



Open Command and Control (OpenC2) Language Specification Version 1.0

Committee Specification Draft 03

03 April 2018

Specification URIs

This version:

<http://docs.oasis-open.org/openc2/oc2ls/v1.0/csd03/oc2ls-v1.0-csd03.docx>
<http://docs.oasis-open.org/openc2/oc2ls/v1.0/csd03/oc2ls-v1.0-csd03.html>
<http://docs.oasis-open.org/openc2/oc2ls/v1.0/csd03/oc2ls-v1.0-csd03.pdf>

Previous version:

<http://docs.oasis-open.org/openc2/oc2ls/v1.0/csd02/oc2ls-v1.0-csd02.docx>
<http://docs.oasis-open.org/openc2/oc2ls/v1.0/csd02/oc2ls-v1.0-csd02.html>
<http://docs.oasis-open.org/openc2/oc2ls/v1.0/csd02/oc2ls-v1.0-csd02.pdf>

Latest version:

<http://docs.oasis-open.org/openc2/oc2ls/v1.0/oc2ls-v1.0.pdf>
<http://docs.oasis-open.org/openc2/oc2ls/v1.0/oc2ls-v1.0.html>
<http://docs.oasis-open.org/openc2/oc2ls/v1.0/oc2ls-v1.0.docx>

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:

- The Authoritative version of this specification, in the Markdown language: <http://docs.oasis-open.org/openc2/oc2ls/v1.0/csd03/md/oc2ls-v1.0-wd05.md>.

Abstract:

Cyberattacks are increasingly sophisticated, less expensive to execute, dynamic and automated. The provision of cyberdefense via statically configured products operating in isolation is no longer tenable. Standardized interfaces, protocols and data models will facilitate the integration of the functional blocks within a system or enterprise. Open Command and Control (OpenC2) is a concise and extensible language to enable the command and control of cyber defense components, subsystems and/or systems in a manner that is agnostic of the underlying products, technologies, transport mechanisms or other aspects of the implementation. It should be understood that a language such as OpenC2 is necessary but insufficient to enable coordinated cyber response. Other aspects of coordinated cyber response such as sensing, analytics, and selecting appropriate courses of action are beyond the scope of OpenC2.

Status:

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

TC members should send comments on this specification to the TC’s email list. Others should send comments to the TC’s public comment list, after subscribing to it by following the instructions at the “[Send A Comment](#)” button on the TC’s web page at <https://www.oasis-open.org/committees/openc2/>.

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

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

Citation format:**[OpenC2-Lang-v1.0]**

Open Command and Control (OpenC2) Language Specification Version 1.0. Edited by Jason Romano and Duncan Sparrell. 03 April 2018. OASIS Committee Specification Draft 03. <http://docs.oasis-open.org/openc2/oc2ls/v1.0/csd03/oc2ls-v1.0-csd03.html>. Latest version: <http://docs.oasis-open.org/openc2/oc2ls/v1.0/oc2ls-v1.0.html>.

Notices

Copyright © OASIS Open 2018. All Rights Reserved.

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

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

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

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

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

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

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

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

Table of Contents

1	Introduction	7
1.1	Goal	7
1.2	Purpose and Scope	7
1.3	IPR Policy	8
1.4	Terminology	8
1.5	Document Conventions	8
1.6	Naming Conventions	8
1.7	Normative References	9
2	OpenC2 Language	10
2.1	Overview	10
2.2	OpenC2 Command	10
2.2.1	Command Structure	10
2.2.2	Action Vocabulary	11
2.2.3	Target Vocabulary	13
2.2.4	Actuator	14
2.2.5	Command-Option Vocabulary	15
2.2.6	Imported Data	15
2.3	OpenC2 Response	16
2.3.1	Response Structure	16
3	OpenC2 Property Tables	17
3.1	Terminology	17
3.2	OpenC2 Messages	17
3.2.1	OpenC2 Command	18
3.2.1.1	Type Name: OpenC2-Command	18
3.2.1.2	Type Name: Action	18
3.2.1.3	Type Name: Target	19
3.2.1.4	Type Name: Actuator	21
3.2.1.5	Type Name: Command-Options	21
3.2.2	OpenC2 Response	22
3.2.2.1	Type Name: OpenC2-Response	22
3.2.2.2	Type Name: Status-Code	22
3.3	Property Details	23
3.3.0	23
3.3.0.1	Type Name: IP-Connection	23
3.3.0.2	Type Name: IP-Addr	23
3.3.0.3	Type Name: Port	24
3.3.0.4	Type Name: L4-Protocol	24
3.3.0.5	Type Name: File	24
3.3.0.6	Type Name: Response-Requested	24
3.3.0.7	Type Name: Command-ID	25
3.3.0.8	Type Name: Identifier	25
3.3.0.9	Type Name: Version	25
3.3.0.10	Type Name: Domain-Name	25
3.3.0.11	Type Name: Email-Message	25

3.3.0.12 Type Name: Process	25
3.3.0.13 Type Name: Hashes	26
3.3.0.14 Type Name: Hostname	26
3.3.0.15 Type Name: Device	26
4 Foundational Actuator Profile	27
5 Conformance	28
Appendix A. Acknowledgments	29
Appendix B. Revision History	30
Appendix C. Acronyms	31
Appendix D. Examples	32

Editor's Note: This document is NOT complete.

The document development process is based on agile software development principles. Iterative, incremental working documents are being developed, reviewed by the Language Subcommittee, and then submitted to the Technical Committee for approval as a Committee Specification Drafts (CSD).

This is iteration 2 and the expectation is there will be 4 or 5 CSD iterations before this document is complete and ready to be submitted for approval as a Committee Specification.

Parenthetical "Editor's Notes" will be removed prior to submitting for Committee Specification. Sections that are expected to added in a later iteration (prior to 1.0) will be labeled with "TBSL" for "To Be Supplied Later", optionally with a guestimate as to which iteration it would be supplied in.

1 Introduction

The OpenC2 Language Specification defines a language used to compose messages for command and control of cyber defense systems and components.

The OpenC2 language defines two message types:

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 command
2. **Response:** Any information captured or necessary to send back to the OpenC2 Producer system that requested the Command be invoked, i.e., the OpenC2 Consumer response to the OpenC2 Producer.

The components of an OpenC2 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 options, which influence how the command is to be performed. An action coupled with a target is sufficient to describe a complete OpenC2 Command. The inclusion of an actuator and/or command-options provide additional precision.

Additional detail regarding the TARGET and ACTUATOR may be included to increase the precision of the command. For example, which target (i.e., target specifier), additional information about what is to be performed on a specific target type (i.e., target option), which actuator(s) (i.e., actuator specifier) and/or additional information regarding how a specific actuator executes the action (i.e., actuator option).

An OpenC2 Response is issued as a result of an OpenC2 command. OpenC2 responses are used to provide acknowledgement, status, results of command execution, or other information in conjunction with a particular command.

1.1 Goal

Editor's Note - TBSL - This section will be included in a future iteration (probably iteration 5) prior to submitting for Committee Specification.

1.2 Purpose and Scope

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

1. the set of actions and options that may be used in OpenC2 commands
2. the set of targets, target specifiers, and target options
3. A syntax that defines the structure of commands and responses
4. an organizational scheme that describes an Actuator Profile
5. The MTI serialization of OpenC2 commands, and responses
6. the procedures for extending the language

The OpenC2 language assumes that the event has been detected, a decision to act has been made, the act is warranted, and the initiator and recipient of the commands are authenticated and authorized. The OpenC2 language was designed to be agnostic of the other aspects of

cyber defense implementations that realize these assumptions. The following items are beyond the scope of this specification:

1. Language extensions applicable to some actuators
2. Alternate serializations of OpenC2 commands
3. The enumeration of the protocols required for transport, information assurance, sensing, analytics and other external dependencies

1.3 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>).

1.4 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD 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.

1.5 Document Conventions

Editor's Note - TBSL - This section will be included in a future iteration (probably iteration 5) prior to submitting for Committee Specification.

1.6 Naming Conventions

RFC2119/RFC8174 key words (see section 1.4) are in all uppercase.

All words in type names are capitalized. All property names and literals are in lowercase, except when referencing canonical names defined in another standard (e.g., literal values from an IANA registry). Words in property names are separated with an underscore (_), while words in string enumerations and type names are separated with a hyphen (-). All type names, property names, object names, and vocabulary terms are between three and 250 characters long.

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


1.7 Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <http://www.rfc-editor.org/info/rfc8174>.

2 OpenC2 Language

2.1 Overview

The OpenC2 language has two distinct message types: Command and Response. The OpenC2 Command describes an action performed on a target. The OpenC2 Response is a means to provide information (such as acknowledgement, status, etc.) as a result of an OpenC2 Command.

2.2 OpenC2 Command

The OpenC2 Command communicates an action to be performed on a target and may include information identifying the actuator(s) that is to execute the command.

2.2.1 Command Structure

An OpenC2 Command has four fields: ACTION, TARGET, ACTUATOR and COMMAND-OPTIONS.

The ACTION and TARGET fields are required and are populated by one of the 'action-types' in Table 2-1 and the 'target-types' in Table 2-2. A particular target-type may be further refined by one or more 'target-specifiers' and/or 'target-options'.

The optional ACTUATOR field identifies the entity or entities that are tasked to execute the OpenC2 Command.

Information with respect to how the action is to be executed is provided with one or more 'actuator-options'.

The optional COMMAND-OPTIONS field is populated by one or more 'command-options' that provide information that influences how the command is executed.

The following list summarizes the fields and subfields of an OpenC2 Command. OpenC2 Commands MUST contain an ACTION and TARGET and MAY contain an ACTUATOR and/or COMMAND-OPTIONS. OpenC2 is agnostic of any particular serialization; however, implementations MUST support JSON serialization of the commands.

- **ACTION** (required): The task or activity to be performed.
- **TARGET** (required): The object of the action. The ACTION is performed on the target.
 - **TARGET-NAME** (required): The name of the object of the action.
 - **TARGET-SPECIFIERS** (optional): The specifier further identifies the target to some level of precision, such as a specific target, a list of targets, or a class of targets.
 - **TARGET-OPTIONS** (optional): Additional information about how to perform the action for a specific target type.
- **ACTUATOR** (optional): The ACTUATOR may perform the ACTION on the TARGET. The ACTUATOR type will be defined within the context of an Actuator Profile.
 - **ACTUATOR-NAME** (required): The name of the set of functions (e.g., "firewall") performed by the actuator, and the name of the profile defining commands applicable to those functions.
 - **ACTUATOR-SPECIFIERS** (optional): The specifier identifies the actuator to some level of precision, such as a specific actuator, a list of actuators, or a group of actuators.

- **ACTUATOR-OPTIONS** (optional): The options specify how a particular ACTION is to be performed for an actuator type.
- **COMMAND-OPTIONS** (optional): Provide additional information on how the command is to be performed, such as date/time, periodicity, duration etc. COMMAND OPTIONS only influence/ impact the command and are defined independently of any ACTION, ACTUATOR or TARGET.

The TARGET of an OpenC2 Command may include a set of targets of the same type, a range of targets, or a particular target. Specifiers for TARGETs are optional and provide additional precision for the target.

The OpenC2 ACTUATOR field identifies the entity(ies) that execute the ACTION on the TARGET. Specifiers for actuators refine the command so that a particular function, system, class of devices, or specific device can be identified. Actuator-options indicate how an action is to be done in the context of the actuator.

Actuator is optional. One case where the Actuator is not specified is the case if the transport provides the mutual authentication so the OpenC2 Producer and Consumer both know the Consumer is the Actuator. One example of this would be an https API with mutual authentication. Another example may be a pub/sub such as OpenDXL. Another case where the actuator is not specified is when 'effects-based actions' are being used such as across trust boundaries - i.e., the Producer says the effect desired (e.g., deny ip, mitigate domain, etc.) but leaves it up to decision making in the OpenC2 Consumer to determine what actuator to use to achieve the desired effect.

COMMAND-OPTIONS influence the command by providing information such as time, periodicity, duration, or other details on what is to be executed. They can also be used to convey the need for acknowledgement or additional status information about the execution of a command.

2.2.2 Action Vocabulary

This section defines the set of OpenC2 actions grouped by their general activity. Table 2-1 summarizes the definition of the OpenC2 actions.

- *Actions that Control Information:* These actions are used to gather information needed to determine the current state or enhance cyber situational awareness.
- *Actions that Control Access:* These actions are used to control traffic flow and file permissions (e.g., allow/deny).
- *Actions that Control Activities/Devices:* These actions are used to control the state or the activity of a system, a process, a connection, a host, or a device. The actions are used to execute tasks, adjust configurations, set and update parameters, and modify attributes.
- *Effects-Based Actions:* Effects-based actions are at a higher level of abstraction for purposes of communicating a desired impact rather than a command to execute specific tasks. This level of abstraction enables coordinated actions between enclaves, while permitting a local enclave to optimize its workflow for its specific environment. Effects-based action assumes that the recipient enclave has a decision-making capability because effects-based actions typically do not have a one-to-one mapping to the other actions.

Editor's Note - This table is largely duplicated in Section 3. The editors plan to defer comments about duplication of tables between Sections 2 and 3 until after enough of the spec is complete to see how to correctly organize it.

Table 2-1. Summary of Action Definitions

Action	Description
Actions that Control Information	
scan	Systematic examination of some aspect of the entity or its environment in order to obtain information.
locate	Find an object either physically, logically, functionally, or by organization.
query	Initiate a request for information.
report	Task an entity to provide information to a designated recipient of the information.
notify	Set an entity's alerting preferences.
Actions that Control Access	
deny	The deny action is used to prevent a certain event or action from completion, such as preventing a flow from reaching a destination (e.g., block) or preventing access.
contain	Isolate a file, process, or entity such that it cannot modify or access assets or processes.
allow	Permit access to or execution of a target.
Actions that Control Activities/Devices	
start	Initiate a process, application, system, or some other activity.
stop	Halt a system or ends an activity.
restart	Stop then start a system or an activity.
pause	Cease a system or activity while maintaining state.
resume	Start a system or activity from a paused state.
cancel	Invalidate a previously issued action.
set	Change a value, configuration, or state of a managed entity within an IT system.
update	Instruct a component to retrieve, install, process, and operate in accordance with a software update, reconfiguration, or some other update.
move	Change the location of a file, subnet, network, or process.
redirect	Change the flow to a particular destination other than its original intended destination.
create	Add a new entity of a known type (e.g., data, files, directories).
delete	Remove an entity (e.g., data, files, flows).
snapshot	Record and store the state of a target at an instant in time.
detonate	Execute and observe the behavior of a target (e.g., file, hyperlink) in an isolated environment.
restore	Return a system to a previously known state.
save	Commit data or system state to memory.

Action	Description
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 data, content, or payload.
copy	Duplicate a file or data flow.
sync	Synchronize a sensor or actuator with other system components.

Effects-Based Actions

investigate	Task the recipient to aggregate and report information as it pertains to a security event or incident.
mitigate	Task the recipient to circumvent a problem without necessarily eliminating the vulnerability or attack point.
remediate	Task the recipient to eliminate a vulnerability or attack point.

2.2.3 Target Vocabulary

The TARGET is the object of the ACTION (or alternatively, the ACTION is performed on the TARGET). The baseline set of TARGETs is summarized in Table 2-2 and a full description of the targets and their associated specifiers is documented in the property tables (TBSL).

Editor's Note - This table is largely duplicated in Section 3. The editors plan to defer comments about duplication of tables between Sections 2 and 3 until after enough of the spec is complete to see how to correctly organize it.

Table 2-2. Summary of Targets.

Target	Description
artifact	An array of bytes representing a file-like object or a link to that object.
command	The Command Object represents a reference to a previously issued OpenC2 Command.
device	The Device Object represents the properties of a hardware or virtual device.
directory	The Directory Object represents the properties common to a file system directory.
disk	The Disk Object represents a disk drive.
disk_partition	The Disk Partition Object represents a single partition of a disk drive.
domain_name	The Domain Name represents the properties of a network domain name.
email_addr	The Email Address Object represents a single email address.
email_message	The Email Message Object represents an instance of an email message, corresponding to the internet message format described in RFC 5322 and related RFCs.

Target	Description
file	The File Object represents the properties of a file.
ipv4_addr	The IPv4 Address Object represents one or more IPv4 addresses expressed using CIDR notation.
ipv6_addr	The IPv6 Address Object represents one or more IPv6 addresses expressed using CIDR notation.
mac_addr	The MAC Address Object represents a single Media Access Control (MAC) address.
memory	The Memory Object represents memory objects.
ip_connection	The IP Connection Object represents a network connection that originates from a source and is addressed to a destination.
openc2	The OpenC2 object is the summation of the actions, targets and profiles supported by the actuator. The target is used with the query action to determine an actuator's capabilities.
process	The Process Object represents common properties of an instance of a computer program as executed on an operating system.
software	The Software Object represents high-level properties associated with software, including software products.
url	The URL Object represents the properties of a uniform resource locator (URL).
user_account	The User Account Object represents an instance of any type of user account, including but not limited to operating system, device, messaging service, and social media platform accounts.
user_session	The User Session Object represents a user session.
volume	The Volume Object represents a generic drive volume.
windows_registry_key	The Registry Key Object represents the properties of a Windows registry key.
x509_certificate	The X509 Certificate Object represents the properties of an X.509 certificate, as defined by ITU recommendation X.509.

Editor's Note - There is agreement that targets be extensible. That is, if an implementer has a target that is not yet in the language, the extensibility would be used. Several alternatives are under considerations so the exact text to go here is still under development.

2.2.4 Actuator

An ACTUATOR is an implementation of a cyber defense function that executes the ACTION on the TARGET. An Actuator Profile is a specification that identifies the subset of ACTIONS, TARGETS and other aspects of this language specification that are mandatory to implement or optional in the context of a particular ACTUATOR. An Actuator Profile also defines ACTUATOR-SPECIFIERS and ACTUATOR-OPTIONS that are meaningful and possibly unique to the actuator.

An Actuator Profile SHALL be composed in accordance with the framework in section 4.

Editor's Note - TBSL - More text be included in a future iteration (probably iteration 4) prior to submitting for Committee Specification.

2.2.5 Command-Option Vocabulary

COMMAND-OPTIONS influence a command and are independent of the TARGET, ACTUATOR and ACTION itself. COMMAND-OPTIONS provide additional information to refine how the command is to be performed such as time, periodicity, or duration, or convey the need for status information such as a response is required. The requested status/information will be carried in a RESPONSE.

Table 2-3 lists the valid command-options.

Editor's Note - This table is largely duplicated in Section 3. The editors plan to defer comments about duplication of tables between Sections 2 and 3 until after enough of the spec is complete to see how to correctly organize it.

Table 2-3. Summary of Command Options.

Command Option	Description
start_time	The specific date/time to initiate the action
stop_time	The specific date/time to terminate the action
duration	The length of time for an action to be in effect
response_requested	Indicate the type of response required for the action

Editor's Note - Additional usage guidance for these command options will be included in a future working draft.

2.2.6 Imported Data

Editor's Note - This section was previously titled "Extensibility".

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

- i. a prefix indication the origin of the imported data object such as:
 1. ap- (actuator profile)
 2. ip- (implementation profile)
 3. vp- (vendor specification)
 4. fs- (external specification)
- ii. a unique name for the specification being imported, e.g., /docs.oasis-open.org/kmip/spec/v1.4/kmip-spec-v1.4
- iii. a namespace identifier - a short reference to the specification, e.g. kmip_1.4
- iv. a list of object identifiers imported from that specification, e.g., Credential
- v. a definition of each imported object, either referenced or contained in the profile
- vi. 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.

A data profile can define imported objects using an abstract syntax, or it can reference content as defined in the specification being imported.

An imported object is identified by namespace and object ids:

```
"target": {
  "fs-kmip_1.4": {
    "Credential": {
      "uid_pwd": {
        "Username": "johndoe",
        "Password": "MyBigS3cret"
      }
    }
  }
}
```

2.3 OpenC2 Response

The OpenC2 Response is a message sent from an entity as the result of a command. Response messages provide acknowledgement, status, results from a query, or other information as requested from the issuer of the command. Response messages are solicited and correspond to a command.

2.3.1 Response Structure

The following list summarizes the fields and subfields of an OpenC2 Response. OpenC2 Responses **MUST** contain an **STATUS** and **MAY** contain an **STATUS_TEXT** and/or **RESULTS**. OpenC2 is agnostic of any particular serialization; however, implementations **MUST** support JSON serialization of the responses.

- **STATUS** (required): An integer containing a numerical status code
- **STATUS_TEXT** (optional): A free-form string containing human-readable description of the response status. The string can contain 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 that was requested from an OpenC2 Command. If not present, the status code is a sufficient response.

3 OpenC2 Property Tables

3.1 Terminology

The syntax of valid OpenC2 messages is defined using the following datatypes:

Type	Description
Primitive Types	
Binary	A sequence of octets or bytes. Serialized either as binary data or as a string using an encoding such as hex or base64.
Boolean	A logical entity that can have two values: true and false. Serialized as either integer or keyword.
Date-Time	TBD, RFC XXXX
Integer	A number that can be written without a fractional component. Serialized either as binary data or a text string.
Number	A real number. Valid values include integers, rational numbers, and irrational numbers. Serialized as either binary data or a text string.
String	A sequence of characters. Each character must have a valid Unicode codepoint.
Structures	
Array	An ordered list of unnamed fields. Each field has an ordinal position and a type. Serialized as a list.
ArrayOf	An ordered list of unnamed fields of the same type. Each field has an ordinal position and must be the specified type. Serialized as a list.
Choice	One field selected from a set of named fields. The value has a name and a type. Serialized as a one-element map.
Enumerated	A set of id:name pairs. Serialized as either the integer id or the name string.
Map	An unordered set of named fields. Each field has a name and a type. Serialized as a mapping type (referred to in various programming languages as: associative array, dict, dictionary, hash, map, object).
Record	An ordered list of named fields, e.g. a message, record, structure, or row in a table. Each field has an ordinal position, a name, and a type. Serialized as either a list or a map.

3.2 OpenC2 Messages

The following subsections provide the permitted values within an OpenC2 message.

3.2.1 OpenC2 Command

The OpenC2 Command describes an action performed on a target. It can be directive or descriptive depending on the context.

3.2.1.1 Type Name: OpenC2-Command

Base Type: Record

ID	Property Name	Type	Description
1	id (required)	Command-ID	Identifier used to link responses to a command
2	action (required)	Action	The task or activity to be performed (i.e., the 'verb')
3	target (required)	Target	The object of the action. The action is performed on the target.
4	actuator (optional)	Actuator	The subject of the action. The actuator executes the action on the target.
5	options (optional)	Command-Options	An object containing additional properties that apply to the command

Editor's Note - In a future working draft, we may reformat these tables to include a cardinality column instead of the required/optional tags on the property names.

3.2.1.2 Type Name: Action

Base Type: Enumerated

ID	Property Name	Description
1	scan	Systematic examination of some aspect of the entity or its environment in order to obtain information.
2	locate	Find an object either physically, logically, functionally, or by organization.
3	query	Initiate a request for information.
4	report	Task an entity to provide information to a designated recipient of the information.
5	notify	Set an entity's alerting preferences.
6	deny	The deny action is used to prevent a certain event or action from completion, such as preventing a flow from reaching a destination (e.g., block) or preventing access.
7	contain	Isolate a file, process, or entity such that it cannot modify or access assets or processes.
8	allow	Permit access to or execution of a target.
9	start	Initiate a process, application, system, or some other activity.
10	stop	Halt a system or ends an activity.

ID	Property Name	Description
11	restart	Stop then start a system or an activity.
12	pause	Cease a system or activity while maintaining state.
13	resume	Start a system or activity from a paused state.
14	cancel	Invalidate a previously issued action.
15	set	Change a value, configuration, or state of a managed entity within an IT system.
16	update	Instruct a component to retrieve, install, process, and operate in accordance with a software update, reconfiguration, or some other update.
17	move	Change the location of a file, subnet, network, or process.
18	redirect	Change the flow to a particular destination other than its original intended 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).
21	snapshot	Record and store the state of a target at an instant in time.
22	detonate	Execute and observe the behavior of a target (e.g., file, hyperlink) in an isolated environment.
23	restore	Return a system to a previously known state.
24	save	Commit data or system state to memory.
25	throttle	Adjust the rate of a process, function, or activity.
26	delay	Stop or hold up an activity or data transmittal.
27	substitute	Replace all or part of the data, content, or payload.
28	copy	Duplicate a file or data flow.
29	sync	Synchronize a sensor or actuator with other system components.
30	investigate	Task the recipient to aggregate and report information as it pertains to a security event or incident.
31	mitigate	Task the recipient to circumvent a problem without necessarily eliminating the vulnerability or attack point.
32	remediate	Task the recipient to eliminate a vulnerability or attack point.

3.2.1.3 Type Name: Target

Base Type: Choice

ID	Property Name	Type	Description
1	artifact	Artifact	An array of bytes representing a file-like object or a link to that object.

ID	Property Name	Type	Description
2	command	Command	The Command Object represents a reference to a previously issued OpenC2 Command.
3	device	Device	The Device Object represents the properties of a hardware device.
4	directory	Directory	The Directory Object represents the properties common to a file system directory.
5	disk	Disk	The Disk Object represents a disk drive.
6	disk_partition	Disk-Partition	The Disk Partition Object represents a single partition of a disk drive.
7	domain_name	Domain-Name	The Domain Name represents the properties of a network domain name.
8	email_addr	Email-Addr	The Email Address Object represents a single email address.
9	email_message	Email-Message	The Email Message Object represents an instance of an email message, corresponding to the internet message format described in RFC 5322 and related RFCs.
10	file	File	The File Object represents the properties of a file.
11	ipv4_addr	IPv4-Addr	The IPv4 Address Object represents one or more IPv4 addresses expressed using CIDR notation.
12	ipv6_addr	IPv6-Addr	The IPv6 Address Object represents one or more IPv6 addresses expressed using CIDR notation.
13	mac_addr	Mac-Addr	The MAC Address Object represents a single Media Access Control (MAC) address.
14	memory	Memory	The Memory Object represents memory objects.
15	ip_connection	IP-Connection	The IP Connection Object represents a network connection that originates from a source and is addressed to a destination.
16	openc2	OpenC2	The OpenC2 object is the summation of the actions, targets and profiles supported by the actuator. The target is used with the query action to determine an actuator's capabilities.
17	process	Process	The Process Object represents common properties of an instance of a computer program as executed on an operating system.

ID	Property Name	Type	Description
18	software	Software	The Software Object represents high-level properties associated with software, including software products.
19	url	Url	The URL Object represents the properties of a uniform resource locator (URL).
20	user_account	User-Account	The User Account Object represents an instance of any type of user account, including but not limited to operating system, device, messaging service, and social media platform accounts.
21	user_session	User-Session	The User Session Object represents a user session.
22	volume	Volume	The Volume Object represents a generic drive volume.
23	windows_registry_key	Windows-Registry-Key	The Registry Key Object represents the properties of a Windows registry key.
24	x509_certificate	X509-Certificate	The X509 Certificate Object represents the properties of an X.509 certificate, as defined by ITU recommendation X.509.

3.2.1.4 Type Name: Actuator

Base Type: Choice

ID	Property Name	Type	Description
1	TBSL	TBSL	TBSL
2	TBSL	TBSL	TBSL

Editor's Note - The intent is to fill in this table with actuators as they are defined by the AP-SC. The AP-SC profiles will define the actuators and they will only be listed here. Once we have a lot of them (not an issue yet), we may figure out how to just put a reference here to a list maintained by the AP-SC.

Editor's Note - The intent is to for the actuators to be extensible. Ie if a vendor has a function that is not yet in an AP-SC profile, the extensibility would be used to add this new function. The text to go here on how to do that is still under development

3.2.1.5 Type Name: Command-Options

Base Type: Record

ID	Property Name	Type	Description
1	start_time (optional)	Date-Time	The specific date/time to initiate the action

ID	Property Name	Type	Description
2	stop_time (optional)	Date-Time	The specific date/time to terminate the action
3	duration (optional)	Duration	The length of time for an action to be in effect
4	response_requested (optional)	Response-Type	Indicate the type of response required for the action

Editor's Note - version is agreed to be needed. It will not appear directly in the OpenC2 Command, instead it will appear in a "header" field of an OpenC2 Message. The OpenC2 Message is a wrapper for an OpenC2 Command or OpenC2 Response. It is still being deliberated where and how the OpenC2 Message will be documented. It may be documented in this Language Specification or within another standalone specification developed by the Implementation Considerations Subcommittee.

3.2.2 OpenC2 Response

3.2.2.1 Type Name: OpenC2-Response

Base Type: Record

ID	Property Name	Type	Description
1	id (required)	Command-ID	Id of the command that induced this response
2	status (required)	Status-Code	An integer containing a numerical status code
3	status_text (optional)	String	A free-form string containing human-readable description of the response status
4	results (optional)	Results	Contains the data or extended status information that was requested from an OpenC2 Command

Example:

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

3.2.2.2 Type Name: Status-Code

Base Type: Enumerated

Value	Description
102	Processing - an interim response used to inform the client that the server 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.

Value	Description
400	Bad Request - the server cannot process the request due to something that is perceived to be a client error (e.g., malformed request syntax).
401	Unauthorized - the request lacks valid authentication credentials for the target resource or authorization has been refused for the submitted credentials.
403	Forbidden - the server understood the request but refuses to authorize it.
500	Server Error - the server encountered an unexpected condition that prevented it from fulfilling the request.
501	Not Implemented - the server does not support the functionality required to fulfill the request.

3.3 Property Details

Editor's Note - The organization of this section will get redone once more property tables exist (probably iterations 5) prior to submitting for Committee Specification. For now placeholder section numbers will be used

3.3.0

3.3.0.1 Type Name: IP-Connection

Base Type: Record

ID	Property Name	Type	Description
1	src_addr	IP-Addr	ip_addr of source, could be ipv4 or ipv6 - see ip_addr section
2	src_port	Port	source service per RFC TBSL
3	dst_addr	IP-Addr	ip_addr of destination, could be ipv4 or ipv6 - see ip_addr section
4	dst_port	Port	destination service per RFC TBSL
5	protocol	L4-Protocol	layer 4 protocol (e.g., TCP) - see l4_protocol section

3.3.0.2 Type Name: IP-Addr

Type Name	Type	Description
IP-Addr	String	IPv4 or IPv6 address or range in CIDR notation. IPv4 address or range in CIDR notation, i.e., a dotted decimal format per RFC TBSL with optional CIDR prefix. IPv6 address or range in CIDR notation, i.e., colon notation per RFC 5952 with optional CIDR prefix

Examples:

- "192.168.10.11" - a single ipv4 address distinguishable because of the dots

- "192.168.10.11/32" - a single ipv4 address in CIDR notation
- "192.168.0.0/16" - a range of 65,536 ipv4 addresses in CIDR notation
- "2001:db8::1" - a single ipv6 address distinguishable because of the colons
- "2001:db8:aaaa:bbbb:cccc:dddd:0:1" - single ipv6 address
- "2001:db8::0/120" - 256 ipv6 addresses

Examples of invalid ipv6 (since violates RFC 5952):

- "2001:DB8::1" - lower case MUST be used
- "2001:db8:0:0:1:0:0:1" - the :: notation MUST be used for zero compression when possible
- "2001:db8::1:1:1:1:1" - the :: notation MUST NOT be used when only one zero is present

3.3.0.3 Type Name: Port

Type Name	Type	Description
Port	String	Service Name or Transport Protocol Port Number, RFC 6335

3.3.0.4 Type Name: L4-Protocol

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

ID	Property Name	Description
1	icmp	Internet Control Message Protocol - RFC 792
6	tcp	Transmission Control Protocol - RFC 793
17	udp	User Datagram Protocol - RFC 768
132	sctp	Stream Control Transmission Protocol - RFC 4960

3.3.0.5 Type Name: File

Base Type: Record

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

3.3.0.6 Type Name: Response-Requested

Base Type: Choice

ID	Name	Type	Description
0	None	TBSL	No response
1	Ack	TBSL	Respond when command received
2	Complete	TBSL	Respond when all aspects of command completed
3	TBSL	TBSL	TBSL
4	TBSL	TBSL	TBSL

Editor's Note - Use cases are needed for the different types of responses needed.

3.3.0.7 Type Name: Command-ID

Type Name	Type	Description
Command-ID	Identifier	Uniquely identifies a particular command

3.3.0.8 Type Name: Identifier

Type Name	Type	Description
Identifier	string = command--UUIDv4	An identifier universally and uniquely identifies an OpenC2 command. Value SHOULD be a UUID generated according to RFC 4122.

3.3.0.9 Type Name: Version

Type Name	Type	Description
Version	String	TBSL

Editor's Note - version is agreed to be needed. It will not appear directly in the OpenC2 Command, instead it will appear in a "header" field of an OpenC2 Message. The OpenC2 Message is a wrapper for an OpenC2 Command or OpenC2 Response. It is still being deliberated where and how the OpenC2 Message will be documented. It may be documented in this Language Specification or within another standalone specification developed by the Implementation Considerations Subcommittee.

3.3.0.10 Type Name: Domain-Name

Type Name	Type	Description
Domain-Name	String	per RFC 1034

3.3.0.11 Type Name: Email-Message

Type Name	Type	Description
Email-Message	String	per RFC TBSL

3.3.0.12 Type Name: Process

Base Type: Map

Property Name	Type	Description
pid (optional)	Integer	Process ID of the process
name (optional)	String	Name of the process
cwd (optional)	String	Current working directory of the process
executable (optional)	File	Executable that was executed to start the process
parent (optional)	Process	Process that spawned this one
command_line (optional)	String	The full command line invocation used to start this process, including all arguments

3.3.0.13 Type Name: Hashes

Base Type: Map

Property Name	Type	Description
md5 (optional)	String	Hex-encoded MD5 hash as defined in RFC 1321
sha1 (optional)	String	Hex-encoded SHA1 hash as defined in RFC 6234
sha256 (optional)	String	Hex-encoded SHA256 hash as defined in RFC 6234

3.3.0.14 Type Name: Hostname

Type Name	Type	Description
Hostname	String	A legal Internet host name as specified in RFC 1123

3.3.0.15 Type Name: Device

Base Type: Map

Property Name	Type	Description
hostname (optional)	Hostname	A hostname that can be used to connect to this device over a network
description (optional)	String	A human-readable description of the purpose, relevance, and/or properties of this device
device_id (optional)	String	An identifier that refers to this device within an inventory or management system

4 Foundational Actuator Profile

Editor's Note - TBSL - This section be included in a future iteration (probably iteration 5) prior to submitting for Committee Specification.

5 Conformance

OpenC2 is a command and control language that converges (i.e., common 'point of understanding') on a common syntax, and lexicon. The tables in Section 3 of this document specify the normative rules for determining if an OpenC2 message (command or response) is syntactically valid. All examples in this document are informative; in case of conflict between the tables and an example, the tables are authoritative. Conformant implementations of OpenC2:

- MUST produce messages that are syntactically valid.
- SHOULD reject messages that are syntactically invalid.
- MUST implement the actions designated as mandatory in this document.
- MUST implement the targets designated as mandatory in this document.
- MAY implement optional targets defined in this document
- MAY implement actuator specifiers, actuator options, target specifiers and/or target options as specified in one or more Actuator Profiles.
- MUST implement JSON serialization of the commands and responses that are consistent with the syntax defined in this document.

Editor's Note - TBSL - More conformance text will be included in a future iteration (probably iteration 5) prior to submitting for Committee Specification.

Appendix A. Acknowledgments

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

Participants:

Editor's Note - TBSL - This section be included in the final iteration prior to submitting for Committee Specification.

Appendix B. Revision History

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
v1.0-wd02	01/12/2018	Romano, Sparrell	csd01 ballot comments
v1.0-wd03		Romano, Sparrell	wd02 review comments
v1.0-csd02		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

Appendix C. Acronyms

Editor's Note - TBSL - This section be included in the final iteration prior to submitting for Committee Specification.

Appendix D. Examples

Editor's Note - TBSL - This section will be populated with examples of json command and responses. The intent is to have each example serve multiple purposes (e.g., one example shows action=allow, command option=start_time, target=....) and then could be referenced with footnotes from several places in spec. This original draft was quite long due to all the inline examples and this is hoped to be a reasonable compromise

Example 1:

Editor's Note - This example shows the structure of an OpenC2 Message containing a header and a command. The command shows the recently relocated command ID field. The structure of the options is still being deliberated.

```
{  "header": {
    "version": "1.0",
    "timestamp": "2018-01-30T18:25:43.511Z"
  },
  "command": {
    "id": "CMD1234",
    "action": "redirect",
    "target": {
      "url": {
        "value": "http://evil.com"
      },
      "options": {
        "destination": "http://newdest.com/home"
      }
    }
  }
}
```