Open Command and Control (OpenC2) Language Specification Version 1.0

Committee Specification Draft 01

14 November 2017

Specification URIs

This version:

http://docs.oasis-open.org/openc2/oc2ls/v1.0/csd01/oc2ls-v1.0-csd01.pdf (Authoritative)
http://docs.oasis-open.org/openc2/oc2ls/v1.0/csd01/oc2ls-v1.0-csd01.html
http://docs.oasis-open.org/openc2/oc2ls/v1.0/csd01/oc2ls-v1.0-csd01.docx

Previous version:

N/A

Latest version:

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

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

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

Citation format:

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

[OpenC2-Lang-v1.0]

Open Command and Control (OpenC2) Language Specification Version 1.0. Edited by Jason Romano and Duncan Sparrell. 14 November 2017. OASIS Committee Specification Draft 01. http://docs.oasis-open.org/openc2/oc2ls/v1.0/csd01/oc2ls-v1.0-csd01.html. Latest version: http://docs.oasis-open.org/openc2/oc2ls/v1.0/oc2ls-v1.0.html.

Notices

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

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

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

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

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

The OpenC2 Language Specification defines a language used to compose messages that instruct and coordinate the command and control of cyber defenses between and within networks and systems.

An OpenC2 command is composed of an action (what is to be done), a target (what is being acted upon), an optional actuator (what is executing the command), and command options, which influence how the command is to be performed.

A OpenC2 command that consists of an action coupled with a target is sufficient for a high-level effects-based command (e.g., mitigate evildomain.com). The inclusion in the command of an actuator and modifiers provides additional precision and specificity (e.g., deny ip=1.2.3.4 by actuator=firewall3 command-id=1eab14...). Additional detail about aspects of a command 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 synchronously issued as a result of an OpenC2 command. OpenC2 responses are used to provide acknowledgement, status, results of a command or other information in conjunction with a particular command.

1.1 Goal

TBSL

1.2 Purpose and Scope

The OpenC2 Language Specification defines the set of components to assemble a complete command and control message capability and provide a framework so that the language can be extended to accommodate new technologies. 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. an organizational scheme that describes an actuator profile.
4. a syntax to express commands and responses.
5. the serialization of OpenC2 commands, and responses.
6. the procedures for extending the language to accommodate new technologies in a manner that is consistent with the OpenC2 Language Specification.

The OpenC2 language is necessary but insufficient for the realization of coordinated cyber response. Though necessary for cyber-response implementations, the following items are beyond the scope of this specification:

1. Language definitions for a particular actuator to extend the OpenC2 language. Extensions to the language will be captured in other specifications.
2. Specifying alternate serializations of OpenC2 commands. However, optional serializations may be documented in other specifications.
3. The enumeration of the protocols required for transport, information assurance, sensing, analytics and other external dependencies. 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.

1.3 IPR Policy

This Committee Specification Draft 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].

1.5 Document Conventions

TBSL

1.6 Naming Conventions

All type names, 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 type names and string enumerations 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 types of messages: Command and Response. At the most basic level, the OpenC2 Command describes an action performed on a target. The OpenC2 Response is used to provide execution status and optional data requested as a result of a command. OpenC2 Response messages may refer to the command that initiated the response.

2.2 OpenC2 Command

The OpenC2 Command communicates an action to be performed on a target and may include the actuator 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 TBD. 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.

Table 2-1 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.

Table 2-1. OpenC2 Command Field Descriptions

Field	Description
ACTION	Required. The task or activity to be performed.
TARGET	Required. The object of the action. The ACTION is performed on the TARGET.
type	Required. The specific type of TARGET.
target-specifier	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-option	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.
type	Required if the actuator is included, otherwise not applicable. The ACTUATOR type will be defined within the context of an actuator profile.
actuator-specifier	Optional if the actuator is included, otherwise not applicable. 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-option	Optional if the actuator is included, otherwise not applicable. Actuator-option identifies how a particular action is to be done for an actuator type.
COMMAND-OPTIONS(<list-of-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 provides information about the entity that will execute the ACTION on the TARGET. Specifiers for actuators provide additional information to refine the command so that a particular function, system, class of devices, or specific device can be identified. Options for actuators provide additional information to refine the command to indicate how an action is to be done in the context of the actuator. Options are distinct from COMMAND-OPTIONS in that options are a function of the actuator and the action.

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 2-2 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 Permissions:
  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.

Table 2-2. Summary of Action Definitions

Action	Description
Actions that Control Information	
scan	The ‘scan’ action is the systematic examination of some aspect of the entity or its environment in order to obtain information.
locate	The ‘locate’ action is used to find an object either physically, logically, functionally, or by organization.
query	The ‘query’ action initiates a single request for information.
report	The ‘report’ action tasks an entity to provide information to a designated recipient of the information.
notify	The ‘notify’ action is used to set an entity's alerting preferences.
Actions that Control Permissions	
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	The ‘contain’ action stipulates the isolation of a file, process, or entity such that it cannot modify or access assets or processes that support the business and/or operations of the enclave.
allow	The ‘allow’ action permits the access to or execution of a target.

Action	Description
Actions that Control Activities/Devices	
start	The ‘start’ action initiates a process, application, system, or some other activity.
stop	The ‘stop’ action halts a system or ends an activity.
restart	The ‘restart’ action conducts a ‘stop’ of a system or an activity followed by a ‘start’ of a system or an activity.
pause	The ‘pause’ action ceases a system or activity while maintaining state.
resume	The ‘resume’ action starts a system or activity from a paused state.
cancel	The ‘cancel’ action invalidates a previously issued action.
set	The ‘set’ action changes a value, configuration, or state of a managed entity within an IT system.
update	The ‘update’ action instructs the component to retrieve, install, process, and operate in accordance with a software update, reconfiguration, or some other update.
move	The ‘move’ action changes the location of a file, subnet, network, or process.
redirect	The ‘redirect’ action changes the flow to a particular destination other than its original intended destination.
create	The ‘create’ action adds a new entity (e.g., data, files, directories, security entities, etc.).
delete	The ‘delete’ action removes an entity (e.g., data, files, flows, etc.).
snapshot	The ‘snapshot’ action records and stores the state of a target at an instant in time.
detonate	The ‘detonate’ action executes and observes the behavior of a target (e.g., file, hyperlink) in a manner that is isolated from assets that support the business or operations of the enclave.
restore	The ‘restore’ action returns to an identical or similar known state.
save	The ‘save’ action commits data or system state to memory.
throttle	The ‘throttle’ action adjusts the rate of a process, function, or activity.
delay	The ‘delay’ action stops or holds up an activity or data transmittal.
substitute	The ‘substitute’ action replaces all or part of the data, content, or payload.
copy	The ‘copy’ action duplicates a file or data flow.
sync	The ‘sync’ action synchronizes a sensor or actuator with other system components.
Effects-Based Actions	
investigate	The ‘investigate’ action tasks the recipient enclave to aggregate and report information as it pertains to an anomaly.
mitigate	The ‘mitigate’ action tasks the recipient enclave to circumvent the problem without necessarily eliminating the vulnerability or attack point.
remediate	The ‘remediate’ action tasks the recipient enclave to eliminate the vulnerability or attack point. Remediate implies that addressing the issue is paramount.

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 2-3 and a full description of the targets and their associated specifiers is documented in the property tables (TBSL).

Table 2-3. Summary of Target Definitions.

Target	Description
TBSL	TBSL

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 meaningful in the context of a particular ACTUATOR. The actuator profile also identifies the portions of this specification that are mandatory to implement as well as optional actions and also defines appropriate actuator specifiers and the actuator options.

An Actuator Profile SHALL be composed in accordance with the following framework: TBSL.

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 2-4 lists the valid modifiers.

Table 2-4. Summary of Command Options.

Command Option	Type	Description
TBSL	TBSL	TBSL

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. The recipient of the OpenC2 Response is typically the entity that issued the command.

2.3.1 Response Structure

TBSL

3 OpenC2 Property Tables

TBSL

4 Foundational Actuator Profile

TBSL

5 Conformance

OpenC2 is a command and control language that converges (i.e. common ‘point of understanding’) on a common syntax, and lexicon. OpenC2 does not have a dependency on a particular programming language, computing platform, transport protocol etc.. Conformant implementations of OpenC2:

- MUST support OpenC2 commands, responses and alerts as defined in this document.
- 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, responses and alerts that are consistent with the syntax defined in this document.
- TBSL

Appendix A. Acknowledgments

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

Participants:

TBSL

Appendix B. Revision History

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