Open Command and Control (OpenC2) Language Specification Version 1.0

Committee Specification Draft ~~07~~08 / Public Review Draft ~~01~~02

~~17 October 2018~~

~~Specification URIs~~

04 April 2019

This version:

https://docs.oasis-open.org/openc2/oc2ls/v1.0/csprd02/oc2ls-v1.0-csprd02.md (Authoritative)
https://docs.oasis-open.org/openc2/oc2ls/v1.0/csprd02/oc2ls-v1.0-csprd02.html
https://docs.oasis-open.org/openc2/oc2ls/v1.0/csprd02/oc2ls-v1.0-csprd02.pdf

Previous version:

http://docs.oasis-open.org/openc2/oc2ls/v1.0/csprd01/oc2ls-v1.0-csprd01.md (Authoritative)
http://docs.oasis-open.org/openc2/oc2ls/v1.0/csprd01/oc2ls-v1.0-csprd01.html
http://docs.oasis-open.org/openc2/oc2ls/v1.0/csprd01/oc2ls-v1.0-csprd01.pdf

~~**Previous version:**~~

- ~~(Authoritative)~~

Latest version:

https://docs.oasis-open.org/openc2/oc2ls/v1.0/oc2ls-v1.0.md (Authoritative)
https://docs.oasis-open.org/openc2/oc2ls/v1.0/oc2ls-v1.0.html
https://docs.oasis-open.org/openc2/oc2ls/v1.0/oc2ls-v1.0.pdf

Technical Committee:

OASIS Open Command and Control (OpenC2) TC

Chairs:

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

Editors:

Jason Romano (jdroman@nsa.gov), National Security Agency
Duncan Sparrell (duncan@sfractal.com), sFractal Consulting LLC

~~Additional artifacts:~~

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

- ~~OpenC2 Language Syntax JSON/JADN schema ():~~
  -
  -
- ~~JADN Syntax JSON/JADN schema ():~~

Abstract:

Cyberattacks are increasingly sophisticated, less expensive to execute, dynamic and automated. The provision of ~~cyberdefense~~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:

Citation format:

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

[OpenC2-Lang-v1.0]

Open Command and Control (OpenC2) Language Specification Version 1.0. Edited by Jason Romano and Duncan Sparrell. ~~17 October 2018.~~04 April 2019. OASIS Committee Specification Draft ~~07~~08 / Public Review Draft ~~01~~02. https://docs.oasis-open.org/openc2/oc2ls/v1.0/csprd02/oc2ls-v1.0-csprd02.html. Latest version: https://docs.oasis-open.org/openc2/oc2ls/v1.0/oc2ls-v1.0.html.

Notices

Table of Contents

---

1 Introduction

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

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

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

1.1 IPR Policy

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

1.2 Terminology

This section is normative.

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

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

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

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

1.3 Normative References

[OpenC2-HTTPS-v1.0]

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

[RFC079~~2~~1]

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.

[RFC0793]

Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, https://www.rfc-editor.org/info/rfc793.

[RFC1034]

Mockapetris, P~~. V~~., "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.

[RFC2673]

Crawford, M., *"Binary Labels in the Domain Name System"*, RFC 2673, August 1999, https://tools.ietf.org/html/rfc2673

[RFC3986]

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

[RFC4122]

Leach, P., Mealling, M., and R. Salz, ~~R.,~~ "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.

[RFC4648]

Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, https://www.rfc-editor.org/info/rfc4648.

[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, ~~S.,~~ "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.

[RFC5~~61~~952]

~~Eronen, P., Harrington, D., "Enterprise Number~~Kawamura, S. and M. Kawashima, "A Recommendation for ~~Documentation Use~~IPv6 Address Text

Representation", RFC ~~5612~~5952, DOI 10.17487/RFC5952, August 2010~~9~~, https://www.rfc-editor.org/info/rfc5952.

[RFC6234]

Eastlake 3rd, D~~.,~~. and T. Hansen, ~~T.,~~ "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,~~ S.,~~ "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.

[RFC6838]

Freed, N., Klensin, J., and T. Hansen, ~~T.,~~ "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 I-JSON 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.

[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

1.4 Non-Normative References

[IACD]

M. J. Herring, K. D. Willett, "Active Cyber Defense: A Vision for Real-Time Cyber Defense~~,~~", Journal of Information Warfare, vol. 13, Issue 2, p. 80, April 2014.
Willett, Keith D., "Integrated Adaptive Cyberspace Defense: Secure Orchestration", International Command and Control Research and Technology Symposium, June 2015.

[UML]

"UML Multiplicity and Collections", https://www.uml-diagrams.org/multiplicity.html

1.5 Document Conventions

1.5.1 Naming Conventions

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

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

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

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

1.5.2 Font Colors and Style

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

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

Example:

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

1.6 Overview

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

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

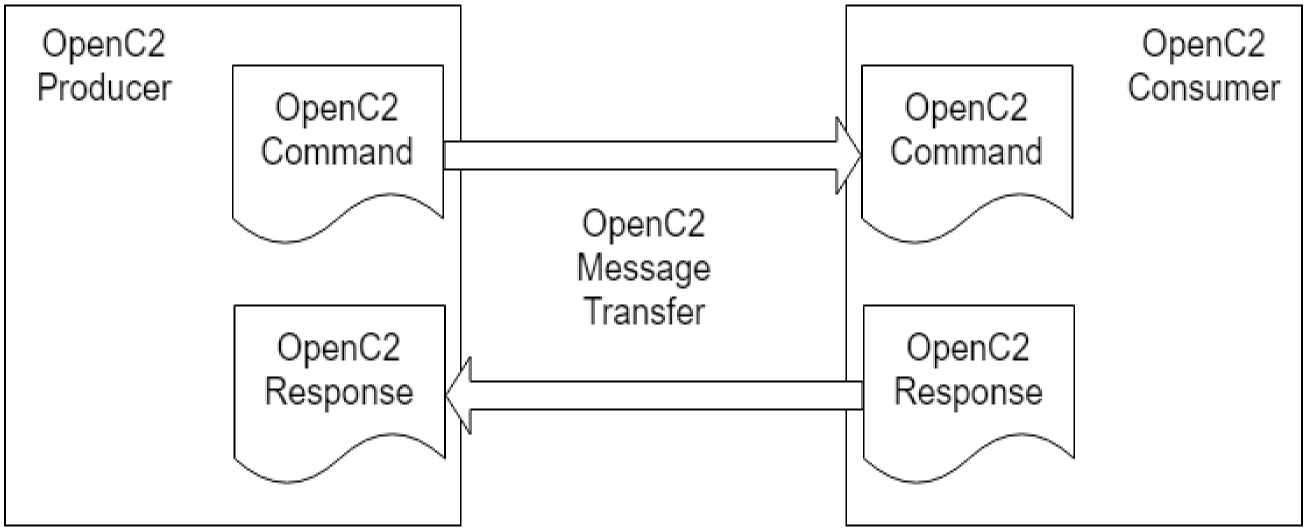


Figure 1-1. OpenC2 Message Exchange

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

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

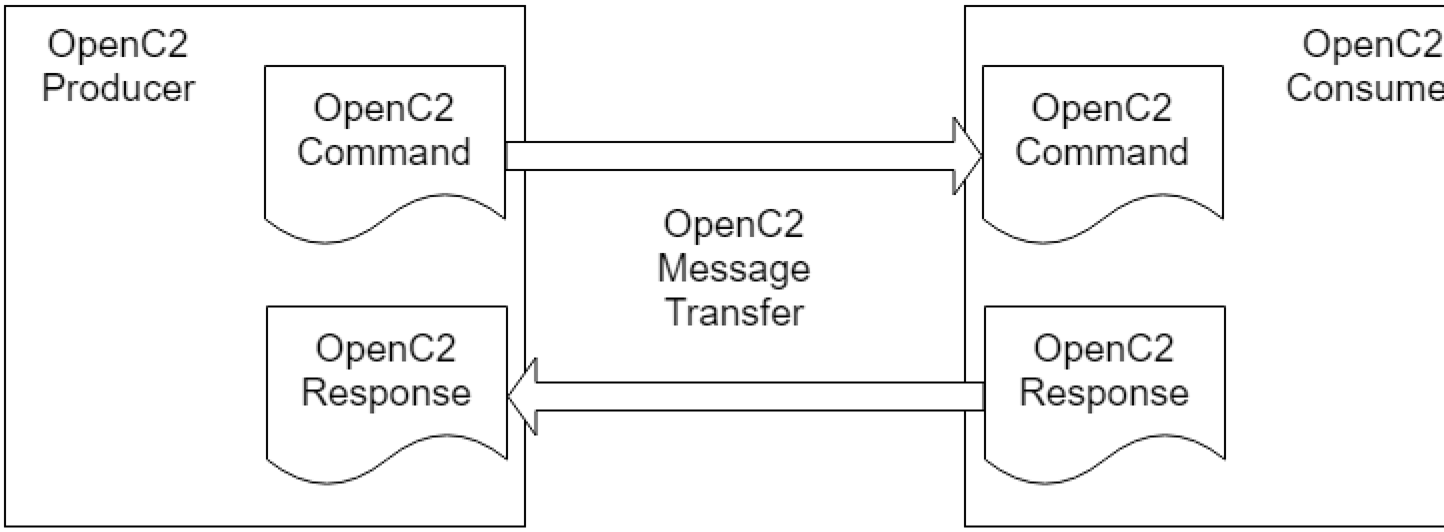
The OpenC2 Language Specification defines a language used to compose ~~m~~Messages for command and control of cyber defense systems and components. A ~~m~~Message consists of a header and a payload (*defined* as a ~~m~~Message body in the OpenC2 Language Specification Version 1.0 and *specified* in one or more ~~a~~Actuator profiles).

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

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

The language defines two payload structures:

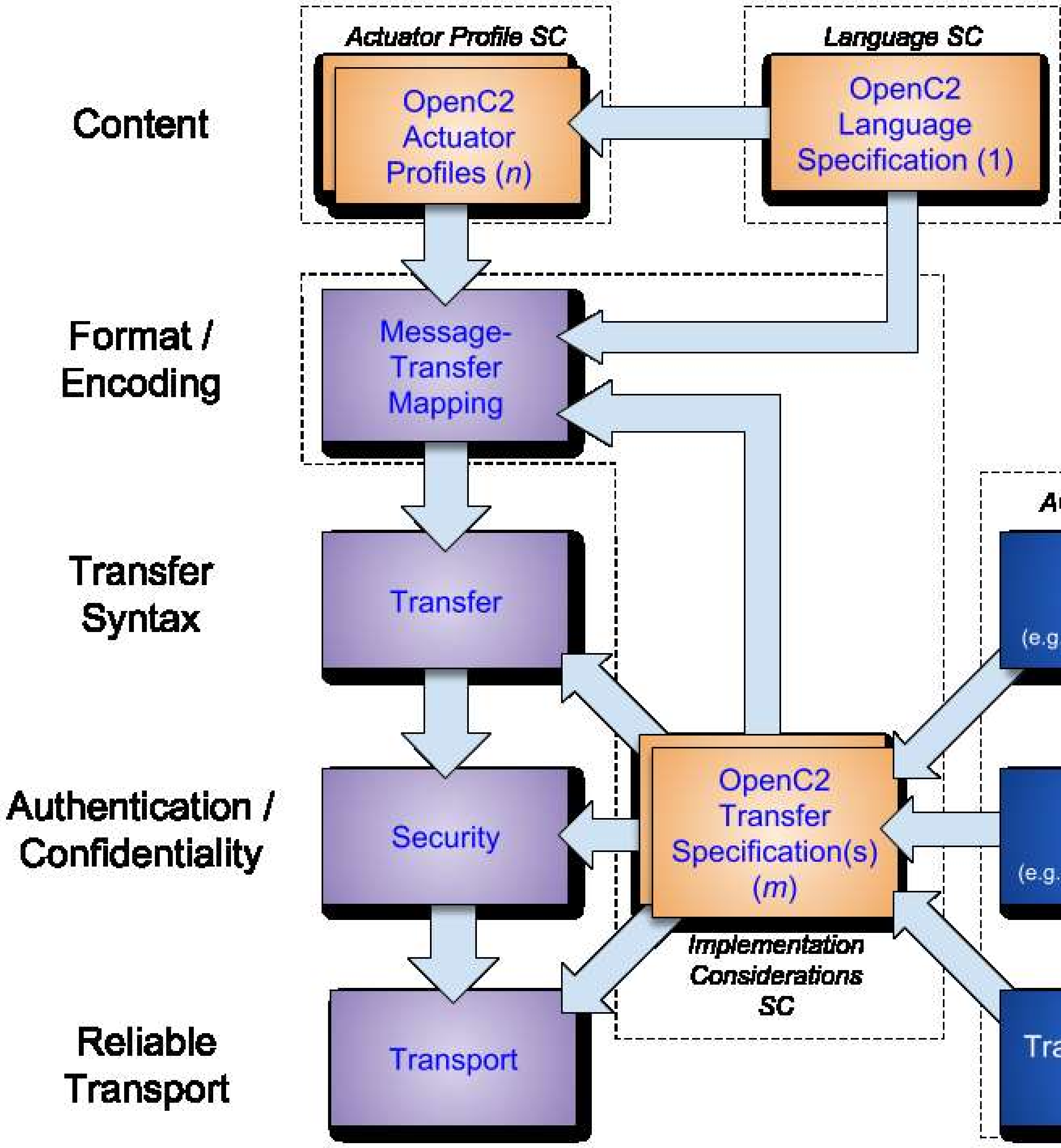
1. **Command**: An instruction from one system known as the ~~OpenC2~~ ~~"~~Producer~~",~~, to one or more systems, the ~~OpenC2 "~~Consumer(s~~)",~~), to act on the content of the ~~c~~Command.
2. **Response**: Any information ~~captured or necessary to send~~sent back to the ~~OpenC2~~ Producer ~~that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



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

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

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


Actuator Profile SC
Language SC
Content
OpenC2 Actuator Profiles (n)
OpenC2 Language Specification (1)
Format / Encoding
Message-Transfer Mapping
Transfer Syntax
Transfer
Authentication / Confidentiality
Security
OpenC2 Transfer Specification(s) (m)
Implementation Considerations SC
Reliable Transport
Transport
(e.g
(e.g.
Tra

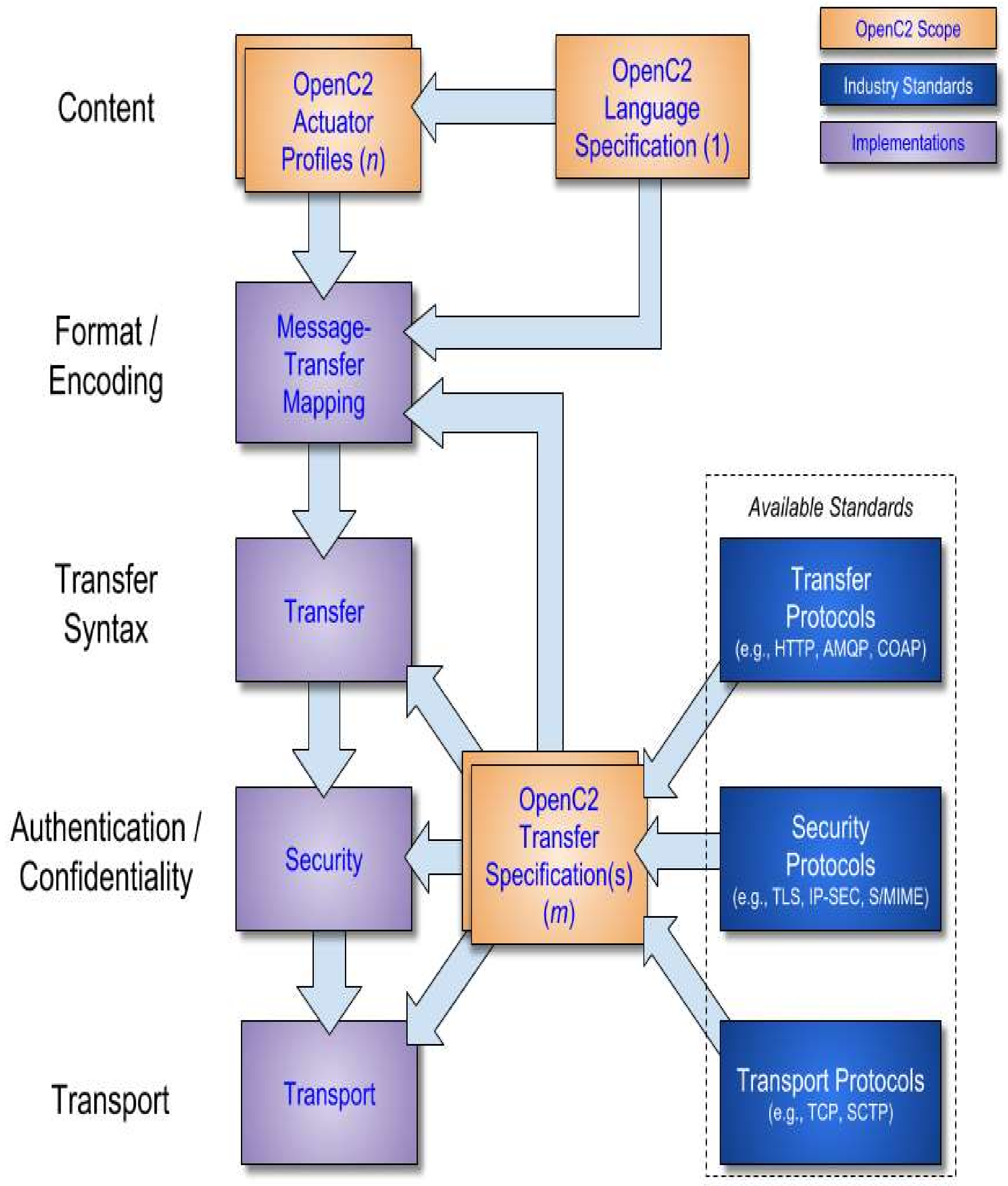

OpenC2 Scope
Industry Standards
Implementations
Content
Format / Encoding
Transfer Syntax
Authentication / Confidentiality
Transport
OpenC2 Actuator Profiles (n)
OpenC2 Language Specification (1)
Message-Transfer Mapping
Transfer
Security
Transport
OpenC2 Transfer Specification(s) (m)
Available Standards
Transfer Protocols (e.g., HTTP, AMQP, COAP)
Security Protocols (e.g., TLS, IP-SEC, S/MIME)
Transport Protocols (e.g., TCP, SCTP)

Figure 1-2. OpenC2 Documentation and Layering Model

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

Table 1-1. OpenC2 Protocol Layers

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

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

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

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

1.7 Goal

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

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

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

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

- **Technology Agnostic:** The OpenC2 language defines a set of abstract atomic cyber defense actions in a platform and ~~product~~implementation agnostic manner
- **Concise:** ~~An OpenC2 command~~A Command is intended to convey only the essential information required to describe the action required and can be represented in a very compact form for communications-constrained environments
- **Abstract:** ~~OpenC2 commands~~Commands and ~~r~~Responses are defined abstractly and can be encoded and transferred via multiple schemes as dictated by the needs of different implementation environments
- **Extensible:** While OpenC2 defines a core set of ~~a~~Actions and ~~t~~Targets for cyber defense, the language is expected to evolve with cyber defense technologies, and permits extensions to accommodate new cyber defense technologies.

1.8 Purpose and Scope

The OpenC2 Language Specification defines the set of components to assemble a complete command and control ~~m~~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 ~~a~~Actions and options that may be used in ~~OpenC2 commands~~Commands
2. the set of ~~t~~Targets and ~~target specifiers~~Target Specifiers
3. a syntax that defines the structure of ~~c~~Commands and ~~r~~Responses
4. a JSON serialization of ~~OpenC2 commands~~Commands and ~~r~~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 ~~c~~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 ~~extensions~~elements applicable to some ~~a~~Actuators, which may be defined in individual ~~a~~Actuator profiles.
2. Alternate serializations of ~~OpenC2 commands~~Commands and ~~r~~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: ~~c~~Command and ~~r~~Response. The ~~c~~Command is sent from a ~~p~~Producer to a ~~c~~Consumer and describes an ~~a~~Action to be performed by an ~~a~~Actuator on a ~~t~~Target. The ~~r~~Response is sent from a ~~c~~Consumer, usually back to the ~~p~~Producer, and is a means to provide information (such as acknowledg~~e~~ment, status, etc.) as a result of a ~~c~~Command.

2.1 OpenC2 Command

The ~~c~~Command describes an ~~a~~Action to be performed on a ~~t~~Target and may include information identifying the ~~a~~Actuator or ~~a~~Actuators that are to execute the ~~c~~Command.

A ~~c~~Command has four main components~~: ACTION, TARGET, ARGUMENTS,~~, two required and two optional. The required components are the Action and the Target. The optional components are Command

Arguments and the Actuator. A Command can also contain an optional Command identifier, if necessary. Section 3.3.1~~ACTUATOR.~~ defines the syntax of an OpenC2 Command.

The following list summarizes the main four components of a ~~c~~Command.

- **~~ACTION~~Action** (required): The task or activity to be performed.

- **~~TARGET~~Target** (required): The object of the action. The ~~ACTION~~Action is performed on the ~~target.~~
  - **~~TARGET-NAME~~** ~~(required): The name of the object~~Target. Properties of the ~~action.~~

  - **~~TARGET-SPECIFIERS~~** ~~(optional): The specifier~~Target, called Target Specifiers, further identifi~~es~~y the ~~t~~Target to some level of precision, such as a specific ~~t~~Target, a list of ~~t~~Targets, or a class of ~~t~~Targets.
  - **~~ARGUMENTS~~Arguments** (optional): Provide additional information on how the command is to be performed, such as date/time, periodicity, duration, etc.

- **~~ACTUATOR~~Actuator** (optional): The ~~ACTUATOR~~Actuator executes the ~~command (the ACTION and TARGET).~~Command. The ~~ACTUATOR type~~ Actuator will be defined within the context of an Actuator Profile.
  - **~~ACTUATOR-NAME~~** ~~(required): The name~~Properties of the ~~set of functions (e.g., "slpf") performed by the actuator, and the name of the profile defining commands applicable to those functions.~~

  - **~~ACTUATOR-SPECIFIERS~~** ~~(optional): The specifier identifies the actuator~~Actuator, called Actuator Specifiers, further identify the Actuator to some level of precision, such as a specific ~~a~~Actuator, a list of ~~a~~Actuators, or a group of ~~a~~Actuators.

The ~~ACTION~~Action and ~~TARGET~~Target components are required and are populated by one of the ~~a~~Actions in Section 3.3.1.1 and the ~~t~~Targets in Section 3.3.1.2. A particular ~~t~~Target may be further refined by the Target type definitions in Section 3.4.1~~one or more TARGET-SPECIFIERS.~~. Procedures to extend the ~~t~~Targets are described in Section 3.~~3.4~~1.5.

~~TARGET-SPECIFIERS provide additional precision to identify the target (e.g., 10.1.2.3) and may include a method of identifying multiple targets of the same type (e.g., 10.1.0.0/16).~~

~~The ARGUMENTS component~~Command Arguments, if present, ~~is populated by one or more 'command arguments' that determine how the command is executed. ARGUMENTS~~ influence the ~~c~~Command by providing information such as tim~~e~~ing, periodicity, duration, or other details on what is to be executed. They can also be used to convey the need for acknowledg~~e~~ment or additional status information about the execution of a ~~c~~Command. The valid

~~ARGUMENTS~~Arguments defined in this specification are in Section 3.3.1.4. Procedures to extend Arguments are described in Section 3.1.5.

An ~~ACTUATOR~~Actuator is an implementation of a cyber defense function that executes the ~~c~~Command. An Actuator Profile is a specification that identifies the subset of ~~ACTIONS, TARGETS~~Actions, Targets and other aspects of this language specification that are ~~mandatory to implement~~required or optional in the context of a particular ~~ACTUATOR~~Actuator. An Actuator Profile may extend the language by defining additional ~~ARGUMENTS, ACTUATOR-SPECIFIERS~~Targets, Arguments, and~~/or TARGETS~~ Actuator Specifiers that are meaningful and possibly unique to the ~~a~~Actuator.

~~The ACTUATOR optionally identifies the entity or entities that are tasked to execute the command. Specifiers for actuators refine the command so that a particular function, system, class of devices, or specific device can be identified.~~

~~The ACTUATOR component may be omitted from a command~~The Actuator may be omitted from a Command and typically will not be included in implementations where the identities of the endpoints are unambiguous or when a high-level effects-based ~~c~~Command is desired and the tactical decisions on how the effect is achieved is left to the recipient.

2.2 OpenC2 Response

The ~~OpenC2~~ Response is a ~~m~~Message sent from the recipient of a ~~c~~Command. Response messages provide acknowledg~~e~~ment, status, results from a query, or other information.

~~The following list summarizes the fields and subfields of an OpenC2 Response.~~

~~• **STATUS** (required): An integer containing~~ At a ~~numerical~~minimum, a Response will contain a status code
~~• **STATUS_TEXT** (optional): A free-form string containing human-readable description of~~ to indicate the result of performing the Command. Additional status text and response ~~status. The string can contain~~ fields optionally provide more ~~detail than is represented by the status code, but does not affect the meaning of the response.~~

~~**RESULTS** (optional): Contains the data or extended status code~~detailed information that ~~was~~ is specific to or requested by the Command. Section 3.3.2~~from~~ defines the syntax of an OpenC2 ~~Command~~Response.

---

3 OpenC2 Language Definition

The content in this section is normative.

3.1 Base Components and Structures

3.1.1 Data Types

~~The syntax of valid OpenC2 messages is defined using an information model constructed from the data types presented here:~~

OpenC2 data types are defined using an abstract notation that is 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	
Any	Anything, used to designate fields with an unspecified value.
Binary	A sequence of octets. Length is the number of octets.
Boolean	~~A logical entity that can have~~An element with one of two values: `true` and `false`.
Integer	A whole number.
Number	A real number.
Null	Nothing, used to designate fields with no value.
String	A sequence of characters~~. Each character must have~~, each of which has a ~~valid~~ Unicode codepoint. Length is the number of characters.
Structures	
Array	An ordered list of unnamed fields~~.~~ with positionally-defined semantics. Each field has ~~an ordinal~~a position, label, and type.

Type	Description
ArrayOf(*vtype*)	An ordered list of ~~unnamed~~ fields ~~of~~with the same ~~type~~semantics. Each field has ~~an ordinal~~a position and ~~the specified~~ type *vtype*.
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 ~~id:name pairs where id~~unnamed integral constants. The API value is an ~~integer. The Enumerated.ID subtype is a set of ids only~~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 ~~named~~ fields. ~~Each~~The API value of each field has an id, ~~name~~label, and type.
MapOf(*ktype*, *vtype*)	An unordered set of keys to values with the same semantics. Each key has key type *ktype* and is mapped to value type *vtype*.
Record	An ordered map from a list of ~~named fields, e.g. a message, record, structure, or row in a table.~~keys iwth positions to values with positionally-defined semantics. Each ~~field~~key has ~~an ordinal~~a position~~,~~ and name, and is mapped to a type. Represents a row in a spreadsheet or database table.

- **API** values do not affect interoperabilty, 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 CBOR formats, exist but are out of scope for this document. Both the format-specific serialization rules in Section 3.1.6 and the format-agnostic type definitions in Section 3.4~~**3.1.2 Derived Data Types**~~

- ~~The following types are defined as value constraints applied to String (text string), Binary (octet string) or~~ are Normative.

Types defined with an ".ID" suffix (Choice.ID, Enumerated.ID, Map.ID) are equivalent to the non-suffixed types except:

1. Field definitions and API values are identified only by ID. The non-normative description may include a suggested name.
2. Serialized values of Enumerated types and keys of Choice/Map types are IDs regardless of serialization format.

OpenC2 type definitions are presented in table format. All table columns except Description are Normative. The Description column is always Non-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*:

~~Integer values. The serialized representation of the base types is specified in , but there are no restrictions on how derived types are represented internally by an implementation.~~

Type Name	~~Base~~Type Definition	Description
~~Domain-Name~~	~~String~~	~~RFC 1034 Section 3.5~~
~~Date-Time~~	~~Integer~~	~~Milliseconds since 00:00:00 UTC, 1 January 1970.~~
~~Duration~~	~~Integer~~	~~Milliseconds.~~
Email-Addr	String (email)	~~RFC 5322 Section 3.4.1~~Email address
~~Identifier~~	~~String~~	~~(TBD rules, e.g., initial alpha followed by alphanumeric or underscore)~~

~~IP-Addr~~	~~Binary~~	~~32 bit IPv4 address or 128 bit IPv6 address~~
~~MAC-Addr~~	~~Binary~~	~~Media Access Control / Extended Unique Identifier address - EUI-48 or EUI-64.~~
~~Port~~	~~Integer~~	~~16 bit RFC 6335 Transport Protocol Port Number~~
~~Request-Id~~	~~Binary~~	~~A value of up to 128 bits~~
~~URI~~	~~String~~	~~RFC 3986~~
~~UUID~~	~~Binary~~	~~128 bit Universal Unique Identifier, RFC 4122 Section 4~~

~~3.1.3 Cardinality~~

For Structure 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 definition is optional (Multiplicity = 0..1), but per the type definition at least one 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
hostname	String	Value must be a hostname as defined in [RFC1034], Section 3.1
uri	String	Value must be a Uniform Resource Identifier (URI) as defined in [RFC3986]
eui	Binary	Value must be an EUI-48 or EUI-64 as defined in [EUI]

3.1.3 Multiplicity

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

- ~~1 Required and not repeatable~~
- ~~0..1 Optional and not repeatable~~
- ~~1..n Required and repeatable~~
- ~~0..n Optional and repeatable~~

~~The cardinality column may also specify a range of sizes, e.g.,:~~

- ~~3..5 Required and repeatable with a minimum of 3 and maximum of 5 values~~

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 used with a Type, multiplicity is enclosed in square brackets, e.g.,:

Type Name	**Base Type**	**Description**
Features	ArrayOf(Feature) [0..10]	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 Derived Enumerations

~~An Enumerated field may be derived ("auto-generated") from the fields of a Choice, Map or Record type by appending ".*" to the type name.~~

It is sometimes useful to reference the fields of a structure definition, for example to list fields that are usable in a particular context, or to read or update the value of a specific field. An instance of a reference can be validated against the set of valid references using either an explicit or a derived Enumerated type. A derived enumeration is created by appending ".Enum" to the type being referenced, and it results in an Enumerated type containing the ID and Name columns of the referenced type.

This example includes a type representing the value of a single picture element ("pixel") in an image, and an operation "SetValue" to set one of the color values of a pixel. It would be possible validate a SetValue operation against an explicit enumeration of the Pixel fields:

Type: ***~~Example-sel (Record~~Pixel (Map)***

ID	Name	Type	#	Description
1	~~targets~~red	Integer~~Target.*~~	1~~..n~~	~~Enumeration auto-generated from a Choice~~
2	green	Integer	1	
3	blue	Integer	1	

Type: SetValue (Record)

ID	Name	Type	#	Description
1	channel	Channel	1	
2	value	Integer	1	

Type: Channel (Enumerated)

ID	Name	Description
1	red	
2	green	
3	blue	

Example **SetValue** operation:

```
{"channel": "green", "value": 95}
```

But it is both easier and more reliable to use a derived enumeration to validate the reference directly against the type being referenced:

Type: SetValue (Record)

ID	Name	Type	#	Description
1	channel	Pixel.Enum	1	
2	value	Integer	1	

3.1.5 Extensions

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

Each Actuator profile has a unique name used to identify the profile document and a short reference called a namespace identifier (NSID). The NSID is used to separate extensions from the core language defined in this specification.

All extension names MUST begin with a namespace identifier followed by a colon (":").

For example, the OASIS standard Stateless Packet Filtering actuator profile has:

- **Unique Name**: http://docs.oasis-open.org/openc2/oc2slpf/v1.0/oc2slpf-v1.0.md
- **NSID**: slpf

The namespace identifier for non-standard extensions MUST be prefixed with "x-".

For example, the fictional, non-standard Superwidget actuator profile has:

- **Unique Name**: http://www.acme.com/openc2/superwidget-v1.0.html
- **NSID**: x-acme

The list of Actions in Section 3.3.1.1 SHALL NOT be extended.

The Targets defined in Section 3.3.1.2 MAY be extended.

Example: In this example Command, the extended Target, `rule_number`, is defined within the Stateless Packet Filtering Profile with the namespace identifier `slpf`.

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

The Arguments defined in Section 3.3.1.4 MAY be extended.

Example: In this example Command, the extended Argument, `direction`, is defined within the Stateless Packet Filtering Profile with the namespace identifier `slpf`.

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

The Actuator property of a Command defined in Section 3.3.1.3 MUST be extended using the namespace identifier as the Actuator name, called an extended Actuator namespace. Actuator Specifiers MUST be defined within the extended Actuator namespace.

Example: In this example Command, the Actuator Specifier `asset_id` is defined within the Stateless Packet Filtering Profile namespace `slpf`.

```
{
    "action": "deny",
    "target": {
        "ipv4_connection": {...}
    },
    "actuator": {
        "slpf": {
            "asset_id": "30"
        }
    }
}
```

The properties of a Response defined in Section 3.3.2 MAY be extended using the namespace identifier as the results name, called an extended results namespace. One or more extended result types MUST be defined with the extended results namespace.

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

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

3.1.6 Serialization

OpenC2 is agnostic of any particular serialization; however, implementations MUST support JSON serialization in accordance with [RFC7493]~~RFC 7493~~ and additional requirements specified in the following table.

JSON Serialization Requirements:

OpenC2 Data Type	JSON Serialization Requirement
Binary	JSON **string** containing Base64url encoding of the binary value as defined in [RFC4648], Section 5~~ of RFC 4648~~.

OpenC2 Data Type	JSON Serialization Requirement
Binary /x	JSON **string** containing Base16 (hex) encoding of a binary value as defined in [RFC4648], Section 8. Note that the Base16 alphabet does not include lower-case letters.
Binary /ipv4-addr	JSON **string** containing the "dotted-quad" representation of an IPv4 address as specified in [RFC2673], Section 3.2.
Binary /ipv6-addr	JSON **string** containing the text representation of an IPv6 address as specified in [RFC5952], Section 4.
Boolean	JSON **true** or **false**
Integer	JSON **number**
Number	JSON **number**
Null	JSON **null**
String	JSON **string**
Array	JSON **array**
Array /ipv4-net	JSON **string** containing the text representation of an IPv4 address range as specified in [RFC4632], Section 3.1.
Array /ipv6-net	JSON **string** containing the text representation of an IPv6 address range as specified in [RFC4291], Section 2.3.
ArrayOf	JSON **array**
Choice	JSON **object** with one member. Member key is the field name.
Choice.ID	JSON **object** with one member. Member key is the integer field id converted to string.

OpenC2 Data Type	JSON Serialization Requirement
Enumerated	JSON **string**
Enumerated.ID	JSON **integer**
Map	JSON **object**. Member keys are field names.
Map.ID	JSON **object**. Member keys are integer field ids converted to strings.
MapOf	JSON **object**. Member keys are as defined in the specified key type.
Record	JSON **object**. Member keys are field names.

3.1.~~5~~6.1 ID and Name Serialization

Instances of Enumerated types and keys for Choice and Map types are serialized as ID values except when using serialization formats intended for human consumption, where Name strings are used instead. Defining a type using ".ID" appended to the base type (e.g., Enumerated.ID, Map.ID) indicates that:

1. Type definitions and application values use only the ID. There is no corresponding name except as an optional part of the description.
2. Instances of Enumerated values and Choice/Map keys are serialized as IDs regardless of serialization format.

3.1.~~5~~6.2 Integer Serialization

For machine-to-machine serialization formats, integers are represented as binary data, e.g., 32 bits, 128 bits. But for human-readable serialization formats (XML and JSON), integers are converted to strings.~~ For example, the~~

Example: The JSON "number" type represents integers and real numbers as decimal strings without quotes~~, e.g., ~~.

```
{ "height": 68.2 ~~}, and as~~}
```

As noted in [RFC7493]~~RFC 7493~~, Section 2.2, a sender cannot expect a receiver to treat an integer with an absolute value greater than 2^^53 as an exact value.

The default representation of Integer types in text serializations is the native integer type for that format, e.g., "number" for JSON. Integer fields with a range larger than the IEEE 754 exact range (e.g., 64, 128, 2048 bit values) are indicated by appending "." or "." *to the type, e.g. Integer.64 or Integer.*. All serializations ensure that large Integer types are transferred exactly, for example in the same manner as Binary types. Integer values support arithmetic operations; Binary values are not intended for that purpose.

3.2 Message

~~As described in Section 1.1, this~~This language specification and one or more ~~a~~Actuator profiles define the content of ~~OpenC2 commands~~Commands and ~~r~~Responses, while transfer specifications define the on-the-wire format of a ~~m~~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~~ and producers~~. 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~~, while other~~ in all messages. Other message elements are not required by this specification but may be required by other ~~documents~~specifications.

Table 3-1. Common Message Elements

Name	Type	Description
content		Message body as specified by content_type and msg_type.
content_type	String	~~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

Name	Type	Description
		1.x, i.e., all versions that are compatible with version 1.0.
msg_type	Message-Type	~~Message-Type.~~ One of **request**, **response**, or **notification**. For the **application/openc2** content_type the request content is an OpenC2-Command and the response content is an OpenC2-Response. OpenC2 does not currently define any notification content.
status	Status-Code	~~Status-Code.~~ Populated with a numeric status code in response messages. Not present in request or notification messages.
request_id	String	~~Request-Id.~~ A unique identifier ~~value of up to 128 bits that is attached to request and response messages. This value is assigned~~created by the ~~sender~~producer and ~~is~~ copied ~~unmodified~~by consumer into all responses, in order to support reference to a particular command, transaction or event chain.
created	Date-Time	~~Date-Time.~~ Creation date/time of the content, the number of milliseconds since 00:00:00 UTC, 1 January 1970.
from	String	~~String.~~ Authenticated identifier of the creator of or authority for execution of a message.
to	ArrayOf(String)	~~ArrayOf(String).~~ Authenticated identifier(s) of the authorized recipient(s) of a message.

Note:

Implementations may use environment variables, private APIs, data structures, class instances, pointers, or other mechanisms to represent messages within the local environment. However the internal representation of a ~~m~~Message does not affect interoperability and is therefore beyond the scope of OpenC2. This means that the ~~m~~Message content is a data structure in whatever form is used within an implementation, not a serialized representation of that structure. Content is the input provided to a serializer or the output of a de-serializer. Msg_type is a three-element enumeration whose protocol representation is defined in each transfer spec, for example as a string, an integer, or a two-bit field. The internal form of enumerations, like content, does not affect interoperability and is therefore unspecified.

Usage Requirements:

- A producer MUST include a request_id in a request message if it expects a response to that request. Absence of a request_id signals consumers that no response is expected.
- The request_id of a request message SHOULD be a Version 4 UUID as specified in [RFC4122], Section 4.4.
- A consumer MUST copy the request_id from a request message into each response to that request.

3.3 Content

The ~~scope~~purpose of this specification is to define the ~~ACTION~~Action and ~~TARGET~~Target portions of ~~an OpenC2 command~~a Command and the common portions of ~~an OpenC2 response.~~a Response. The properties of the ~~OpenC2 command~~Command are defined in Section 3.3.1 and the properties of the ~~r~~Response are defined in Section 3.3.2.

In addition to the ~~ACTION~~Action and ~~TARGET, an OpenC2 command~~Target, a Command has an optional ~~ACTUATOR~~Actuator. Other than identification of namespace identifier, the semantics associated with the ~~ACTUATOR specifiers~~Actuator Specifiers are ~~beyond the scope of this specification. The actuators~~defined in Actuator Profiles. The Actuators and ~~a~~Actuator-specific results contained in a ~~r~~Response are specified in ~~“~~'Actuator Profile Specifications~~”~~' such as StateLess Packet Filtering Profile, Routing Profile etc.

3.3.1 OpenC2 Command

The ~~OpenC2~~ Command de~~scr~~fi~~b~~nes an ~~action~~Action to be performed on a ~~t~~Target.

Type: OpenC2-Command (Record)

ID	Name	Type	#	Description
1	**action**	Action	1	The task or activity to be performed (i.e., the 'verb').

ID	Name	Type	#	Description
2	**target**	Target	1	The object of the ~~a~~Action. The ~~a~~Action is performed on the ~~t~~Target.
3	**args**	Args	0..1	Additional information that applies to the ~~c~~Command.
4	**actuator**	Actuator	0..1	The subject of the ~~a~~Action. The ~~a~~Actuator executes the ~~a~~Action on the ~~t~~Target.
5	**command_id**	String	0..1	An identifier of this Command.

Usage Requirements:

- A Consumer receiving a command with command_id absent and request_id present 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.

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.
6	**deny**	Prevent a certain event or action from completion, such as preventing a flow from reaching a destination or preventing access.

ID	Name	Description
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 ~~t~~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 ~~a~~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 ~~t~~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.

ID	Name	Description
32	**remediate**	Task the recipient to eliminate a vulnerability or attack point.

Usage Requirements:

~~The following actions are under consideration for use in future versions of the Language Specification. Implementers may use these actions with the understanding that they may not be in future versions of the language.~~

- ~~**report** - Task an entity to provide information to a designated recipient~~
- ~~**pause** - Cease operation of a system or activity while maintaining state.~~
- ~~**resume** - Start a system or activity from a paused state~~
- ~~**move** - Change the location of a file, subnet, network, or process~~
- ~~**snapshot** - Record and store the state of a target at an instant in time~~
- ~~**save** - Commit data or system state to memory~~
- ~~**throttle** - Adjust the rate of a process, function, or activity~~
- ~~**delay** - Stop or hold up an activity or data transmittal~~
- ~~**substitute** - Replace all or part of the payload~~
- ~~**sync** - Synchronize a sensor or actuator with other system components~~
- ~~**mitigate** - Task the recipient to circumvent a problem without necessarily eliminating the vulnerability or attack point~~

~~**Usage Requirements:**~~

- Each ~~c~~Command MUST contain exactly one ~~action.~~
- ~~All commands MUST only use actions from this section (either the table or the list)~~
    - ~~Actions~~Action defined in Section 3.3.1.1~~external to this section SHALL NOT be used.~~.

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**	~~Request-Id~~String	1	A reference to a previously issued ~~OpenC2~~ Command.

ID	Name	Type	#	Description
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.
~~16~~9	**features**	Features	1	A set of items used with the query ~~a~~Action to determine an ~~a~~Actuator's capabilities.
10	**file**	File	1	Properties of a file.
~~11~~13	**~~ip_addr~~ipv4_net**	~~IP-Addr~~IPv4-Net	1	An IPv4 address ~~(either version 4 or version 6).~~range including CIDR prefix length.
14	**ipv6_net**	IPv6-Net	1	An IPv6 address range including prefix length.
15	**ipv4_connection**	IPv4-Connection	1	A ~~network connection that originates from a~~5-tuple of source and ~~is addressed to a~~ destination~~. Source~~ IPv4 address ranges, source and destination ~~addresses may be either IPv4 or IPv6; both should be the same version~~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
~~13~~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
~~17~~18	**process**	Process	1	Common properties of an instance of a computer program as executed on an operating system.
25	**properties**	Properties	1	Data attribute associated with an ~~a~~Actuator
19	**uri**	URI	1	A uniform resource identifier (URI).
~~1000~~	~~**extension**~~	~~PE-Target~~	~~1~~	~~Targets defined in a Private Enterprise extension profile.~~
~~1001~~	~~**extension_unr**~~	~~Unr-Target~~	~~1~~	~~Targets defined in an Unregistered extension profile~~
~~1024~~	~~**slpf**~~	~~slpf:Target~~	~~1~~	~~**Example Target Extension**: Targets defined in the Stateless Packet Filter profile~~

Usage Requirements:

The ~~following targets are under consideration for use in future versions of the Language Specification. Implementers may use these targets with the understanding that they may not be in future versions of the language.~~

- ~~directory~~
- ~~disk~~
- ~~disk_partition~~
- ~~email_message~~
- ~~memory~~
- ~~software~~
- ~~user_account~~
- ~~user_session~~
- ~~volume~~
- ~~windows_registry_key~~
- ~~x509_certificate~~

~~**Usage Requirements:**~~

- ~~The TARGET~~`target` field in ~~an OpenC2~~a Command MUST contain exactly one type of ~~t~~Target (e.g.~~. ip_addr~~., ipv4_net).

3.3.1.3 Actuator

Type: Actuator (Choice)

ID	Name	Type	#	Description
~~1000~~	~~**extension**~~	~~PE-Specifiers~~	~~0..1~~	~~Specifiers defined in a Private Enterprise extension profile.~~
~~1001~~	~~**extension_unr**~~	~~Unr-Specifiers~~	~~0..1~~	~~Specifiers defined in an Unregistered extension profile~~
1024	**slpf**	slpf:Actuator	1	**Example**: Actuator Specifiers defined in the Stateless Packet Filtering Profile

3.3.1.4 Command Arguments

Type: Args (Map)

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
4	**response_requested**	Response-Type	0..1	The type of Response required for the Action: `none`, `ack`, `status`, `complete`.

Usage Requirements:

~~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~~
4	~~**response_requested**~~	~~Response-Type~~	~~0..1~~	~~The type of response required for the action: `none`, `ack`, `status`, `complete`.~~
~~1000~~	~~**extension**~~	~~PE-Args~~	~~0..1~~	~~Command arguments defined in a Private Enterprise extension profile~~
~~1001~~	~~**extension_unr**~~	~~Unr-Args~~	~~0..1~~	~~Command arguments defined in an Unregistered extension profile~~

~~**Usage Requirements:**~~

- ~~When~~`start_time`, `end_time`, `duration`:
    - If none are specified, then `start_time` is now, `end_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 `end_time` = `start_time` + `duration`.
    - If only `start_time` is specified then `end_time` is never and `duration` is infinity.
    - If only `end_time` is specified then `start_time` is now and `duration` is derived.
    - If only `duration` is specified then `start_time` is now and `end-time` is derived.
- `response_requested`:
    - If `response_requested` is specified as `none` then the Consumer SHOULD NOT send a Response.
    - 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 ~~contained in an OpenC2 Command, a~~specified then the Consumer ~~MUST~~SHOULD respond ~~in the same manner~~ as ~~{"response_requested": "~~if `complete`~~"}.~~ was specified.

3.3.2 OpenC2 Response

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 ~~r~~Response status
3	**strings**	String	0~~..n~~..*	Generic set of string values
4	**ints**	Integer	0~~..n~~..*	Generic set of integer values
5	**~~kvps~~results**	~~KVP~~MapOf(String, Any)	0~~..n~~..*	~~Generic set of key:value pairs~~Generic Map of key:value pairs (keys are strings, and values are any valid JSON value). A JSON value can be an object, array, number, string, true, false, or null, as defined by ECMA-404.
6	**versions**	Version	0~~..n~~..*	List of OpenC2 language versions supported by this ~~a~~Actuator

ID	Name	Type	#	Description
7	**profiles**	~~jadn:Uname~~ArrayOf(Nsid)	0~~..n~~..*	List of profiles supported by this ~~a~~Actuator
~~8~~	~~**schema**~~	~~jadn:Schema~~	~~0..1~~	~~Syntax of the OpenC2 language elements supported by this actuator~~
9	**pairs**	Action-Targets	0~~..n~~..*	List of targets applicable to each supported ~~a~~Action
10	**rate_limit**	Number	0..1	Maximum number of requests per minute supported by design or policy
~~1000~~	~~**extension**~~	~~PE-Results~~	~~0..1~~	~~Response data defined in a Private Enterprise extension profile~~
~~1001~~	~~**extension_unr**~~	~~Unr-Results~~	~~0..1~~	~~Response data defined in an unregistered extension profile~~

Example:

```
{
    "status": 200,
    "status_text": "All endpoints successfully updated",
    "strings": ["wd-394", "sx-2497"]
}
```

Usage Requirements:

- All Responses MUST contain a status.

- ~~Responses MAY contain status_text and/or results.~~

3.3.2.1 ~~OpenC2~~ Response Status Code

Type: Status-Code (Enumerated.ID)

ID	Description
102	**Processing** - an interim ~~r~~Response used to inform the ~~p~~Producer that the ~~c~~Consumer has accepted the request but has not yet completed it.
200	**OK** - the request has succeeded.
~~301~~	~~**Moved Permanently** - the target resource has been assigned a new permanent URI.~~
400	**Bad Request** - the ~~c~~Consumer cannot process the request due to something that is perceived to be a ~~p~~Producer error (e.g., malformed request syntax).
401	**Unauthorized** - the request lacks valid authentication credentials for the target resource or authorization has been refused for the submitted credentials.
403	**Forbidden** - the ~~c~~Consumer understood the request but refuses to authorize it.
404	**Not Found** - the ~~c~~Consumer has not found anything matching the request.
500	**Internal Error** - the ~~c~~Consumer encountered an unexpected condition that prevented it from fulfilling the request.
501	**Not Implemented** - the ~~c~~Consumer does not support the functionality required to fulfill the request.
503	**Service Unavailable** - the ~~c~~Consumer is currently unable to handle the request due to a temporary overloading or maintenance of the ~~c~~Consumer.

~~3.3.3 Imported Data~~

~~In addition to the targets, actuators, arguments, and other language elements defined in this specification, OpenC2 messages may contain data objects imported from other specifications and/or custom data objects defined by the implementers. The details are specified in a data profile which contains:~~

1. ~~a prefix indicating the origin of the imported data object is outside OpenC2:~~
    - ~~x_ (profile)~~
2. ~~a unique name for the specification being imported, e.g.:~~

    - ~~For shortname `x_kmipv2.0` the full name would be `oasis-open.org/openc2/profiles/kmip-v2.0`,~~
    - ~~For shortname `x_sfslpf` the full name would be `sfractal.com/slpf/v1.1/x_slpf-profile-v1.1`~~
3. ~~a namespace identifier (nsid) - a short reference, e.g., `kmipv2.0`, to the unique name of the specification~~
4. ~~a list of object identifiers imported from that specification, e.g., `Credential`~~
5. ~~a definition of each imported object, either referenced or contained in the profile~~
6. ~~conformance requirements for implementations supporting the profile~~

~~The data profile itself can be the specification being imported or the data profile can reference an existing specification. In the example above, the data profile created by the OpenC2 TC to represent KMIP could have a unique name of `oasis-open.org/openc2/profiles/kmip-v2.0`. The data profile would note that it is derived from the original specification `oasis-open.org/kmip/spec/v2.0/kmip-spec-v2.0`. In the example for shortname `x_sfslpf`, the profile itself could be defined in a manner directly compatible with OpenC2 and would not reference any other specification.~~

~~An imported object is identified by namespace identifier and object identifier. While the data profile may offer a suggested nsid, the containing schema defines the nsids that it uses to refer to objects imported from other specifications:~~

```
import oasis-open.org/openc2/profiles/kmip-v2.0 as x_kmip_2.0
```

~~An element using an imported object identifies it using the nsid:~~

```
{
        "target": {
        "x_kmip_2.0": {
                {"kmip_type": "json"},
                {"operation": "RekeyKeyPair"},
                {"name": "publicWebKey11DEC2017"}
        }
    }
}
```

~~A data profile can define its own schema for imported objects, or it can reference content as defined in the specification being imported. Defining an abstract syntax allows imported objects to be represented in the same format as the containing object. Referencing content directly from an imported specification results in it being treated as an opaque blob if the imported and containing formats are not the same (e.g., an XML or TLV object imported into a JSON OpenC2 command, or a STIX JSON object imported into a CBOR OpenC2 command).~~

~~The OpenC2 Language MAY be extended using imported data objects for TARGET, TARGET_SPECIFIER, ACTUATOR, ACTUATOR_SPECIFIER, ARGUMENTS, and RESULTS. The list of ACTIONS in Section 3.2.1.2 SHALL NOT be extended.~~

~~3.3.4 Extensions~~

~~Organizations may extend the functionality of OpenC2 by defining organization-specific profiles. OpenC2 defines two methods for defining organization-specific profiles: using a registered namespace or an unregistered namespace. Organizations wishing to create non-standardized OpenC2 profiles SHOULD use a registered Private Enterprise Number namespace. Private Enterprise Numbers are managed by the Internet Assigned Numbers Authority (IANA) as described in RFC 5612, for example:~~

- ~~32473~~
    - ~~Example Enterprise Number for Documentation Use~~
        - ~~See [RFC5612]~~
            - ~~iana&iana.org~~

~~OpenC2 contains four predefined extension points to support registered private enterprise profiles: PE-Target, PE-Specifiers, PE-Args, and PE-Results. An organization can develop a profile that defines custom types, create an entry for their organization's namespace under each extension point used in the profile, and then use their custom types within OpenC2 commands and responses.~~

~~By convention ID values of 1000 and above within OpenC2-defined data types are namespace identifiers, although there is no restriction against assigning non-namespaced IDs in that range.~~

~~This is an example target from a registered profile containing a "lens" extension defined by the organization with IANA Private Enterprise Number 32473. This hypothetical target might be used with the "set" action to support an IoT camera pan-tilt-zoom use case. This example is for illustrative purposes only and MUST NOT use this in actual implementations.~~

```
~~{~~
~~    "target": {~~
~~        "extension": {~~
~~            "32473": {~~
~~                "lens": {"focal_length": 240, "aperture": "f/1.6"}~~
~~            }~~
~~        }~~
~~    }~~
~~}~~
```

~~This is an example of the same target from a profile defined by an organization that has not registered a Private Enterprise Number with IANA. This example is for illustrative purposes only and MUST NOT use this in actual implementations.~~

```
~~{~~
~~    "target": {~~
~~        "unregistered": {~~
~~            "x-foo.com": {~~
~~                "lens": {"focal_length": 240, "aperture": "f/1.6"}~~
~~            }~~
~~        }~~
~~    }~~
~~}~~
```

~~Using DNS names provides collision resistance for names used in x- namespaces, but the corresponding IDs are not coordinated through a registration process and are subject to collisions.~~

~~OpenC2 implementations MAY support registered and unregistered extension profiles regardless of whether those profiles are listed by OASIS. Implementations MUST NOT use the "Example" registered extension entries shown below, and MAY use one or more actual registered extensions by replacing the example entries.~~

~~3.3.4.1 Private Enterprise Target~~

~~Because target is a required element, implementations receiving an OpenC2 Command with an unsupported target type MUST reject the command as invalid.~~

~~Type: PE-Target (Choice.ID)~~

~~ID~~	~~Type~~	~~#~~	~~Description~~
~~32473~~	~~32473:Target~~	~~1~~	~~"Example": Targets defined in the Example Inc. extension profile~~

~~3.3.4.2 Private Enterprise Specifiers~~

~~The behavior of an implementation receiving an OpenC2 Command with an unsupported actuator type is undefined. It MAY ignore the actuator field or MAY reject the command as invalid.~~

~~Type: PE-Specifiers (Choice.ID)~~

~~ID~~	~~Type~~	~~#~~	~~Description~~
~~32473~~	~~32473:Specifiers~~	~~1~~	~~"Example": Actuator Specifiers defined in the Example Inc. extension profile~~

~~3.3.4.3 Private Enterprise Command Arguments~~

~~The behavior of an implementation receiving an OpenC2 Command with an unsupported arg type is undefined. It MAY ignore the unrecognized arg or MAY reject the command as invalid.~~

~~***Type: PE-Args (Map.ID)***~~

~~ID~~	~~Type~~	~~#~~	~~Description~~
~~32473~~	~~32473:Args~~	~~1~~	~~"Example": Command Arguments defined in the Example Inc. extension profile~~

~~**3.3.4.4 Private Enterprise Results**~~

~~The behavior of an implementation receiving an OpenC2 Response with an unsupported results type is undefined. An unrecognized response has no effect on the OpenC2 protocol but implementations SHOULD log it as an error.~~

~~***Type: PE-Results (Map.ID)***~~

~~ID~~	~~Type~~	~~#~~	~~Description~~
~~32473~~	~~32473:Results~~	~~1~~	~~"Example": Results defined in the Example Inc. extension profile~~

3.4 Type Definitions

3.4.1 Target Types

3.4.1.1 Artifact

Type: Artifact (Record)

ID	Name	Type	#	Description
1	**mime_type**	String	0..1	Permitted values specified in the IANA Media Types registry, [RFC6838]~~RFC 6838~~
2	**payload**	Payload	0..1	Choice of literal content or URL
3	**hashes**	Hashes	0..1	Hashes of the payload content

3.4.1.~~3~~2 Device

Type: Device (Map)

ID	Name	Type	#	Description
1	**hostname**	Hostname	1	A hostname that can be used to connect to this device over a network
2	**description**	String	0..1	A human-readable description of the purpose, relevance, and/or properties of this device
3	**device_id**	String	0..1	An identifier that refers to this device within an inventory or management system

3.4.1.~~4~~3 Domain Name

Type Name	~~Base~~ Type Definition	Description
Domain-Name	String (hostname)	[RFC1034]~~RFC 1034, section~~, Section 3.5

3.4.1.~~5~~4 Email Address

Type Name	~~Base~~ Type Definition	Description
Email-Addr	String (email)	Email address, [RFC5322]~~RFC 5322, section~~, Section 3.4.1

3.4.1.~~6~~5 Features

Type Name	~~Base~~ Type Definition	Description
Features	ArrayOf(Feature) [0..10]	An array of zero to ten names used to query an ~~a~~Actuator 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.

Usage Notes:

- A Producer may send a query command containing an empty list of features 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 will return an empty response to this command, minimizing the amount of traffic used to perform heartbeat / keep-alive functions.

3.4.1.~~7~~6 File

Type: File (Map)

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

3.4.1.~~8 IP~~7 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).

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.

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

~~Type Name~~ID	~~Base~~ Type	#	Description
1	IPv4-Addr	~~Binary~~1	~~32 bit IPv4~~ ipv4-address ~~or 128 bit IPv6 address~~as defined in [RFC0791]
2	Integer	0..1	CIDR prefix-length. If omitted, refers to a single host address.

3.4.1.~~9 IP~~8 IPv4 Connection

Type: IPv4-Connection (Record)

ID	Name	Type	#	Description
1	**src_addr**	~~IP-Addr~~IPv4-Net	0..1	~~ip_addr of source, could be ipv4 or ipv6 - see ip_addr section~~IPv4 source address range
2	**src_port**	Port	0..1	source service per [RFC6335]~~RFC 6335~~
3	**dst_addr**	~~IP-Addr~~IPv4-Net	0..1	~~ip_addr of~~IPv4 destination~~, could be ipv4 or ipv6 - see ip_addr section~~ address range
4	**dst_port**	Port	0..1	destination service per [RFC6335]~~RFC 6335~~
5	**protocol**	L4-Protocol	0..1	layer 4 protocol (e.g., TCP) - see Section 3.4.2.9~~l4_protocol section~~

~~**Usage Requirements:**~~

3.4.1.9 IPv6 Address Range

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

ID	Type	#	Description

- ~~src_addr and dst_addr MUST be the same version (ipv4 or ipv6) if both are present.~~

1	IPv6-Addr	1	ipv6-address as defined in [RFC8200]
2	Integer	0..1	prefix-length. If omitted, refers to a single host address.

3.4.1.10 IPv6 Connection

Type: IPv6-Connection (Record)

ID	Name	Type	#	Description
1	**src_addr**	IPv6-Net	0..1	IPv6 source address range
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 3.4.2.9

Editor's Note: Renumber

3.4.1.11 MAC Address

~~Type Name~~	~~Base Type~~	~~Description~~
~~**MAC-Addr**~~	~~Binary~~	~~Media Access Control / Extended Unique Identifier address - EUI-48 or EUI-64.~~

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.~~11~~12 Process

Type: Process (Map)

ID	Name	Type	#	Description
1	**pid**	Integer	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**	Process	0..1	Process that spawned this one
6	**command_line**	String	0..1	The full command line invocation used to start this process, including all arguments

3.4.1.~~12~~13 Properties

Type Name	~~Base~~ Type Definition	Description
Properties	ArrayOf(String)	A list of names that uniquely identify properties of an ~~a~~Actuator.

3.4.1.~~13~~14 URI

Type Name	~~Base~~ Type Definition	Description
URI	String (uri)	Uniform Resource Identifier

3.4.2 Data Types

3.4.2.1 ~~Request Identifier~~Action-Targets

Type Name	~~Base~~ Type Definition	Description
Action-Targets~~Request-Id~~	~~Binary~~MapOf(Action, Targets)	~~A value~~Map of ~~up~~each action supported by this actuator to ~~128 bits~~the list of targets applicable to that ~~uniquely identifies a particular command~~action.

~~**3.4.2.2 Date-Time**~~

Type Name	~~Base~~ Type Definition	Description
Targets~~Date-Time~~	~~Integer~~ArrayOf(Target.Enum) [1..*]	~~Milliseconds since 00:00:00 UTC, 1 January 1970~~List of Target fields

3.4.2.2 Date-Time

Type Name	Type Definition	Description
Date-Time	Integer	Date and Time

Usage Requirements:

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

3.4.2.3 Duration

Type Name	~~Base~~ Type Definition	Description
Duration	Integer	~~Milliseconds~~A length of time

Usage Requirements:

- Value is a number of milliseconds

3.4.2.4 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 Actuator
2	**profiles**	List of profiles supported by this Actuator
3	**pairs**	List of supported Actions and applicable Targets
4	**rate_limit**	Maximum number of requests per minute supported by design or policy

3.4.2.5 Hashes

Type: Hashes (Map)

ID	Name	Type	#	Description
1	**md5**	Binary /x	0..1	MD5 hash as defined in [RFC1321]~~RFC 1321~~
2	**sha1**	Binary /x	0..1	SHA1 hash as defined in [RFC6234]~~RFC 6234~~
3	**sha256**	Binary /x	0..1	SHA256 hash as defined in [RFC6234]~~RFC 6234~~

3.4.2.~~5~~6 Hostname

Type Name	~~Base~~ Type Definition	Description
Hostname	String (hostname)	~~A legal~~ Internet host name as specified in [RFC1123]~~RFC 1123~~

3.4.2.7 IPv4 Address

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

3.4.2.8 IPv6 Address

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

3.4.2.9 L4 Protocol

Value of the protocol (IPv4) or next header (IPv6) field in an IP packet. Any IANA value, [RFC5237]~~RFC 5237~~

Type: L4-Protocol (Enumerated)

ID	Name	Description
1	**icmp**	Internet Control Message Protocol - [RFC0792]~~RFC 792~~
6	**tcp**	Transmission Control Protocol - [RFC0793]~~RFC 793~~
17	**udp**	User Datagram Protocol - [RFC0768]~~RFC 768~~
132	**sctp**	Stream Control Transmission Protocol - [RFC4960]~~RFC 4960~~

3.4.2.10 Namespace Identifier

Type Name	Base Type	Description
Nsid	String [1..16]	A short identifier that refers to a namespace.

~~8~~

3.4.2.11 Payload

Type: Payload (Choice)

ID	Name	Type	#	Description
1	**bin**	Binary	1	Specifies the data contained in the artifact

ID	Name	Type	#	Description
2	**url**	URI	1	MUST be a valid URL that resolves to the un-encoded content

3.4.2.~~9~~12 Port

Type Name	~~Base~~ Type Definition	Description
Port	Integer [0..65535]	Transport Protocol Port Number, [RFC6335]~~RFC 6335~~

3.4.2.~~10 Feature~~

~~Specifies the results to be returned from a query features command.~~

13 Response-Type~~: Feature~~

***Type: Response-Type* (Enumerated)**

ID	Name	Description
0	**none**	No response
1	**~~version~~ack**	~~List of OpenC2 Language versions supported by this actuator~~Respond when Command received
2	**status~~profiles~~**	~~List of profiles supported by this actuator~~Respond with progress toward Command completion
3	**complete~~schema~~**	~~Definition of the command syntax supported by this actuator~~Respond when all aspects of Command completed
~~4~~	~~**pairs**~~	~~List of supported actions and applicable targets~~
~~5~~	~~**rate_limit**~~	~~Maximum number of requests per minute supported by design or policy~~

3.4.2.~~11 Response-Type~~14 Version

~~*Type: Response-Type (Enumerated)*~~

~~ID~~Type Name	~~Name~~Type Definition	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~~

~~3.4.2.12 Version~~

~~Type Name~~	~~Base Type~~	~~Description~~
Version	String	Major.Minor version number

~~3.4.2.14 Key-Value Pair~~

~~*Type: KVP (Array)*~~

~~ID~~	~~Type~~	~~#~~	~~Description~~
~~1~~	~~String~~	~~1~~	~~"key": name of this item~~
~~2~~	~~String~~	~~1~~	~~"value": string value of this item~~

~~3.4.2.15 Action-Targets Array~~

~~*Type: Action-Targets (Array)*~~

~~ID~~	~~Type~~	~~#~~	~~Description~~
~~1~~	~~Action~~	~~1~~	~~An action supported by this actuator.~~
~~2~~	~~Target.*~~	~~1..n~~	~~List of targets applicable to this action. The targets are enumerated values derived from the set of Target types.~~

~~3.4.3 Schema Syntax~~

~~3.4.3.1 Schema~~

~~Type: Schema (Record)~~

~~ID~~	~~Name~~	~~Type~~	~~#~~	~~Description~~
~~1~~	**~~meta~~**	~~Meta~~	~~1~~	~~Information about this schema module~~
~~2~~	**~~types~~**	~~Type~~	~~1..n~~	~~Types defined in this schema module~~

~~3.4.3.1 Meta~~

~~Meta-information about this schema~~

~~Type: Meta (Map)~~

~~ID~~	~~Name~~	~~Type~~	~~#~~	~~Description~~
~~1~~	**~~module~~**	~~Uname~~	~~1~~	~~Unique name~~
~~2~~	**~~title~~**	~~String~~	~~0..1~~	~~Title~~
~~3~~	**~~version~~**	~~String~~	~~0..1~~	~~Patch version (module includes major.minor version)~~
~~4~~	**~~description~~**	~~String~~	~~0..1~~	~~Description~~
~~5~~	**~~imports~~**	~~Import~~	~~0..n~~	~~Imported schema modules~~

~~6~~	~~**exports**~~	~~Identifier~~	~~0..n~~	~~Data types exported by this module~~
~~7~~	~~**bounds**~~	~~Bounds~~	~~0..1~~	~~Schema-wide upper bounds~~

~~3.4.3.2 Import~~

~~***Type: Import (Array)***~~

~~ID~~	~~Type~~	~~#~~	~~Description~~
~~1~~	~~Nsid~~	~~1~~	~~**nsid** - A short local identifier (namespace id) used within this module to refer to the imported module~~
~~2~~	~~Uname~~	~~1~~	~~**uname** - Unique name of the imported module~~

~~3.4.3.3 Bounds~~

~~Schema-wide default upper bounds. If included in a schema, these values override codec default values but are limited to the codec hard upper bounds. Sizes provided in individual type definitions override these defaults.~~

~~***Type: Bounds (Array)***~~

~~ID~~	~~Type~~	~~#~~	~~Description~~
~~1~~	~~Integer~~	~~1~~	~~**max_msg** - Maximum serialized message size in octets or characters~~
~~2~~	~~Integer~~	~~1~~	~~**max_str** - Maximum text string length in characters~~
~~3~~	~~Integer~~	~~1~~	~~**max_bin** - Maximum binary string length in octets~~
~~4~~	~~Integer~~	~~1~~	~~**max_fields** - Maximum number of elements in ArrayOf~~

~~3.4.3.4 Type~~

~~Definition of a data type.~~

~~***Type: Type (Array)***~~

~~ID~~	~~Type~~	~~#~~	~~Description~~
~~1~~	~~Identifier~~	~~1~~	~~**tname** - Name of this data type~~
~~2~~	~~JADN-Type.*~~	~~1~~	~~**btype** - Base type. Enumerated value derived from the list of JADN data types.~~
~~3~~	~~Option~~	~~1..n~~	~~**topts** - Type options~~
~~4~~	~~String~~	~~1~~	~~**tdesc** - Description of this data type~~
~~5~~	~~JADN-Type.&2~~	~~1..n~~	~~**fields** - List of fields for compound types. Not present for primitive types.~~

~~3.4.3.5 JADN Type~~

~~Field definitions applicable to the built-in data types (primitive and compound) used to construct a schema.~~

~~***Type: JADN-Type (Choice)***~~

~~ID~~	~~Name~~	~~Type~~	~~#~~	~~Description~~
~~1~~	~~Binary~~	~~Null~~		~~Octet (binary) string~~
~~2~~	~~Boolean~~	~~Null~~		~~True or False~~
~~3~~	~~Integer~~	~~Null~~		~~Whole number~~
~~4~~	~~Number~~	~~Null~~		~~Real number~~
~~5~~	~~Null~~	~~Null~~		~~Nothing~~
~~6~~	~~String~~	~~Null~~		~~Character (text) string~~
~~7~~	~~Array~~	~~FullField~~		~~Ordered list of unnamed fields~~

~~ID~~	~~Name~~	~~Type~~	~~#~~	~~Description~~
~~8~~	~~ArrayOf~~	~~Null~~		~~Ordered list of fields of a specified type~~
~~9~~	~~Choice~~	~~FullField~~		~~One of a set of named fields~~
~~10~~	~~Enumerated~~	~~EnumField~~		~~One of a set of id:name pairs~~
~~11~~	~~Map~~	~~FullField~~		~~Unordered set of named fields~~
~~12~~	~~Record~~	~~FullField~~		~~Ordered list of named fields~~

~~3.4.3.6 Enum Field~~

~~Item definition for Enumerated types~~

~~Type: EnumField (Array)~~

~~ID~~	~~Type~~	~~#~~	~~Description~~
~~1~~	~~Integer~~	~~1~~	~~Item ID~~
~~2~~	~~Identifier~~	~~1~~	~~Item name~~
~~3~~	~~String~~	~~1~~	~~Item description~~

~~3.4.3.7 Full Field~~

~~Field definition for compound types Array, Choice, Map, Record~~

~~Type: FullField (Array)~~

~~ID~~	~~Type~~	~~#~~	~~Description~~
~~1~~	~~Integer~~	~~1~~	~~Field ID or ordinal position~~

~~ID~~	~~Type~~	~~#~~	~~Description~~
~~2~~	~~Identifier~~	~~1~~	~~Field name~~
~~3~~	~~Identifier~~	~~1~~	~~Field type~~
~~4~~	~~Options~~	~~1~~	~~Field options. This field is an empty array (not omitted) if there are none.~~
~~5~~	~~String~~	~~1~~	~~Field description~~

~~3.4.3.8 Identifier~~

~~Type Name~~	~~Base Type~~	~~Description~~
~~Identifier~~	~~String~~	~~A string beginning with an alpha character followed by zero or more alphanumeric~~

~~3.4.3.9 Nsid~~

~~Type Name~~	~~Base Type~~	~~Description~~
~~Nsid~~	~~String~~	~~Namespace ID - a short identifier, max length 8 characters~~

~~3.4.3.10 Uname~~

~~Type Name~~	~~Base Type~~	~~Description~~
~~Uname~~	~~String~~	~~Unique name (e.g., of a schema) - typically a set of Identifiers separated by forward slashes~~

~~3.4.3.11 Options~~

~~Type Name~~	~~Base Type~~	~~Description~~
~~**Options**~~	~~ArrayOf(Option)~~	~~An array of zero to ten option strings.~~

~~**3.4.3.12 Option**~~

~~Type Name~~	~~Base Type~~	~~Description~~
~~**Option**~~	~~String~~	~~An option string, minimum length = 1. The first character is the option id. Remaining characters if any are the option value.~~

4 Mandatory Commands/Responses

~~An OpenC2 command~~ *The content in this section is normative, except where it is marked non-normative.*

A Command consists of an ~~ACTION/TARGET~~Action/Target pair and associated ~~SPECIFIERS~~Specifiers and ~~ARGUMENTs~~Arguments. This section enumerates the allowed ~~commands, identify~~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 3.4.2.4. The 'query features' Command MAY include the `"response_type": "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_type": "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 Sample 1

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 Sample 2

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": {
```

~~"An OpenC2 Consumer MUST process an OpenC2 Command where "query" is specified for the ACTION and "features" is specified for the TARGET, hereafter, referred to as a 'query features' command".~~

~~Upon processing a 'query features' command, an OpenC2 Consumer MUST issue an OpenC2 Response to the OpenC2 Producer that issued the OpenC2 Command.~~

```
features": ["versions", "profiles", "rate_limit"]
    }
}
```

Response:

```
{
    "status": 200,
    "versions": ["1.0"],
    "profiles": ["slpf", "x-lock"],
    "rate_limit": 30
}
```

5 Conformance

~~5.1 OpenC2 Message Content~~

This content in this section is normative.

5.1 Conformance Clause 1: Command

A conformant ~~OpenC2~~ Command

- 5.1-1 MUST be structured in accordance with Section 3.3.1~~Section 3.4.1, and~~.
- 5.1-2 MUST include exactly one ~~ACTION specified~~`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 3.1.5~~Section 3.~~.
- 5.1-4 MUST include zero or one `actuator` property defined in accordance with Section 3.3.1.3 or zero or one imported `actuator` property defined in accordance with Section 3.1.5~~.1.1.~~.
- 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 3.1.5.

5.2 Conformance Clause 2: Response

A conformant ~~OpenC2~~ Response

- 5.2-1 MUST be structured in accordance with Section 3.3.2~~Section 3.4.2, and~~.
- 5.2-2 MUST include exactly one ~~STATUS specified~~`status` property defined in accordance with Section 3.3.2.1~~Section 3.4.2.1.~~.

~~5.2 OpenC2~~5.3 Conformance Clause 3: Producer

A conformant ~~OpenC2~~ Producer

- 5.3-1 MUST issue ~~OpenC2~~ Commands and process ~~OpenC2~~ Responses ~~specified~~ in accordance with Section 4~~Section 4~~.
- 5.3-2 MUST implement JSON serialization of generated ~~OpenC2~~ Commands in accordance with [RFC7493]~~RFC 7493~~.

5.~~3 OpenC2~~4 Conformance Clause 4: Consumer

A conformant ~~OpenC2~~ Consumer

- 5.4-1 MUST process ~~OpenC2~~ Commands and issue ~~OpenC2~~ Responses ~~specified~~ in accordance with Section 4~~Section 4~~.
- 5.4-2 MUST implement JSON serialization of generated ~~OpenC2~~ Responses in accordance with [RFC7493]~~RFC 7493~~.

Annex A. ~~Schemas~~Examples

~~This annex defines the information model used by conforming OpenC2 implementations in JSON Abstract Data Notation (JADN) format. JADN is a structured textual representation of the tables shown in Section 3. Schema files referenced by the URLs include descriptive text shown in the tables. Descriptions are omitted from the figures in this section in order to: 1) illustrate that descriptive text is not part of the language syntax, 2) show what an actuator would return in response to a schema query, and 3) improve readability of the figures.~~

~~A.1 OpenC2 Language Syntax~~

~~Schema File:~~

~~The normative schema file (oc2ls.json) and formatted version (oc2ls.pdf) may be found at the link under above.~~

~~Schema:~~

```
~~{~~
~~ "meta": {~~
~~  "module": "oasis-open.org/openc2/v1.0/openc2-lang",~~
~~  "patch": "wd09",~~
~~  "title": "OpenC2 Language Objects",~~
~~  "description": "Datatypes that define the content of OpenC2 commands and responses.",~~
~~  "imports": [~~
~~    ["slpf", "oasis-open.org/openc2/v1.0/ap-slpf"],~~
~~    ["jadn", "oasis-open.org/openc2/v1.0/jadn"]~~
~~  ],~~
~~  "exports": ["OpenC2-Command", "OpenC2-Response", "Message-Type", "Status-Code", "Request-Id", "Date-Time"]~~
~~ },~~
~~ "types": [~~
~~  ["Message", "Array", [], "", [~~
~~      [1, "msg_type", "Message-Type", [], ""],~~
~~      [2, "content_type", "String", [], ""],~~
```

~~[3, "~~*The* *content*~~", "Null", [], ""],~~ *in this section is non-normative.*

```
~~      [4, "status", "Status-Code", ["[0"], ""],~~
~~      [5, "request_id", "Request-Id", ["[0"], ""],~~
~~      [6, "to", "String", ["[0", "]0"], ""],~~
~~      [7, "from", "String", ["[0"], ""],~~
~~      [8, "created", "Date-Time", ["[0"], ""]~~
```

```
  ~~]],~~
  ~~["OpenC2-Command", "Record", [], "", [~~
    ~~[1, "action", "Action", [], ""],~~
    ~~[2, "target", "Target", [], ""],~~
    ~~[3, "args", "Args", ["[0"], ""],~~
    ~~[4, "actuator", "Actuator", ["[0"], ""]~~
  ~~]],~~
  ~~["Action", "Enumerated", [], "", [~~
    ~~[1, "scan", ""],~~
    ~~[2, "locate", ""],~~
    ~~[3, "query", ""],~~
    ~~[6, "deny", ""],~~
    ~~[7, "contain", ""],~~
    ~~[8, "allow", ""],~~
    ~~[9, "start", ""],~~
    ~~[10, "stop", ""],~~
    ~~[11, "restart", ""],~~
    ~~[14, "cancel", ""],~~
    ~~[15, "set", ""],~~
    ~~[16, "update", ""],~~
    ~~[18, "redirect", ""],~~
    ~~[19, "create", ""],~~
    ~~[20, "delete", ""],~~
    ~~[22, "detonate", ""],~~
    ~~[23, "restore", ""],~~
    ~~[28, "copy", ""],~~
    ~~[30, "investigate", ""],~~
    ~~[32, "remediate", ""]~~
  ~~]],~~
  ~~["Target", "Choice", [], "", [~~
    ~~[1, "artifact", "Artifact", [], ""],~~
    ~~[2, "command", "Request-Id", [], ""],~~
    ~~[3, "device", "Device", [], ""],~~
    ~~[7, "domain_name", "Domain-Name", [], ""],~~
    ~~[8, "email_addr", "Email-Addr", [], ""],~~
    ~~[16, "features", "Features", [], ""],~~
    ~~[10, "file", "File", [], ""],~~
    ~~[11, "ip_addr", "IP-Addr", [], ""],~~
    ~~[15, "ip_connection", "IP-Connection", [], ""],~~
    ~~[13, "mac_addr", "MAC-Addr", [], ""],~~
    ~~[17, "process", "Process", [], ""],~~
    ~~[25, "properties", "Properties", [], ""],~~
    ~~[19, "uri", "URI", [], ""],~~
    ~~[1000, "extension", "PE-Target", [], ""],~~
    ~~[1001, "extension_unr", "Unr-Target", [], ""],~~
    ~~[1024, "slpf", "slpf:Target", [], ""]~~
  ~~]],~~
  ~~["Actuator", "Choice", [], "", [~~
    ~~[1000, "extension", "PE-Specifiers", [], ""],~~
    ~~[1001, "extension_unr", "Unr-Specifiers", [], ""]~~
  ~~]],~~
  ~~["Args", "Map", [], "", [~~
    ~~[1, "start_time", "Date-Time", ["[0"], ""],~~
    ~~[2, "stop_time", "Date-Time", ["[0"], ""],~~
    ~~[3, "duration", "Duration", ["[0"], ""],~~
    ~~[4, "response_requested", "Response-Type", ["[0"], ""],~~
    ~~[1000, "extension", "PE-Args", ["[0"], ""],~~
    ~~[1001, "extension_unr", "Unr-Args", ["[0"], ""]~~
  ~~]],~~
  ~~["OpenC2-Response", "Map", [], "", [~~
    ~~[1, "status", "Status-Code", ["[0"], ""],~~
    ~~[2, "status_text", "String", ["[0"], ""],~~
    ~~[3, "strings", "String", ["[0", "]0"], ""],~~
    ~~[4, "ints", "Integer", ["[0", "]0"], ""],~~
    ~~[5, "kvps", "KVP", ["[0", "]0"], ""],~~
    ~~[6, "versions", "Version", ["[0", "]0"], ""],~~
    ~~[7, "profiles", "jadn:Uname", ["[0", "]0"], ""],~~
    ~~[8, "schema", "jadn:Schema", ["[0"], ""],~~
    ~~[9, "pairs", "Action-Targets", ["[0", "]0"], ""],~~
    ~~[10, "rate_limit", "Number", ["[0"], ""],~~
    ~~[1000, "extension", "PE-Results", ["[0"], ""],~~
    ~~[1001, "extension_unr", "Unr-Results", ["[0"], ""]~~
  ~~]],~~
  ~~["Status-Code", "Enumerated", ["="], "", [~~
    ~~[102, "Processing", ""],~~
    ~~[200, "OK", ""],~~
    ~~[301, "Moved Permanently", ""],~~
    ~~[400, "Bad Request", ""],~~
    ~~[401, "Unauthorized", ""],~~
    ~~[403, "Forbidden", ""],~~
    ~~[404, "Not Found", ""],~~
    ~~[500, "Internal Error", ""],~~
    ~~[501, "Not Implemented", ""],~~
    ~~[503, "Service Unavailable", ""]~~
  ~~]],~~
  ~~["PE-Target", "Choice", ["="], "", [~~
    ~~[32473, "Example", "32473:Target", [], ""]~~
  ~~]],~~
  ~~["PE-Specifiers", "Choice", ["="], "", [~~
    ~~[32473, "Example", "32473:Specifiers", [], ""]~~
  ~~]],~~
  ~~["PE-Args", "Map", ["="], "", [~~
    ~~[32473, "Example", "32473:Args", [], ""]~~
  ~~]],~~
  ~~["PE-Results", "Map", ["="], "", [~~
    ~~[32473, "Example", "32473:Results", [], ""]~~
  ~~]],~~
  ~~["Artifact", "Record", [], "", [~~
```

```
~~    [1, "mime_type", "String", ["[0"], ""],~~
~~    [2, "payload", "Payload", ["[0"], ""],~~
~~    [3, "hashes", "Hashes", ["[0"], ""]~~
~~  ]],~~
~~  ["Device", "Map", [], "", [~~
~~    [1, "hostname", "Hostname", [], ""],~~
~~    [2, "description", "String", ["[0"], ""],~~
~~    [3, "device_id", "String", ["[0"], ""]~~
~~  ]],~~
~~  ["Domain-Name", "String", ["@hostname"], ""],~~
~~  ["Email-Addr", "String", ["@email"], ""],~~
~~  ["Features", "ArrayOf", ["*Feature", "[0"], ""],~~
~~  ["File", "Map", [], "", [~~
~~    [1, "name", "String", ["[0"], ""],~~
~~    [2, "path", "String", ["[0"], ""],~~
~~    [3, "hashes", "Hashes", ["[0"], ""]~~
~~  ]],~~
~~  ["IP-Addr", "Binary", ["@ip-addr"], ""],~~
~~  ["IP-Connection", "Record", [], "", [~~
~~    [1, "src_addr", "IP-Addr", ["[0"], ""],~~
~~    [2, "src_port", "Port", ["[0"], ""],~~
~~    [3, "dst_addr", "IP-Addr", ["[0"], ""],~~
~~    [4, "dst_port", "Port", ["[0"], ""],~~
~~    [5, "protocol", "L4-Protocol", ["[0"], ""]~~
~~  ]],~~
~~  ["MAC-Addr", "Binary", [], ""],~~
~~  ["Process", "Map", [], "", [~~
~~    [1, "pid", "Integer", ["[0"], ""],~~
~~    [2, "name", "String", ["[0"], ""],~~
~~    [3, "cwd", "String", ["[0"], ""],~~
~~    [4, "executable", "File", ["[0"], ""],~~
~~    [5, "parent", "Process", ["[0"], ""],~~
~~    [6, "command_line", "String", ["[0"], ""]~~
~~  ]],~~
~~  ["Properties", "ArrayOf", ["*String"], ""],~~
~~  ["URI", "String", ["@uri"], ""],~~
~~  ["Message-Type", "Enumerated", [], "", [~~
~~    [0, "notification", ""],~~
~~    [1, "request", ""],~~
~~    [2, "response", ""]~~
~~  ]],~~
~~  ["Request-Id", "Binary", [], ""],~~
~~  ["Date-Time", "Integer", [], ""],~~
~~  ["Duration", "Integer", [], ""],~~
~~  ["Hashes", "Map", [], "", [~~
~~    [1, "md5", "Binary", ["[0"], ""],~~
~~    [4, "sha1", "Binary", ["[0"], ""],~~
~~    [6, "sha256", "Binary", ["[0"], ""]~~
~~  ]],~~
~~  ["Hostname", "String", [], ""],~~
~~  ["L4-Protocol", "Enumerated", [], "", [~~
~~    [1, "icmp", ""],~~
~~    [6, "tcp", ""],~~
~~    [17, "udp", ""],~~
~~    [132, "sctp", ""]~~
~~  ]],~~
~~  ["Payload", "Choice", [], "", [~~
~~    [1, "bin", "Binary", [], ""],~~
~~    [2, "url", "URI", [], ""]~~
~~  ]],~~
~~  ["Port", "Integer", ["[0", "]65535"], ""],~~
~~  ["Feature", "Enumerated", [], "", [~~
~~    [1, "versions", ""],~~
~~    [2, "profiles", ""],~~
~~    [3, "schema", ""],~~
~~    [4, "pairs", ""],~~
~~    [5, "rate_limit", ""]~~
~~  ]],~~
~~  ["Response-Type", "Enumerated", [], "", [~~
~~    [0, "~~A~~none", ""],~~
~~    [1, "ack", ""],~~
~~    [2, "status", ""],~~
~~    [3, "complete", ""]~~
~~  ]],~~
~~  ["Version", "String", [], ""],~~
~~  ["KVP", "Array", [], "", [~~
~~    [1, "key", "String", [], ""],~~
~~    [2, "value", "String", [], ""]~~
~~  ]],~~
~~  ["Action-Targets", "Array", [], "", [~~
~~    [1, "action", "Action", [], ""],~~
~~    [2, "targets", "Target", ["]0", "*"], ""]~~
~~  ]]~~
~~ ]~~
~~}~~
```

~~A.2 JADN Syntax~~

~~Schema File:~~

~~The normative schema file (jadn.json) and formatted version (jadn.pdf) may be found at the link under above.~~

~~Schema:~~

```
~~{~~
~~ "meta": {~~
~~  "module": "oasis-open.org/openc2/v1.0/jadn",~~
~~  "patch": "wd01",~~
~~  "title": "JADN Syntax",~~
~~  "description": "Syntax of a JSON Abstract Data Notation (JADN)~~
~~module.",~~
~~  "exports": ["Schema", "Uname"]~~
~~ },~~

~~ "types": [~~
~~  ["Schema", "Record", [], "", [~~
~~     [1, "meta", "Meta", [], ""],~~
~~     [2, "types", "Type", ["]0"], ""]]~~
~~  ],~~

~~  ["Meta", "Map", [], "", [~~
~~     [1, "module", "Uname", [], ""],~~
~~     [2, "patch", "String", ["[0"], ""],~~
~~     [3, "title", "String", ["[0"], ""],~~
~~     [4, "description", "String", ["[0"], ""],~~
~~     [5, "imports", "Import", ["[0", "]0"], ""],~~
~~     [6, "exports", "Identifier", ["[0", "]0"], ""],~~
~~     [7, "bounds", "Bounds", ["[0"], ""]]~~
~~  ],~~

~~  ["Import", "Array", [], "", [~~
~~     [1, "nsid", "Nsid", [], ""],~~
~~     [2, "uname", "Uname", [], ""]]~~
~~  ],~~

~~  ["Bounds", "Array", [], "", [~~
~~     [1, "max_msg", "Integer", [], ""],~~
~~     [2, "max_str", "Integer", [], ""],~~
~~     [3, "max_bin", "Integer", [], ""],~~
~~     [4, "max_fields", "Integer", [], ""]]~~
~~  ],~~

~~  ["Type", "Array", [], "", [~~
~~     [1, "tname", "Identifier", [], ""],~~
~~     [2, "btype", "JADN-Type", ["*"], ""],~~
~~     [3, "opts", "Option", ["]0"], ""],~~
~~     [4, "desc", "String", [], ""],~~
~~     [5, "fields", "JADN-Type", ["&btype", "]0"], ""]]~~
~~  ],~~

~~  ["JADN-Type", "Choice", [], "", [~~
~~     [1, "Binary", "Null", [], ""],~~
~~     [2, "Boolean", "Null", [], ""],~~
~~     [3, "Integer", "Null", [], ""],~~
~~     [4, "Number", "Null", [], ""],~~
~~        [5, "Null", "Null", [], ""],~~
~~        [6, "String", "Null", [], ""],~~
~~        [7, "Array", "FullField", ["]0"], ""],~~
~~        [8, "ArrayOf", "Null", [], ""],~~
~~        [9, "Choice", "FullField", ["]0"], ""],~~
~~        [10, "Enumerated", "EnumField", ["]0"], ""],~~
~~        [11, "Map", "FullField", ["]0"], ""],~~
~~        [12, "Record", "FullField", ["]0"], ""]]~~
~~    ],~~

~~    ["EnumField", "Array", [], "", [~~
~~        [1, "", "Integer", [], ""],~~
~~        [2, "", "String", [], ""],~~
~~        [3, "", "String", [], ""]]~~
~~    ],~~

~~    ["FullField", "Array", [], "", [~~
~~        [1, "", "Integer", [], ""],~~
~~        [2, "", "Identifier", [], ""],~~
~~        [3, "", "Identifier", [], ""],~~
~~        [4, "", "Options", [], ""],~~
~~        [5, "", "String", [], ""]]~~
~~    ],~~

~~    ["Identifier", "String", ["$^[a-zA-Z][\\w-]*$", "[1", "]32"],~~
~~""],~~

~~    ["Nsid", "String", ["$^[a-zA-Z][\\w-]*$", "[1", "]8"], ""],~~

~~    ["Uname", "String", ["[1", "]100"], ""],~~

~~    ["Options", "ArrayOf", ["*Option", "[0", "]10"], ""],~~

~~    ["Option", "String", ["[1", "]100"], ""]]~~
~~}~~
```

~~Annex B. Examples~~

~~B~~.1 Example 1

This example shows the elements of an OpenC2 Message containing an OpenC2 Command. The content of the ~~m~~Message is the de-serialized ~~c~~Command structure in whatever format is used by the implementation, independent of the transfer protocol and serialization format used to transport the ~~m~~Message.

The request_id in this example is a 64 bit binary value which can be displayed in many ways, including hex: `'d937 fca9 2b64 4e71'`, base64url: `'2Tf8qStkTnE'`, and Python byte string - ASCII characters with hex escapes (\xNN) for non-ASCII bytes: `b'\xd97\xfc\xa9+dNq'`. If ~~OpenC2 producers~~Producers generate numeric or alphanumeric request_ids, they are still binary values and are limited to 128 bits, e.g.,: hex: `'6670 2d31 3932 352d 3337 3632 3864 3663'`, base64url: `'ZnAtMTkyNS0zNzYyOGQ2Yw'`, byte string: `b'fp-1925-37628d6c'`.

The created element is a Date-Time value of milliseconds since the epoch. The example `1539355895215` may be displayed ~~as '12~~as '12 October 2018 14:51:35 UTC'.

This example, illustrating an internal representation of a ~~m~~Message, is non-normative. Other programming languages (e.g., Java, Javascript, C, Erlang) have different representations of literal values. There are no interoperability considerations or conformance requirements for how ~~m~~Message elements are represented internally within an implementation. Only the serialized values of the ~~m~~Message elements embedded within a protocol is relevant to interoperability.

~~B~~A.1.1 Example 1: Command ~~Message~~

```
content-type: 'application/openc2'
msg_type: 'request'
request_id: b'\xd97\xfc\xa9+dNq'
from: 'nocc-3497'
to: ['#filter-devices']
created: 1539355895215
content: {'action': 'query', 'target': {'features':
['versions', 'profiles']}}
```

~~B~~A.1.2 Example 1: Response ~~Message~~

The ~~response message~~Response Message contains a status code, a content-type that is normally the same as the request content type, a msg_type of `'response'`, and the ~~r~~Response content. The request_id from the ~~command message~~Command Message, if present, is returned unchanged in the ~~response message.~~Response Message. The "to" element of the ~~r~~Response normally echoes the "from" element of the ~~command message~~Command Message, but the "from" element of the ~~r~~Response is the ~~a~~Actuator's identifier regardless of whether the ~~c~~Command was sent to an individual actuator or a group. The "created" element, if present, contains the creation time of the ~~r~~Response.

A responder could potentially return non-openc2 content, such as a PDF report or an HTML document, in response to ~~an openc2 command.~~a

Command. No ~~a~~Actuator profiles currently define response content types other than openc2.

```
status: 200
content-type: 'application/openc2'
msg_type: 'response'
request_id: b'\xd97\xfc\xa9+dNq'
from: 'pf72394'
to: ['nocc-3497']
created: 1539355898000
content: {'status': 200, 'versions': ['1.3'], 'profiles':
[~~'oasis-open.org/openc2/v1.0/ap-slpf'~~'slpf']}
```

~~B~~A.2 Example 2

This example is for a transport where the header information is outside the JSON (e.g., HTTPS API) and only body is in JSON.

A.2.1 Example 2: Command~~:~~

```
{
    "action": "query",
    "target": {
        "properties": ["battery~~_percentage~~"]
    },
    "actuator": {
        "x-esm": {
            "asset_id": "TGEadsasd"
        }
    ~~}:~~}
}
```

A.2.2 Example 2: Response~~:~~

```
{
    "status": 200,
    "~~kvps": [["~~results": {
        "battery~~_percentage", "~~": {
            "capacity": 0.577216~~"]]~~,
~~}~~
~~B~~            "charged at": 1547506988,
            "status": 12,
            "mode": {
                "output": high",
                "supported": [ "high", "trickle" ]
            }
            "visible on display": true
        },
        "asset id": "TGEadsasd"
    }
}
```

A.3 Example 3

A.3.1 Example 3: Command~~:~~

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

A.3.2 Example 3: Response~~:~~

```
{
    "status_text": "ACME Corp Internet Toaster",
    "versions": ["1.0"],
    "profiles": ~~[]~~["slpf", "x-acme"]
}
```

Annex B. Acronyms

~~**Command:**~~

~~This command queries the actuator for the syntax of its supported commands.~~

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

~~**Response:**~~

~~This example illustrates how actuator developers tailor the OpenC2 schema to communicate the capabilities of their products to producers. This example actuator supports the mandatory requirements of the language specification plus a random subset of optional language elements (cancel, create, and delete actions, and the command, ip_addr, and properties targets). The example actuator supports a subset of the core OpenC2 language but no profile-defined targets, actuator specifiers, command arguments, or responses.~~

~~The example do-nothing actuator appears to support create and delete ip_addr commands, but without a profile there is no definition of what the actuator would do to "create" an IP address. The schema is used by producers to determine what commands are syntactically valid for an actuator, but it does not assign meaning to those commands.~~

```
~~{~~
  ~~"pairs": [~~
    ~~["query", ["features", "properties"]],~~
    ~~["cancel", ["command"]],~~
    ~~["create", ["ip_addr"]],~~
    ~~["delete", ["ip_addr"]]~~
  ~~],~~
  ~~"schema": {~~
     ~~"meta": {~~
      ~~"module": "oasis-open.org/openc2/v1.0/openc2-lang",~~
      ~~"patch": "wd09_example",~~
      ~~"title": "OpenC2 Language Objects",~~
      ~~"description": "Example Actuator",~~
      ~~"exports": ["OpenC2-Command", "OpenC2-Response", "Message-~~
~~Type", "Status-Code", "Request-Id", "Date-Time"]~~
     ~~},~~
     ~~"types": [~~
      ~~["OpenC2-Command", "Record", [], "", [~~
        ~~[1, "action", "Action", [], ""],~~
        ~~[2, "target", "Target", [], ""],~~
        ~~[3, "args", "Args", ["[0"], ""]~~
      ~~]],~~
      ~~["Action", "Enumerated", [], "", [~~
        ~~[3, "query", ""],~~
        ~~[14, "cancel", ""],~~
        ~~[19, "create", ""],~~
        ~~[20, "delete", ""]~~
      ~~]],~~
      ~~["Target", "Choice", [], "", [~~
        ~~[2, "command", "Request-Id", [], ""],~~
        ~~[16, "features", "Features", [], ""],~~
        ~~[11, "ip_addr", "IP-Addr", [], ""],~~
        ~~[25, "properties", "Properties", [], ""]~~
      ~~]],~~
      ~~["Args", "Map", [], "", [~~
        ~~[1, "start_time", "Date-Time", ["[0"], ""],~~
        ~~[4, "response_requested", "Response-Type", ["[0"], ""]~~
      ~~]],~~
      ~~["OpenC2-Response", "Map", [], "", [~~
        ~~[2, "status_text", "String", ["[0"], ""],~~
        ~~[3, "strings", "String", ["[0", "]0"], ""],~~
        ~~[4, "ints", "Integer", ["[0", "]0"], ""],~~
        ~~[5, "kvps", "KVP", ["[0", "]0"], ""],~~
        ~~[6, "versions", "Version", ["[0", "]0"], ""],~~
        ~~[7, "profiles", "jadn:Uname", ["[0", "]0"], ""],~~
        ~~[8, "schema", "jadn:Schema", ["[0"], ""],~~
        ~~[9, "pairs", "Action-Targets", ["[0", "]0"], ""],~~
        ~~[10, "rate_limit", "Number", ["[0"], ""]~~
      ~~]],~~
```

```
~~        ["Status-Code", "Enumerated", ["="], "", [~~
~~            [200, "OK", ""],~~
~~            [400, "Bad Request", ""],~~
~~            [404, "Not Found", ""],~~
~~            [500, "Internal Error", ""],~~
~~            [501, "Not Implemented", ""]~~
~~        ]],~~
~~        ["Features", "ArrayOf", ["*Feature", "[0"], ""],~~
~~        ["IP-Addr", "Binary", ["@ip-addr"], ""],~~
~~        ["Properties", "ArrayOf", ["*String"], ""],~~
~~        ["Message-Type", "Enumerated", [], "", [~~
~~            [1, "request", ""],~~
~~            [2, "response", ""]~~
~~        ]],~~
~~        ["Request-Id", "Binary", [], ""],~~
~~        ["Date-Time", "Integer", [], ""],~~
~~        ["Feature", "Enumerated", [], "", [~~
~~            [1, "versions", ""],~~
~~            [2, "profiles", ""],~~
~~            [3, "schema", ""],~~
~~            [4, "pairs", ""]~~
~~        ]],~~
~~        ["Response-Type", "Enumerated", [], "", [~~
~~            [0, "none", ""],~~
~~            [1, "ack", ""],~~

~~            [3, "~~
```

Acronym	Definition
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
BCP	Best Current Practice
CBOR	Concise Binary Object Representation
CIDR	Classless Inter-Domain Routing
CoAP	Constrained Application Protocol
DOI	Digital Object Identifier

Acronym	Definition
EUI	Extended Unique Identifier
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
IACD	Integrated Adaptive Cyber Defense
IANA	Internet Assigned Numbers Authority
ICMP	Internet Control Message Protocol
ID	Identifier
IP	Internet Protocol
IPR	Intellectual Property Rights
JSON	JavaScript Object Notation
MAC	Media Access Control
MD5	Message Digest
MQTT	Message Queuing Telemetry Transfer
OASIS	Organization for the Advancement of Structured Information Standards
OODA	Observe-Orient-Decide-Act
OpenC2	Open Command and Control
OpenDXL	Open Data eXchange Layer

Acronym	Definition
PDF	Portable Document Format
RFC	Request for Comment
SCTP	Stream Control Transmission Protocol
SHA	Security Hash Algorithm
SLPF	StateLess Packet Filtering
STD	Standard
TC	Technical Committee
TCP	Transmission Control Protocol
UDP	User Datagram Control Protocol
UML	Unified Modeling Language
URI	Uniform Resource Identifier
UTC	Coordinated Universal Time
UUID	Universally Unique IDentifier
XML	eXtensibel Markup Language

~~complete", ""]~~
~~]],~~
~~["Version", "String", [], ""],~~
~~["KVP", "Array", [], "", [~~
~~[1, "key", "String", [], ""],~~
~~[2, "value", "String", [], ""]~~
~~]],~~
~~["Action-Targets", "Array", [], "", [~~

```
        ~~[1, "action", "Action", [], ""],~~
        ~~[2, "targets", "Target", ["]0", "*"], ""]~~
      ~~]],~~
      ~~["jadn:Schema", "Record", [], "", [~~
        ~~[1, "meta", "jadn:Meta", [], ""],~~
        ~~[2, "types", "jadn:Type", ["]0"], ""]~~
      ~~]],~~
      ~~["jadn:Meta", "Map", [], "", [~~
        ~~[1, "module", "jadn:Uname", [], ""],~~
        ~~[2, "patch", "String", ["[0"], ""],~~
        ~~[3, "title", "String", ["[0"], ""],~~
        ~~[4, "description", "String", ["[0"], ""],~~
        ~~[5, "imports", "jadn:Import", ["[0", "]0"], ""],~~
        ~~[6, "exports", "jadn:Identifier", ["[0", "]0"], ""],~~
        ~~[7, "bounds", "jadn:Bounds", ["[0"], ""]~~
      ~~]],~~
      ~~["jadn:Import", "Array", [], "", [~~
        ~~[1, "nsid", "jadn:Nsid", [], ""],~~
        ~~[2, "uname", "jadn:Uname", [], ""]~~
      ~~]],~~
      ~~["jadn:Bounds", "Array", [], "", [~~
        ~~[1, "max_msg", "Integer", [], ""],~~
        ~~[2, "max_str", "Integer", [], ""],~~
        ~~[3, "max_bin", "Integer", [], ""],~~
        ~~[4, "max_fields", "Integer", [], ""]~~
      ~~]],~~
      ~~["jadn:Type", "Array", [], "", [~~
        ~~[1, "tname", "jadn:Identifier", [], ""],~~
        ~~[2, "btype", "jadn:JADN-Type", ["*"], ""],~~
        ~~[3, "opts", "jadn:Option", ["]0"], ""],~~
        ~~[4, "desc", "String", [], ""],~~
        ~~[5, "fields", "jadn:JADN-Type", ["&btype", "]0"], ""]~~
      ~~]],~~
      ~~["jadn:JADN-Type", "Choice", [], "", [~~
        ~~[1, "Binary", "Null", [], ""],~~
        ~~[2, "Boolean", "Null", [], ""],~~
        ~~[3, "Integer", "Null", [], ""],~~
        ~~[4, "Number", "Null", [], ""],~~
        ~~[5, "Null", "Null", [], ""],~~
        ~~[6, "String", "Null", [], ""],~~
        ~~[7, "Array", "jadn:FullField", ["]0"], ""],~~
        ~~[8, "ArrayOf", "Null", [], ""],~~
        ~~[9, "Choice", "jadn:FullField", ["]0"], ""],~~
        ~~[10, "Enumerated", "jadn:EnumField", ["]0"], ""],~~
        ~~[11, "Map", "jadn:FullField", ["]0"], ""],~~
        ~~[12, "Record", "jadn:FullField", ["]0"], ""]~~
      ~~]],~~
      ~~["jadn:EnumField", "Array", [], "", [~~
        ~~[1, "", "Integer", [], ""],~~
```

```
~~        [2, "", "String", [], ""],~~
~~        [3, "", "String", [], ""]~~
~~      ]],~~
~~      ["jadn:FullField", "Array", [], "", [~~
~~        [1, "", "Integer", [], ""],~~
~~        [2, "", "jadn:Identifier", [], ""],~~
~~        [3, "", "jadn:Identifier", [], ""],~~
~~        [4, "", "jadn:Options", [], ""],~~
~~        [5, "", "String", [], ""]~~
~~      ]],~~
~~      ["jadn:Identifier", "String", ["$^[a-zA-Z][\\w-]*$", "[1",~~
~~"]32"], ""],~~
~~      ["jadn:Nsid", "String", ["$^[a-zA-Z][\\w-]*$", "[1", "]8"],~~
~~""],~~
~~      ["jadn:Uname", "String", ["[1", "]100"], ""],~~
~~      ["jadn:Options", "ArrayOf", ["*jadn:Option", "[0", "]10"],~~
~~""],~~
~~      ["jadn:Option", "String", ["[1", "]100"], ""]~~
~~    ]~~
~~  }~~
~~}~~
```

Annex C. ~~Acronyms~~

~~**Editor's Note** - TBSL - This section be included in the final version of the initial Committee Specification.~~

~~Annex D.~~ Revision History

The content in this section is non-normative.

Revision	Date	Editor	Changes Made
v1.0-wd01	10/31/2017	Romano, Sparrell	Initial working draft
v1.0-csd01	11/14/2017	Romano, Sparrell	approved wd01

Revision	Date	Editor	Changes Made
v1.0-wd02	01/12/2018	Romano, Sparrell	csd01 ballot comments ~~t~~Targets
v1.0-wd03	01/31/2018	Romano, Sparrell	wd02 review comments
v1.0-csd02	02/14/2018	Romano, Sparrell	approved wd03
v1.0-wd04	03/02/2018	Romano, Sparrell	Property tables threads (cmd/resp) from use cases previous comments
v1.0-wd05	03/21/2018	Romano, Sparrell	wd04 review comments
v1.0-csd03	04/03/2018	Romano, Sparrell	approved wd05
v1.0-wd06	05/15/2018	Romano, Sparrell	Finalizing message structure message=header+body Review comments Using word ~~“~~'arguments~~”~~' instead of ~~“~~'options~~”~~'
v1.0-csd04	5/31/2018	Romano, Sparrell	approved wd06
v1.0-wd07	7/11/2018	Romano, Sparrell	Continued refinement of details Review comments Moved some ~~a~~Actions and ~~t~~Targets to reserved lists
v1.0-wd08	10/05/2018	Romano, Sparrell	Continued refinement of details Review comments

Revision	Date	Editor	Changes Made
v1.0-wd09	10/17/2018	Romano, Sparrell	Additional review comments to create wd09 for CSD approval and release for public review.
v1.0-wd10	03/04/2019	Romano, Sparrell	Produce interim working draft.
v1.0-wd11	03/21/2019	Romano, Sparrell	Produce interim working draft.
v1.0-wd12	03/27/2019	Romano, Sparrell	Produce candidate working draft for next public review.

Annex ~~E~~D. Acknowledgments

The content in this section is non-normative.

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

OpenC2 TC Members:

First Name	Last Name	Company
Philippe	Alcoy	Arbor Networks
~~Alex~~	~~Amirnovin~~	~~Viasat~~
~~Kris~~	~~Anderson~~	~~Trend Micro~~
Darren	Anstee	Arbor Networks
~~Jonathan~~	~~Baker~~	~~Mitre Corporation~~

~~Theodor~~	~~Balanescu~~	~~TELUS~~
~~Stephen~~	~~Banghart~~	~~NIST~~
~~Sean~~	~~Barnum~~	~~FireEye Inc.~~
Michelle	Barry	AT&T
~~Omer~~	~~Ben-Shalom~~	~~Intel Corporation~~
Brian	Berliner	Symantec Corp.
~~Adrian~~	~~Bishop~~	~~Huntsman Security~~
~~Tom~~	~~Blauvelt~~	~~Symantec Corp.~~
~~Phillip~~	~~Boles~~	~~FireEye Inc.~~
Adam	Bradbury	EclecticIQ
~~Sarah~~	~~Brown~~	~~NCI Agency~~
Joe	Brule	National Security Agency
Michael	Butt	NC4
Toby	Considine	University of North Carolina at Chapel Hill
~~Gus~~	~~Creedon~~	~~Logistics Management Institute~~
~~James~~	~~Crossland~~	~~Northrop Grumman~~
Trey	Darley	New Context Services Inc.

David	Darnell	North American Energy Standards Board
Sudeep	Das	McAfee
~~Mark~~	~~Davidson~~	~~NC4~~
~~Stefano~~Andrea	De ~~Crescenzo~~Bernardi	~~Cisco Systems~~Moviri SPA
~~Michele~~	~~Drgon~~	~~Individual~~
~~Alexandre~~	~~Dulaunoy~~	~~CIRCL~~
~~Daniel~~	~~Dye~~	~~NC4~~
~~Chet~~	~~Ensign~~	~~OASIS~~
Blake	Essing	AT&T
Alex	Everett	University of North Carolina at Chapel Hill
Joyce	Fai	National Security Agency
Travis	Farral	Anomali
~~Jessica~~	~~Fitzgerald-McKay~~	~~National Security Agency~~
~~Jim~~	~~Fowler~~	~~US Department of Defense (DoD)~~
David	Girard	Trend Micro
~~Russell~~	~~Glenn~~	~~Viasat~~
~~Juan~~	~~Gonzalez~~	~~DHS Office of Cybersecurity and Communications (CS&C)~~

Andy	Gray	ForeScout
John-Mark	Gurney	New Context Services Inc.
~~Pavel~~	~~Gutin~~	~~G2~~
~~Allen~~	~~Hadden~~	~~IBM~~
Stefan	Hagen	Individual
David	Hamilton	AT&T
~~Daichi~~	~~Hasumi~~	~~NEC Corporation~~
~~Tim~~	~~Hudson~~	~~Cryptsoft Pty Ltd.~~
Nick	Humphrey	Huntsman Security
Christian	Hunt	New Context Services Inc.
~~Andras~~April	~~Iklody~~Jackson	~~CIRCL~~G2
~~Erick~~	~~Ingleby~~	~~ForeScout~~
Sridhar	Jayanthi	Individual
~~Tim~~	~~Jones~~	~~ForeScout~~
Bret	Jordan	Symantec Corp.
~~Takahiro~~Jason	~~Kakumaru~~Keirstead	~~NEC Corporation~~IBM
~~Kirill~~	~~Kasavchenko~~	~~Arbor Networks~~

David	Kemp	National Security Agency
~~Himanshu~~	~~Kesar~~	~~LookingGlass~~
~~Ivan~~	~~Kirillov~~	~~Mitre Corporation~~
~~Lauri~~	~~Korts-Pärn~~	~~NEC Corporation~~
~~Anuj~~	~~Kumar~~	~~FireEye Inc.~~
~~Kent~~	~~Landfield~~	~~McAfee~~
~~Cheolho~~	~~Lee~~	~~NSRI~~
David	Lemire	G2
~~ChangKun~~	~~Li~~	~~360 Enterprise Security Group~~
~~Anthony~~	~~Librera~~	~~AT&T~~
Jason	Liu	Northrop Grumman
~~Terry~~	~~MacDonald~~	~~Individual~~
~~Scott~~	~~MacGregor~~	~~McAfee~~
Radu	Marian	Bank of America
Danny	Martinez	G2
~~Web~~	~~Master~~	~~OASIS~~
~~Ryusuke~~	~~Masuoka~~	~~Fujitsu Limited~~

Lisa	Mathews	National Security Agency
~~Vasileios~~	~~Mavroeidis~~	~~IFI~~
~~Andrew~~	~~May~~	~~Viasat~~
James	Meck	FireEye Inc.
~~Andrew~~	~~Mellinger~~	~~Carnegie Mellon University~~
~~Adam~~	~~Montville~~	~~CIS~~
~~Christopher~~	~~O'Brien~~	~~EclecticIQ~~
Efrain	Ortiz	Symantec Corp.
Paul	Patrick	FireEye Inc.
~~Andrew~~	~~Pendergast~~	~~ThreatConnect, Inc.~~
Michael	Pepin	NC4
~~Wende~~	~~Peters~~	~~Bank of America~~
~~Hugh~~	~~Pyle~~	~~IBM~~
Nirmal	Rajarathnam	ForeScout
~~Greg~~	~~Reaume~~	~~TELUS~~
~~Joe~~	~~Reese~~	~~ThreatConnect, Inc.~~
~~Brennen~~	~~Reynolds~~	~~ForeScout~~

Chris	Ricard	Financial Services Information Sharing and Analysis Center (FS-ISAC)
Daniel	Riedel	New Context Services Inc.
~~Robert~~	~~Roll~~	~~Arizona Supreme Court~~
Jason	Romano	National Security Agency
~~Michael~~	~~Rosa~~	~~DHS Office of Cybersecurity and Communications (CS&C)~~
Philip	Royer	Splunk Inc.
~~Anthony~~	~~Rutkowski~~	~~Yanna Technologies LLC~~
~~Steven~~	~~Ryan~~	~~Individual~~
~~Omar~~	~~Santos~~	~~Cisco Systems~~
~~Sourabh~~	~~Satish~~	~~Splunk Inc.~~
~~Aleksandra~~	~~Scalco~~	~~US Department of Defense (DoD)~~
Thomas	Schreck	Siemens AG
~~Dee~~	~~Schur~~	~~OASIS~~
~~Randall~~	~~Sharo~~	~~US Department of Defense (DoD)~~
~~Eric~~	~~Shulze~~	~~Trend Micro~~
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	New Context Services Inc.
Gerald	Stueve	Fornetix
~~Natalie~~	~~Suarez~~	~~NC4~~
Rodney	Sullivan	NCI Agency
~~Sam~~	~~Taghavi Zargar~~	~~Cisco Systems~~
Allan	Thomson	LookingGlass
Bill	Trost	AT&T
~~Ryan~~	~~Trost~~	~~ThreatQuotient, Inc.~~
Raymon	van der Velde	EclecticIQ
~~Drew~~	~~Varner~~	~~NineFX, Inc.~~
~~Tom~~	~~Vaughan~~	~~EclecticIQ~~
Jyoti	Verma	Cisco Systems

~~Kamer~~	~~Vishi~~	~~IFI~~
~~Eric~~	~~Voit~~	~~Cisco Systems~~
David	Waltermire	NIST
Jason	Webb	LookingGlass
~~David~~	~~Webber~~	~~Huawei Technologies Co., Ltd.~~
Sean	Welsh	AT&T
~~Remko~~	~~Weterings~~	~~FireEye Inc.~~
Charles	White	Fornetix
~~Koji~~	~~Yamada~~	~~Fujitsu Limited~~
Sounil	Yu	Bank of America
~~Paolo~~	~~Zaino~~	~~LookingGlass~~