain",
  "target": {
    "device": {
      "device_id": "9BCE8431AC106FAA3861C7E771D20E53"
    }
  }
}
```

E.2 Example 2: Block Connection

This Command blocks a particular connection within the domain. The standard Actuator Profile `slpf` defines the extended Command Argument `drop_process`. The Response is a simple acknowledgment that was requested in the Command arguments.

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",
    "slpf": {
      "drop_process": "none"
    }
  },
  "profile": "slpf"
}
```

```
}
```

Response:

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

Editor's Note: Replace with an example that does not use "properties".

Appendix F. Notices

Copyright © OASIS Open 2024. All Rights Reserved.

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 implementations of this OASIS Standards Final Deliverable, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this deliverable.]

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

[OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this OASIS Standards Final Deliverable or the extent to which any license under

such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Standards Final Deliverable, 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.]

The name "OASIS" is a trademark of [OASIS](https://www.oasis-open.org/policies-guidelines/trademark/), 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.