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]

Table of Contents

1 Introduction .................................................................................................................................................. 5
  1.1 Goal ......................................................................................................................................................... 5
  1.2 Purpose and Scope ................................................................................................................................ 5
  1.3 IPR Policy ............................................................................................................................................... 6
  1.4 Terminology .......................................................................................................................................... 6
  1.5 Document Conventions .......................................................................................................................... 6
  1.6 Naming Conventions ............................................................................................................................... 6
  1.7 Normative References .............................................................................................................................. 6

2 OpenC2 Language ...................................................................................................................................... 7
  2.1 Overview ............................................................................................................................................... 7
  2.2 OpenC2 Command .................................................................................................................................. 7
    2.2.1 Command Structure .......................................................................................................................... 7
    2.2.2 Action Vocabulary ........................................................................................................................... 8
    2.2.3 Target Vocabulary ............................................................................................................................ 9
    2.2.4 Actuator .......................................................................................................................................... 9
    2.2.5 Command Option Vocabulary .......................................................................................................... 10
  2.3 OpenC2 Response .................................................................................................................................. 10
    2.3.1 Response Structure ......................................................................................................................... 10

3 OpenC2 Property Tables .............................................................................................................................. 11

4 Foundational Actuator Profile .................................................................................................................. 12

5 Conformance ............................................................................................................................................. 13

Appendix A. Acknowledgments ..................................................................................................................... 14
Appendix B. Revision History ........................................................................................................................ 15
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”, “SHALT 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.

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

1.7 Normative References


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

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTION</td>
<td>Required. The task or activity to be performed.</td>
</tr>
<tr>
<td>type</td>
<td>Required. The specific type of TARGET.</td>
</tr>
<tr>
<td>target-specifier</td>
<td>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.</td>
</tr>
<tr>
<td>target-option</td>
<td>Optional. Additional information about how to perform the action for a specific target type.</td>
</tr>
<tr>
<td>ACTUATOR</td>
<td>Optional. The ACTUATOR may perform the ACTION on the TARGET.</td>
</tr>
<tr>
<td>type</td>
<td>Required if the actuator is included, otherwise not applicable. The ACTUATOR type will be defined within the context of an actuator profile.</td>
</tr>
<tr>
<td>actuator-specifier</td>
<td>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.</td>
</tr>
<tr>
<td>actuator-option</td>
<td>Optional if the actuator is included, otherwise not applicable. Actuator-option identifies how a particular action is to be done for an actuator type.</td>
</tr>
<tr>
<td>COMMAND-OPTIONS</td>
<td>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.</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Actions that Control Information</strong></td>
<td></td>
</tr>
<tr>
<td>scan</td>
<td>The ‘scan’ action is the systematic examination of some aspect of the entity or its environment in order to obtain information.</td>
</tr>
<tr>
<td>locate</td>
<td>The ‘locate’ action is used to find an object either physically, logically, functionally, or by organization.</td>
</tr>
<tr>
<td>query</td>
<td>The ‘query’ action initiates a single request for information.</td>
</tr>
<tr>
<td>report</td>
<td>The ‘report’ action tasks an entity to provide information to a designated recipient of the information.</td>
</tr>
<tr>
<td>notify</td>
<td>The ‘notify’ action is used to set an entity’s alerting preferences.</td>
</tr>
<tr>
<td><strong>Actions that Control Permissions</strong></td>
<td></td>
</tr>
<tr>
<td>deny</td>
<td>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.</td>
</tr>
<tr>
<td>contain</td>
<td>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.</td>
</tr>
<tr>
<td>allow</td>
<td>The ‘allow’ action permits the access to or execution of a target.</td>
</tr>
</tbody>
</table>
### Actions that Control Activities/Devices

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>The ‘start’ action initiates a process, application, system, or some other activity.</td>
</tr>
<tr>
<td>stop</td>
<td>The ‘stop’ action halts a system or ends an activity.</td>
</tr>
<tr>
<td>restart</td>
<td>The ‘restart’ action conducts a ‘stop’ of a system or an activity followed by a ‘start’ of a system or an activity.</td>
</tr>
<tr>
<td>pause</td>
<td>The ‘pause’ action ceases a system or activity while maintaining state.</td>
</tr>
<tr>
<td>resume</td>
<td>The ‘resume’ action starts a system or activity from a paused state.</td>
</tr>
<tr>
<td>cancel</td>
<td>The ‘cancel’ action invalidates a previously issued action.</td>
</tr>
<tr>
<td>set</td>
<td>The ‘set’ action changes a value, configuration, or state of a managed entity within an IT system.</td>
</tr>
<tr>
<td>update</td>
<td>The ‘update’ action instructs the component to retrieve, install, process, and operate in accordance with a software update, reconfiguration, or some other update.</td>
</tr>
<tr>
<td>move</td>
<td>The ‘move’ action changes the location of a file, subnet, network, or process.</td>
</tr>
<tr>
<td>redirect</td>
<td>The ‘redirect’ action changes the flow to a particular destination other than its original intended destination.</td>
</tr>
<tr>
<td>create</td>
<td>The ‘create’ action adds a new entity (e.g., data, files, directories, security entities, etc.).</td>
</tr>
<tr>
<td>delete</td>
<td>The ‘delete’ action removes an entity (e.g., data, files, flows, etc.).</td>
</tr>
<tr>
<td>snapshot</td>
<td>The ‘snapshot’ action records and stores the state of a target at an instant in time.</td>
</tr>
<tr>
<td>detonate</td>
<td>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.</td>
</tr>
<tr>
<td>restore</td>
<td>The ‘restore’ action returns to an identical or similar known state.</td>
</tr>
<tr>
<td>save</td>
<td>The ‘save’ action commits data or system state to memory.</td>
</tr>
<tr>
<td>throttle</td>
<td>The ‘throttle’ action adjusts the rate of a process, function, or activity.</td>
</tr>
<tr>
<td>delay</td>
<td>The ‘delay’ action stops or holds up an activity or data transmittal.</td>
</tr>
<tr>
<td>substitute</td>
<td>The ‘substitute’ action replaces all or part of the data, content, or payload.</td>
</tr>
<tr>
<td>copy</td>
<td>The ‘copy’ action duplicates a file or data flow.</td>
</tr>
<tr>
<td>sync</td>
<td>The ‘sync’ action synchronizes a sensor or actuator with other system components.</td>
</tr>
</tbody>
</table>

### Effects-Based Actions

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>investigate</td>
<td>The ‘investigate’ action tasks the recipient enclave to aggregate and report information as it pertains to an anomaly.</td>
</tr>
<tr>
<td>mitigate</td>
<td>The ‘mitigate’ action tasks the recipient enclave to circumvent the problem without necessarily eliminating the vulnerability or attack point.</td>
</tr>
<tr>
<td>remediate</td>
<td>The ‘remediate’ action tasks the recipient enclave to eliminate the vulnerability or attack point. Remediate implies that addressing the issue is paramount.</td>
</tr>
</tbody>
</table>

#### 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.**

<table>
<thead>
<tr>
<th>Target</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBSL</td>
<td>TBSL</td>
</tr>
</tbody>
</table>

#### 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>
<thead>
<tr>
<th>Command Option</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBSL</td>
<td>TBSL</td>
<td>TBSL</td>
</tr>
</tbody>
</table>

#### 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

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Editor</th>
<th>Changes Made</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1.0-wd01</td>
<td>10/31/2017</td>
<td>Romano, Sparrell</td>
<td>Initial working draft</td>
</tr>
</tbody>
</table>