Open Command and Control (OpenC2) Language Description Document

Version 1.0 – Release Candidate 4

27 March 2017

FOREWORD

The Open Command and Control Forum (OpenC2 or the Forum) supports the cyber defense community of interest by developing and promoting the adoption of the OpenC2 language, data models, prototype implementations, and reference material that addresses the command and control of cyber defense components, technologies, and systems.

This Forum serves developers, users, and the entire cybersecurity ecosystem by providing a set of shared resources to expand the use of standardized command and control for cyber defense activities, to enable technology vendors building orchestration and cyber response technologies, and to assist developers in producing response technologies that can be readily used in coordinated responses. The goal of the Forum is to provide an open and collaborative environment and to present its findings and artifacts to recognized standards bodies for the standardization of the command and control language.

This document represents the outcome of collaboration between technology vendors, government agencies, and academia on the topic of command and control for cyber defensive measures. We gratefully acknowledge their contributions to the definition of the OpenC2 language. As we exercise the language in reference implementations and in real-world operations, we expect to continue to refine the language to ensure its suitability to support machine-to-machine command and control communications in response to cyber threats in cyber-relevant time.

Visit openc2.org for other on-line resources.

TABLE OF CONTENTS

1. Introduction

Cyberattacks are increasingly more sophisticated, less expensive to execute, dynamic, and automated. Current cyber defense products are typically integrated in a unique or proprietary manner and statically configured. As a result, upgrading or otherwise modifying tightly integrated, proprietary cyber defense's functional blocks is resource intensive; cannot be realized within a cyber-relevant timeframe; and the upgrades may degrade the overall performance of the system.

Future cyber defenses against current and pending attacks require the integration of new or upgraded functional capabilities, the coordination of responses across domains, synchronization of response mechanisms, and deployment of automated actions in cyber relevant time.

Standardization of the languages, including lexicons, syntaxes, and encodings, used within the interfaces and protocols necessary for machine-to-machine command and control communications will enable cyber defense system flexibility, interoperability, and responsiveness in cyber-relevant time.

1.1 Purpose

The purpose of the Open Command and Control (OpenC2) Language Description Document is to define a lexicon (language and semantics) at a level of abstraction that will enable the coordination and execution of command and control of cyber defense components between and within networks. It is expected that the OpenC2 language will define profiles (i.e., applicable commands, applicable values) by community groups for specific cyber defense functions such as Software Defined Networking, Firewall, routing.

1.2 Scope

The scope of this document is to create a lexicon of actions and define the semantics, syntax and other aspects of a language that will couple an action with the target of the actions, and the entities that execute the actions. The document also defines an extensible syntax to accommodate attributes that further specify the targets, and modify the actions to support a wide range of operational environments.

Other aspects of OpenC2, such as implementation considerations, further refinement of the lexicon to accommodate specific cyber defense functions, encoding of commands for machine-to-machine communications, and reference implementations will be addressed in other artifacts. These other efforts will be consistent with this language description.

The definition of a language such as OpenC2 is necessary but insufficient to enable future cyber defenses. OpenC2 Commands can be carried within any number of constructs (e.g., STIX, workflows, playbooks, APIs). In addition, OpenC2 is designed to be flexible, agnostic of external protocols that provide services such as transport, authentication, key management and other services. Cyber defense implementations must consider and will require other protocols and security services.

1.3 Intended Audience

This OpenC2 Language Description Document is intended for organizations investigating the implementation of automated pre-approved cyber defensive measures as well as academia and industry partners involved with the development and integration of security orchestration, network components or services, endpoint security applications, and security services for cyber defenses.

1.4 Document Overview

Section 1, Introduction, describes the impetus for the OpenC2 language and lays out the purpose, scope, and intended audience of the document.

Section 2, Background, describes the design principles for the language and how the language can be contextualized for different operating environments.

Section 3, OpenC2 Language, describes the abstract syntax and the basic building blocks of the language. It also further specifies the vocabulary for actions, universal modifiers, action specific modifiers, and a default namespace for targets and target specifiers.

Section 4, Example OpenC2 Usage, provides examples of OpenC2 Command constructs. For each action, the supported targets, actuators, and action-specific modifiers are identified and example usages are provided.

Section 5, Example OpenC2 Use Case, depicts an example use case for mitigating an evil domain. The use case shows the OpenC2 Commands that could be used to mitigate the attacks or vulnerabilities and where they could be applied.

1.5 Document Conventions

The following typographical conventions are used in this document.

italics

Indicates new terms, URLs, email addresses, filenames, and file extensions.

ALL CAPS

Used for components of the abstract syntax: ACTION, TARGET, ACTUATOR, MODIFIERS.

bold blue

Used for action names.

`constant width`

Indicates new terms, URLs, email addresses, filenames, and file extensions.

2. Background

2.1 Design Principles

OpenC2 can be implemented in a variety of systems to perform the secure delivery and management of command and control messages in a context-specific way. OpenC2 Commands are vendor neutral and message fabric agnostic, thus can be incorporated in different architectures and environments (such as connection-oriented, connectionless, pub-sub, hub and spoke, etc.).

OpenC2 was designed to have a concise set of extensible commands in order to provide context-specific details. Conciseness ensures minimal overhead to meet possible latency and overhead constraints while extensions enable greater utility and flexibility.

There is an underlying assumption that issuing OpenC2 Commands is event-driven and that an action is warranted. OpenC2 was designed to focus on the actions that are to be executed in order to thwart an attack, mitigate some vulnerability, or otherwise address a threat. The exchange of indicators, rationale for the decision to act, and/or threat information sharing are beyond the scope of OpenC2 and left to other standards, such as STIX and TAXII.

The actual performance and efficacy of OpenC2 will be implementation-specific and will require the incorporation of other technologies. The OpenC2 design principles include the following:

- Support cyber-relevant response time for coordination and response actions.
- Be infrastructure, architecture, and vendor agnostic.
- Support multiple levels of abstraction necessary to permit the contextualization of commands for a wide variety of operating environments.
- Permit commands to be invoked that are either tasking/response actions or notifications.
    - Tasking/response actions result in a state change.
    - Notifications require supporting analytics/decision processes.
- Provide an extensible syntax to accommodate different types of actions, targets, and actuators (e.g., sensor, endpoint, network device, human) at varying levels of specificity.

- Ensure the OpenC2 language is independent of underlying message constructs that provide transport, identify priority/ quality of service, and support security attributes.

By design, OpenC2 is dependent upon but agnostic of the transport infrastructure and message fabric. Confidentiality, integrity, availability, and authentication must be identified and provisioned by the message fabric.

Traditional command and control implementations utilize complete, self-standing constructs. OpenC2 decouples the actions from the targets of the actions and from the recipients of the commands. An OpenC2 Command is not complete until an action is paired with a target, providing the command context for the action. This enables the OpenC2 language to be more concise, yet still support the entire C2 space. This characteristic of OpenC2 also permits a more flexible and extensible approach to accommodate future technologies and varying network environments.

2.2 OpenC2 and Deployment Environments

OpenC2 is defined at a level of abstraction such that an inter-domain tasking or coordination effort can be described without requiring in-depth knowledge of the recipient network's components, but, through the use of specifiers and modifiers, enough detail can be appended to carry out specific tasks on particular devices to support intra-domain command and control.

This level of abstraction permits end-to-end applicability of OpenC2. As depicted in Figure 2-1, an OpenC2 Command is sent to enable coordination or send a high level tasking from the peer or upper tier enclave. An OpenC2 Command received by an enclave will trigger events within the enclave to annotate the command with context-specific information so that specific devices within the enclave can respond appropriately. This allows the enclave to take advantage of this context-specific knowledge to interpret and appropriately execute OpenC2 Commands .

Each network contextualizes an OpenC2 action for the specific sensors and actuators within its environment so it can further specify the command to reflect the implementations of which it is capable. Context-specific modifiers provide an ability to further specify the action while enabling the set of actions to remain tightly constrained.

This minimizes the overhead, permits further contextualization of the OpenC2 Commands for specific environments, and thereby enables flexibility and extensibility.

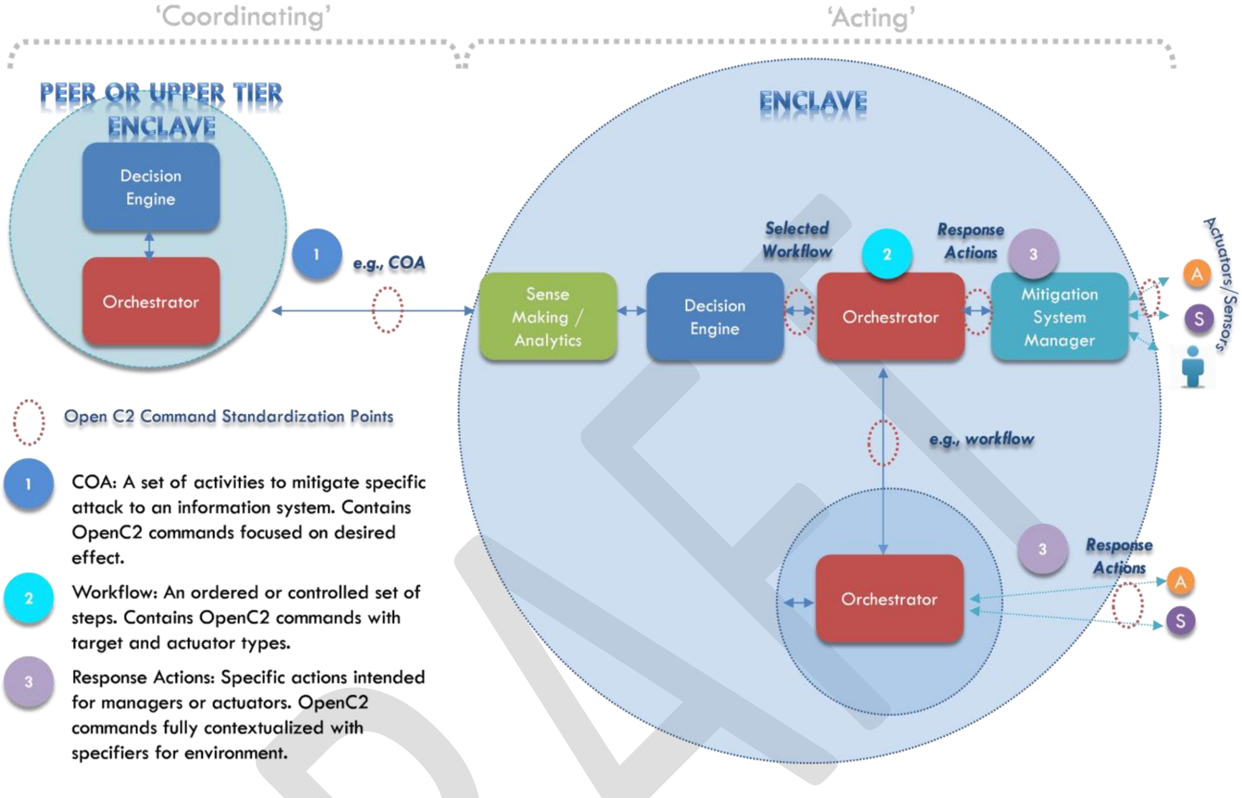
'Coordinating'
'Acting'
PEER OR UPPER TIER ENCLAVE
Decision Engine
Orchestrator
1
e.g., COA
ENCLAVE
Selected Workflow
2
Response Actions
3
Sense Making / Analytics
Decision Engine
Orchestrator
Mitigation System Manager
Actuators/Sensors
A
S
e.g., workflow
Orchestrator
3
Response Actions
A
S
Open C2 Command Standardization Points
1
COA: A set of activities to mitigate specific attack to an information system. Contains OpenC2 commands focused on desired effect.
2
Workflow: An ordered or controlled set of steps. Contains OpenC2 commands with target and actuator types.
3
Response Actions: Specific actions intended for managers or actuators. OpenC2 commands fully contextualized with specifiers for environment.

Figure 2-1. OpenC2 Deployment Environments

For example, an organization may have executed a series of actions to protect against a particular attack that was signaled by an external indicator (such as a STIX message). In order to elicit a consistent response across an organization (whether hierarchical or peer-to-peer), a complex course of action can be constructed and shared. The use of standardized OpenC2 Commands will be more precise and more quickly actionable than a set of recommended steps within a text document, which must be parsed, analyzed, and interpreted, prior to execution. Standardizing OpenC2 Commands helps to ensure a more uniform response at enterprises/enclaves that reflects enterprise-wide level decisions.

3. OpenC2 Language

3.1 Overview

The OpenC2 language is designed at a level of abstraction high enough such that it enables persistence as technologies advance and is implementation agnostic, but with enough precision that the need for specifiers and modifiers is limited.

The OpenC2 language has three distinct types of messages: *Command*, *Response*, and *Alert*. The OpenC2 Command describes an *action* performed on a *target*. It can be directive or descriptive depending on the context. The OpenC2 Response is used to provide data requested as a result of an action. The OpenC2 Response message will contain the requested data and have a reference to the action that initiated the response. The OpenC2 Alert is used to signal the occurrence of an event or error. It is an unsolicited message that does not need to reference a previously issued action.

3.2 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 Abstract Syntax

Conceptually, an OpenC2 Command has the following form:

```
(
      ACTION = <ACTION_TYPE>,
      TARGET (
            type = <data-model>:<TARGET_TYPE>,
            <target-specifier>
      ),
      ACTUATOR (
            type = <data-model>:<ACTUATOR_TYPE>,
            <actuator-specifier>
      ),
      MODIFIERS (
            <list-of-modifiers>
      )
```

```
        )
```

Fields denoted with angle brackets ("<>") are replaced with the appropriate details. Some of the fields are considered optional. The table below describes these fields semantically and whether they are required, optional or ignored in certain situations. Actual encoding will leverage pre-existing conventions and notations such as XML, JSON, TLV, or others.

The following table describes the fields that can be contained in an OpenC2 Command.

Table 3-1. OpenC2 Command Field Descriptions

Field	Description
ACTION	Required. The task or activity to be performed (i.e., the ‘verb’).
data-model	Required. The data model for the TARGET.
TARGET	Required. The object of the action. The ACTION is performed on the TARGET.
type	Required. The TARGET type will be defined within the context of a target data model.
target-specifier	Optional. The specifier further describes a specific target, a list of targets, or a class of targets.
ACTUATOR	Optional. The subject of the action. The ACTUATOR executes 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.
data-model	Required if the actuator is included, otherwise not applicable. The data model for the ACTUATOR.
actuator-specifier	Optional if the actuator is included, otherwise not applicable. The specifier further describes a specific actuator, a list of actuators, or a class of actuators.

Field	Description
MODIFIERS (<list-of-modifiers>)	Optional. Provide additional information about the action such as date/time, periodicity, duration, and location.

There are cases where an ACTION and TARGET are sufficient to complete the command, especially in the case of inter-domain commands where the method or approach to complete or execute the action can be determined within the receiving domain/enclave.

The majority of commands within an enclave will have an ACTION, TARGET, and ACTUATOR. Inclusion of the ACTUATOR provides additional context for the command as a whole and enables precision. .

Specifiers for TARGETs and ACTUATORs are optional and can be used to provide context specific information that could be used to reflect the local environment, policies, and operational conditions within an enterprise/enclave. Specifiers can call out a specific target/actuator, a list of targets/actuators, or a class of targets/actuators.

Modifiers to the ACTION are optional and are used to provide effects-based context to the ACTION. Modifiers are further discussed in Section 3.2.5.

Table 3-2 illustrates the use of specifiers and modifiers to extend the range of OpenC2 Commands to cover the higher level 'strategic' commands to the unambiguous enclave-specific use case. This provides greater flexibility to the language and allows the OpenC2 Actions to be further contextualized for the mission environment. The table below provides some examples of the different levels of specificity achievable in an OpenC2 Command.

Table 3-2. OpenC2 Syntax Flexibility Examples

Description	Action	Target	Actuator	Modifier
		Target-Specifier	**Actuator-Specifier**	
	deny	Network Connection		

Description	Action	Target	Actuator	Modifier
		Target-Specifier	Actuator-Specifier	
Block traffic to/from specific IP address(es) [effects-based, no actuator specified]; suitable for inter-domain coordination		Source and/or Destination IP Address(es)		
Block traffic at all network devices [specify actuator class]; suitable for inter-domain coordination or as a command to an orchestration engine which further contextualizes to the enclave's environment	deny	Network Connection	Network (any devices)	
		Source and/or Destination IP Address(es)		
Block traffic at network routers [specify type of network device actuator]; suitable within an enclave	deny	Network Connection	Network.router	
		Source and/or Destination IP Address	(optional)	
Block traffic at specific network router; [specify identity of network router]; suitable within an enclave	deny	Network Connection	Network.router	
		Source and/or Destination IP Address	Router identity	
	deny	Network Connection	Network.router	

Description	Action	Target	Actuator	Modifier
		Target-Specifier	Actuator-Specifier	
Block access to bad external IP by null routing; [specify method of performing action]; suitable within an enclave		Source and/or Destination IP Address	(optional)	Method= blackhole

3.2.1.1 Action

All OpenC2 Commands start with an ACTION, which indicates the type of command to perform such as gather and convey information, control activities and devices, and control permissions and access. The range of options and potential impact on the information system associated with a particular ACTION is a function of the ACTUATOR. For cases that involve multiple options for an ACTION, modifiers may be used.

Refer to Section 3.3 for the list of ACTIONs and their definitions and usage.

3.2.1.2 Target

All OpenC2 Commands include a TARGET. The TARGET is the object of the ACTION (or alternatively, the ACTION is performed on the TARGET). Targets include objects such as network connections, URLs, hashes, IP addresses, files, processes, fully qualified domain names etc.

3.2.1.3 Actuator

An ACTUATOR[1] is the entity that puts command and control into motion or action. The ACTUATOR is the subject of the ACTION which performs the ACTION on the TARGET. There

[1] Some academic circles model all cyber defense components as sensors and/or actuators. It is acknowledged that OpenC2 will be used for C2 of sensors as well, but in the interest of being concise within this document, actuators encompass sensors.

are varying levels of abstraction and functionality for an ACTUATOR ranging from a specific sensor to an entire system or even system of systems.

The source of a command may need to communicate an action that must be taken against a target, but will not necessarily have knowledge of the cyber defense technologies deployed in other enclaves so the inclusion of an actuator is optional within an OpenC2 Command. As a command is propagated through the system and context specific information is gained, the command can appended with an actuator and appropriate specifiers.

There will be only one ACTUATOR type per OpenC2 Command. The actuator namespace is specified in the OpenC2 profiles.

3.2.1.4 Specifiers

Specifiers are used to identify specific individual or groups of targets or actuators. Table 3-3 illustrates how the commands are appended with specifiers as context-specific details become available. The actuator specifiers presented in Table 3-3 are for illustrative purposes. The actual specifiers are defined in the appropriate actuator profiles.

Table 3-3. Example Usage of Specifiers

Description	Action	Target	Actuator	Modifier
		Target-Specifier	**Actuator-Specifier**	
Block malicious URL	deny	URI/URL		
		Value Condition = Equals		
Quarantine Artifact with particular byte string	quarantine	Artifact		
		Condition = Contains		
Block access to external IP address by	deny	Network Connection	Network router	

null routing at specific network routers		Condition = Contains	Manufacturer, Model, Serial Number Value = 123	

3.2.1.5 Modifiers

Modifiers provide additional precision about the action such as time, periodicity, duration, or other details on what is to be done. Modifiers can denote the when, where, and how aspects of an action. The modifier can also be used to convey the need for acknowledgement or additional status information about the execution of an action. Modifiers are similar to specifiers in that they can provide additional context-specific details, and are intended to provide additional details for action/actuator pairs. A modifier may be “actuator-specific”, “action-specific”, or “universal” depending on the applicability of the modifier within the language.

Actuator-specific modifiers are described in Actuator Profiles. Action-specific modifiers are described in Section 4. Universal modifiers are described in Section 3.2.5.

Table 3-4. Example Usage of Modifiers

Description	Action	Target	Actuator	Modifier
		Target-Specifier	**Actuator-Specifier**	
Shutdown a system, immediate	stop	Device	endpoint	method = immediate
		Device Object Type	(optional)	
Start Process with Delay	start	Process	endpoint	start_time
		Process Object Type	(optional)	
Quarantine a device	contain	Device	network	

Description	Action	Target	Actuator	Modifier
		Target-Specifier	Actuator-Specifier	
		Device Object Type	(optional)	where (network segment, vlan)
Block access to suspicious external IP address by redirecting external DNS queries to an internal DNS server	deny	Network Connection	DNS Server	method = sinkhole
		Network Connection Object Type		

3.2.2 Action Vocabulary

This section defines the set of OpenC2 actions grouped by their general activity. The following table summarizes the definition of the OpenC2 actions. Subsequent sections will identify the appropriate targets for each action and the appropriate actuators for the action-target pair. Further details will be defined in the actuator profiles.

- Actions that Control Information:
  These actions are used to gather information needed to determine the current state or enhance cyber situational awareness. These actions typically do not impact the state of the target and are normally not detectable by external observers.

- Actions that Control Permissions:
  These actions are used to control permissions and manage accesses.

- 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 (e.g., endpoint, sensor, actuator). The actions are used to execute tasks, adjust configurations, set and update parameters, and modify attributes.

- Sensor-Related Actions:
  These actions are used to control the activities of a sensor in terms of how to collect and provide the sensor data.

- 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 within an enclave. This level of abstraction enables coordinated actions between enclaves, while permitting a local enclave to optimize its workflow for its specific environment.

  Implementation of an effects-based action requires that the recipient enclave has a decision making capability because an effects-based action permits multiple possible responses.

Table 3-5. Summary of Action Definitions

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. This action enables one to tell where in the system an event or trigger occurred.
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 or 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.
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 of traffic to a particular destination other than its original intended destination.
delete	The **delete** action removes data and files.
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 deletes and/or replaces files, settings, or attributes to return the system to an identical or similar known state.
save	The **save** action commits data or system state to memory.
throttle	The **throttle** action adjusts the throughput of a data flow.
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 in the least detectable manner.
copy	The **copy** action duplicates a file or data flow.
sync	The **sync** action synchronizes a sensor or actuator with other system components.
Sensor-Related Actions	
distill	The **distill** action tasks the sensor to send a summary or abstraction of the sensing information instead of the raw data feed.

augment	The **augment** action tasks the sensor to do a level of preprocessing or sense making prior to sending the sensor data.
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.

3.2.3 Target Vocabulary

The TARGET is the object of the ACTION (or alternatively, the ACTION is performed on the TARGET). OpenC2 defines a default Target Data Model to support all of the actions. It is derived largely on the STIX Cyber Observables v2.x.

In addition to the default Target Data Model, the OpenC2 syntax can support any other data model. To differentiate alternative data models, a data model prefix is used to qualify the target type. The default target data model will prefix "openc2:" to the target type. The implementer will need to supply a unique data model prefix for non-standard target types. It is the responsibility of the implementer to ensure that there are no namespace collisions when using alternative data models. Refer to the following table for a summary of the OpenC2 Target Data Models.

Table 3-6. Target Data Model

Type	Description	Options

data-model	Used to uniquely identify a set of target types so there is no ambiguity; defines the context in which target types are defined.	Choice of: ● openc2 ● <external-ref>

Targets include objects such as network connections, URLs, hashes, IP addresses, files, processes, and domains. Refer to the following table for a summary of the supported OpenC2 Targets in the default Target Data Model.

Table 3-7. Summary of Supported Targets

Target Type	Description	Target Specifier
openc2:artifact	The Artifact Object permits capturing an array of bytes (8-bits), as a base64-encoded string or linking to a file-like payload.	mime_type : string, payload_bin : binary, url : string, hashes : hashes-type
openc2:command	The Command Object represents an OpenC2 Command.	command_id : command-ref
openc2:device	The Device Object represents the properties of a hardware device.	description: string, device_type: string, manufacturer: string, model : string, serial_number : string, firmware_version : string
openc2:directory	The Directory Object represents	path : string, path_enc : string,

Target Type	Description	Target Specifier
	the properties common to a file system directory.	created : timestamp, modified : timestamp, accessed : timestamp, contains_refs : list of type object-ref
openc2:disk	The Disk Object represents a disk drive.	disk_name : string, disk_size : integer, free_space : integer, partition_list : list of type disk-partition type : string
openc2:disk-partition	The Disk Partition Object represents a single partition of a disk drive.	created : timestamp, device_name : string, mount_point : string, partition_id : string, partition_length : integer, partition_offset : integer, space_left : integer, space_used : integer, total_space : integer, type : string
openc2:domain-name	The Domain Name represents the properties of a network domain name.	value : string, resolves_to_refs : list of type object-ref
openc2:email-addr	The Email Address Object represents a single email address.	value : string, display_name : string, belongs_to_ref : object-ref
openc2:email-message	The Email Message Object represents an instance of an email message,	is_multipart : boolean, date : timestamp, content_type : string, from_ref : object-ref,

Target Type	Description	Target Specifier
	corresponding to the internet message format described in RFC 5322 and related RFCs.	sender_ref : object-ref, to_refs : list of type object-ref, cc_refs : list of type object-ref, bcc_refs : list of type object-ref, subject : string, received_lines : list of type string, additional_header_fields : dictionary, body : string, body_multipart : list of type mime-part-type, raw_email_ref : object-ref
openc2:file	The File Object represents the properties of a file.	extensions : dictionary, hashes : hashes-type, size : integer, name : string, name_enc : string, magic_number_hex : hex, mime_type : string, created : timestamp, modified : timestamp, accessed : timestamp, parent_directory_ref : object-ref, is_encrypted : boolean, encryption_algorithm : open-vocab, decryption_key : string, contains_refs : list of type object-ref, content_ref: object-ref
openc2:ipv4-addr	The IPv4 Address Object represents one or more IPv4 addresses expressed using CIDR notation.	value : string, resolves_to_refs : list of type object-ref, belongs_to_refs : list of type object-ref

Target Type	Description	Target Specifier
openc2:ipv6-addr	The IPv6 Address Object represents one or more IPv6 addresses expressed using CIDR notation.	value : string, resolves_to_refs : list of type object-ref, belongs_to_refs : list of type object-ref
openc2:mac-addr	The MAC Address Object represents a single Media Access Control (MAC) address.	value : string
openc2:memory	The Memory Object represents memory objects.	hashes : list of type string, name : string, memory_source : string, region_size : integer, block_type : string, region_start_address : string, region_end_address : string, extracted_features : string
openc2:network-traffic	The Network Traffic Object represents arbitrary network traffic that originates from a source and is addressed to a destination.	extensions : dictionary, start : timestamp, end : timestamp, is_active : boolean, src_ref : object-ref, dst_ref : object-ref, src_port : integer, dst_port : integer, protocols : list of type string, src_byte_count : integer, dst_byte_count : integer, src_packets : integer, dst_packets : integer, ipfix : dictionary,

Target Type	Description	Target Specifier
		src_payload_ref : object-ref, dst_payload_ref : object-ref, encapsulates_refs : list of type object-ref, encapsulated_by_ref : object-ref
openc2:openc2	The OpenC2 Object is a subset of the Artifact Object that represents an Actuator's OpenC2 supported capabilities.	value : string, attributes : list of type string, search : string
openc2:process	The Process Object represents common properties of an instance of a computer program as executed on an operating system.	extensions : dictionary, is_hidden : boolean, pid : integer, name : string, created : timestamp, cwd : string, arguments : list of type string, encironment_variables : dictionary, opened_connection_refs : list of type object-ref, creator_user_ref : object-ref, binary_ref : object-ref, parent_ref : object-ref, child_refs : list of type object-ref
openc2:software	The Software Object represents high-level properties associated with software, including	name : string, cpe : string, language : string, vendor : string, version : string

Target Type	Description	Target Specifier
	software products.	
openc2:url	The URL Object represents the properties of a uniform resource locator (URL).	value : string
openc2: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.	extensions : dictionary, user_id : string, account_login : string, account_type : open-vocab, display_name : string, is_service_account : boolean, is_privileged : boolean, can_escalate_privs : boolean, is_disabled : boolean, account_created : timestamp, account_expires : timestamp, password_last_changed : timestamp, account_first_login : timestamp, account_last_login : timestamp
openc2:user-session	The User Session Object represents a user session.	effective_group : string, effective_group_id : string, effective_user : string, effective_user_id : string, login_time : timestamp, logout_time : timestamp
openc2:volume	The Volume Object represents a generic drive volume.	name : string, device_path : string, file_system_type : string, total_allocation_units : integer, sectors_per_allocation_unit : integer, bytes_per_sector : integer,

Target Type	Description	Target Specifier
		actual_available_allocation_units : integer, creation_time : timestamp, file_system_flag_list : list of type string, serial_number : string
openc2:windows-registry-key	The Registry Key Object represents the properties of a Windows registry key.	key : string, values : list of type windows-registry-value-type, modified : timestamp, creator_user_ref : object-ref, number_of_subkeys : integer
openc2:x509-certificate	The X509 Certificate Object represents the properties of an X.509 certificate, as defined by ITU recommendation X.509.	is_self_signed : boolean, hashes : hashes-type, version : string, serial_number : string, signature_algorithm : string, issuer : string, validity_not_before : timestamp, validity_not_after : timestamp, subject : string, subject_public_key_algorithm : string, subject_public_key_modulus : string, subject_public_key_exponent : integer, x509_v3_extensions : x509-v3-extensions-type

3.2.4 Actuator Vocabulary

An ACTUATOR is the entity that puts command and control into motion or action. The ACTUATOR executes the ACTION on the TARGET. The Actuator Data Model is defined in one or more *actuator profiles* where an actuator profile is a document that defines actions that are mandatory to implement, optional actions, and the appropriate actuator specifiers and the actuator-specific modifiers. The data model identifies which actuator profile is being referenced. The actuator profiles referenced in this document are for illustrative purposes.

In addition to the default Actuator Data Model, the OpenC2 syntax can support any other data model. To differentiate alternative data models, a data model prefix is used to qualify the actuator type. The default actuator data model will prefix "openc2:" to the actuator type. The implementer will need to supply a unique data model prefix for non-standard actuator types. It is the responsibility of the implementer to ensure that there are no namespace collisions when using alternative data models. Refer to the following table for a summary of the OpenC2 Actuator Data Models.

Table 3-8. Actuator Data Model

Type	Description	Options
data-model	Used to uniquely identify a set of actuator types so there is no ambiguity; defines the context in which target types are defined.	Choice of: ● openc2 ● <external-ref> ●

Table 3-9. List of Functional Actuators

Actuator Type	Description
endpoint	Endpoint Device
endpoint-workstation	
endpoint-server	
network	Network Platform
network-firewall	

Actuator Type	Description
network-router	
network-proxy	
network-sensor	
network-hips	
network-sense-making	
process	Services/Processes
process-anti-virus-scanner	
process-aaa-service	
process-virtualization-service	
process-sandbox	
process-email-service	
process-directory-service	
process-remediation-service	
process-location-service	

3.2.5 Modifier Vocabulary

Modifiers provide additional information about the action such as time, periodicity, duration, and location. Modifiers can denote the when, where, and how aspects of an action. The modifier can also be used to convey the need for additional status information about the execution of an action such as a response is required. The requested status/information will be carried in a RESPONSE. Refer to Section 4.6.

Modifiers are similar to specifiers in that they can provide additional context specific-details for an action. Modifiers that are applicable to any action are referred to as 'universal modifiers' and are presented in table 3-10. Modifiers that are applicable to a particular action , regardless of the actuator are referred to as 'Action-specific' and are are identified in the sections detailing each action. Modifiers that are only applicable to an action for a particular actuator are referred to as 'Actuator-Specific' and are defined within the actuator profiles.

The following table lists the set of universal modifiers that are applicable to all types of actions.

Table 3-10. Summary of Universal Modifiers

Modifier	Type	Description	Target Applicability
context	string	A reference that provides context for the action.	All
start_time	date-time (RFC 3339)	The specific date/time to initiate the action.	All
end_time	date-time (RFC 3339)	The specific date/time to terminate the action.	All
command_id	command-id	The unique identifier for the action.	All
response	ack, status	Indicate the type of response required for the action.	All
respond_to	string	The location where the response should be sent.	All

3.3 OpenC2 Response

OpenC2 Response is used to provide any data requested as a result of an action. It can be used to signal the acknowledgement of an action, provide the status of an action along with additional information related to the requested action, or signal the completion of the action. The recipient of the OpenC2 Response can be the original requester of the action or to another recipient(s) designated in the modifier of the action.

3.3.1 Abstract Syntax

Conceptually, an OpenC2 Response has the following form:

```
(
    RESPONSE (
        SOURCE (
            type = <data-model>:<ACTUATOR_TYPE>,
            <actuator-specifier>
        ),
    [CMDREF = <COMMAND_REFERENCE>,]
    STATUS = <STATUS_CODE>,
    [STATUS_TEXT = <STATUS_TEXT>,]
    RESULTS (
        <DEFINED_VALUE>
    )
)
```

Fields denoted with angle brackets ("<>") are replaced with the appropriate details. Some of the fields are considered optional. The table below describes these fields semantically and whether they are required, optional or ignored in certain situations. Actual encoding will leverage pre-existing conventions and notations such as XML, JSON, TLV or others.

The following table contains the description of the fields that can be contained in an OpenC2 Response.

Table 3-11. OpenC2 Response Field Descriptions

Field	Description
SOURCE	The origin of the Response.

Field	Description
type	The SOURCE type will be defined within the context of a data model.
data-model	The data model for the SOURCE.
actuator-specifier	The specifier describes a specific actuator, a list of actuators, or a class of actuators that is the source of the Response.
CMDREF	Optional. Refers to the command_id modifier of the OpenC2 Command that the Response is associated with.
STATUS	A status code.
STATUS_TEXT	Optional. Any descriptive text associated with the status code.
RESULTS	Optional. Contains the data that was requested from an OpenC2 Command. If not present, the STATUS field is a sufficient response.

3.3.2 Source Vocabulary

TBD

3.3.3 Command Reference

TBD

3.3.4 Status

TBD

3.3.5 Results

TBD

3.4 OpenC2 Alert

The OpenC2 Alert is used to signal the occurrence of an event or error. It is an unsolicited message that does not need to reference a previously issued command.

3.4.1 Abstract Syntax

Conceptually, an OpenC2 Alert has the following form:

```
    (
        ALERT (
            SOURCE (
                type = <data-model>:<ACTUATOR_TYPE>,
                <actuator-specifier>
            ),
        [CMDREF = <COMMAND_REFERENCE>,]
        STATUS = <STATUS_CODE>,
        [STATUS_TEXT = <STATUS_TEXT>,]
        RESULTS (
            <DEFINED_VALUE>
        )
    )
```

Fields denoted with angle brackets ("<>") are replaced with the appropriate details. Some of the fields are considered optional. The table below describes these fields semantically and whether they are required, optional or ignored in certain situations. Actual encoding will leverage pre-existing conventions and notations such as XML, JSON, TLV or others.

The following table contains the description of the fields that can be contained in an OpenC2 Alert.

Table 3-12. OpenC2 Alert Field Descriptions

Field	Description
SOURCE	The origin of the Alert.
type	The SOURCE type will be defined within the context of a data model.

Field	Description
data-model	The data model for the SOURCE.
actuator-specifier	The specifier describes a specific actuator, a list of actuators, or a class of actuators that is the source of the Alert.
CMDREF	Optional. Refers to an OpenC2 Command that the Response is associated with.
STATUS	A status code.
STATUS_TEXT	Optional. Any descriptive text associated with the status code.
RESULTS	Optional. Contains the data relevant to the Alert.

3.4.2 Source Vocabulary

TBD

3.4.3 Command Reference

TBD

3.4.4 Status

TBD

3.4.5 Results

TBD

4. EXAMPLE OpenC2 Command USAGE

This section provides examples of OpenC2 Commands that correspond to each OpenC2 action and its applicable targets. This section also defines any action specific modifiers. The purpose of this section is to provide sample commands that are consistent with the syntax defined in this document and to illustrate the flexibility of the OpenC2 language.

4.1 Actions that Control Information

These actions are used to gather information needed to further determine courses of action or assess the effectiveness of courses of action. These actions can be used to support data enrichment use cases and maintain situational awareness. These actions typically do not impact the state of the target and are normally not detectable by external observers.

4.1.1 **scan**

The **scan** action is the systematic examination of some aspect of the entity or its environment in order to obtain information.

The **scan** action can be used to command the characterization of an environment (e.g., perform network, port, or vulnerability scanning) or to look for a specific occurrence of an object (e.g., file, IP, process). The **scan** action is distinct from **query** in that **scan** implies an analytic while a **query** implies a routine retrieval of data.

Table. Supported Targets and Actuators: scan

Target Type	Actuator Type
openc2:device openc2:disk openc2:disk-partition openc2:domain-name openc2:email-message openc2:file openc2:ipv4-addr openc2:memory openc2:network-traffic openc2:process openc2:software openc2:url openc2:user-account openc2:user-session openc2:volume	network-sensor

Table. Modifiers: scan

Modifier	Type	Description	Target Applicability
method	enumeration: non-authenticated, authenticated	Optional. When there is more than one way to perform the action, the method can be specified, if necessary.	All
search	cve, patch, vendor bulletin, signature	Required. The search criteria for performing the scan.	All

Below is a sample usage table for the **scan** action, depicting actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table. Sample of OpenC2 Commands: scan

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Scan a device for vulnerabilities	scan	openc2:device	network-sensor	search = CVE
			(as required)	(optional)	
2	Scan email messages for malware	scan	openc2:email-message	network-sensor	search = malware signature

	Description	Action	Target Target-Specifier	Actuator Actuator-Specifier	Modifier
			(as required)	(optional)	
3	Scan network traffic for malicious activities	scan	openc2:network-traffic (as required)	network-sensor (optional)	search = network signature

4.1.2 **locate**

The **locate** action is used to find an object either physically, logically, functionally, or by organization. This action enables one to tell where in the system an event or trigger occurred.

The **locate** action is used for example to enable one to tell where in the system an event or trigger occurred, confirm that an asset is appropriately deployed, or ascertain details regarding a rogue device.

Table. Supported Targets and Actuators: locate

Target Type	Actuator Type
openc2:device openc2:file openc2:ipv4-addr openc2:user-account	process-location-service

Table. Modifiers: locate

Modifier	Type	Description	Target Applicability
None to Date			

Below is a sample usage table for the **locate** action, depicting actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table. Sample of OpenC2 Commands: locate

	Description	Action	Target Target-Specifier	Actuator Actuator-Specifier	Modifier
1	Geolocate a device	locate	openc2:device (as required)	process-location-service (optional)	
2	Get location of an IP address	locate	openc2:ipv4-addr (as required)	process-location-service (optional)	

4.1.3 **query**

The **query** action initiates a single request for information.

The **query** action, like **scan**, is used to find out more information about the system or its environment. In the case of **query**, however, it is an isolated or specific information request, rather than a broadly scoped scan or on-going check. The **query** action is used to retrieve data that is already present in a database or data store, while **scan** implies a more thorough examination and identification of anomalies (relative to a known good state). The response to a **query** is typically (but not necessarily) conveyed within the command and control channel.

The target for **query** is usually `openc2:artifact`. The target-specifier describes the search criteria for the information request.

A special target for **query** is `openc2:openc2` which signifies a request for an actuator's OpenC2 capabilities (i.e., a list of supported actions, targets). If not target-specifier is included in the request then the full report of the actuator's capabilities should be provided. A response could be filtered for a particular capability by providing details in the target-specifier.

Table. Supported Targets and Actuators: query

Target Type	Actuator Type
openc2:artifact openc2:openc2	endpoint network-firewall network-router process-directory-service

Table. Modifiers: query

Modifier	Type	Description	Target Applicability
None to Date			

Below is a sample usage table for the **query** action, depicting actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table. Sample of OpenC2 Commands: query

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	**Actuator-Specifier**	
1	List all network connections	query	openc2:artifact	network-router	response
			(as required)	(optional)	
2	List running processes on a machine	query	openc2:artifact	endpoint	response
			(as required)	(optional)	
3	Request an Actuator's supported OpenC2 capabilities	query	openc2:openc2	network-firewall	response
			(as required)	(optional)	

4.1.4 **report**

The **report** action tasks an entity to provide information to a designated recipient of the information.

The **report** action is used to request an actuator to provide certain information. Along with the **report** action and the type of information being requested, the recipient of the information must be specified in the command. The response to a **report** action is typically (but not necessarily) conveyed outside of the command and control channel.

Table. Supported Targets and Actuators: report

Target Type	Actuator Type
openc2:artifact	

Table. Modifiers: report

Modifier	Type	Description	Target Applicability
frequency	duration (RFC 3339)	Optional. The frequency at which to perform the action. The value is the requested time between execution events.	All
report_to	openc2:ipv4-addr, openc2:ipv6-addr	Required. This modifier identifies where to send the report.	All

Below is a sample usage table for the **report** action, depicting actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table. Sample of OpenC2 Commands: report

	Description	Action	Target Target-Specifier	Actuator Actuator-Specifier	Modifier
1	Produce and send a report	report	openc2:artifact (as required)		report_to

4.1.5 **notify**

The **notify** action is used to direct an entity to send information to another entity.

The **notify** action is distinct from **report** in that **notify** is used for time sensitive event notification and carries a sense of persistence.

Table. Supported Targets and Actuators: notify

Target Type
openc2:process openc2:user-account

Actuator Type
endpoint-server process-email-service

Table. Modifiers: notify

Modifier	Type	Description	Target Applicability
frequency	duration (RFC 3339)	Optional. The frequency at which to perform the action. The value is the requested time between execution events.	All
message		The intended message to notify the target.	All

Below is a sample usage table for the **notify** action, depicting actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table. Sample of OpenC2 Commands: notify

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	**Actuator-Specifier**	
1	Notify security officer to report compliance with change of configuration	notify	openc2:user-account	process-email-service	message
			(as required)	(optional)	
2	Send a command to notify an external enclave	notify	openc2:process		message = acknowledge
			(as required)		

4.2 Actions that Control Permissions

These actions are used to control permissions and accesses.

4.2.1 **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.

The **deny** action is a superset of current terms such as *block* (network perimeter devices) and *deny* (user, access to system, access to files).

Table. Supported Targets and Actuators: deny

Target Type
openc2:device
openc2:network-traffic
openc2:process
openc2:software
openc2:url
openc2:user-account

Actuator Type
endpoint
network-firewall
network-proxy
network-router
process
process-aaa-service

Table. Modifiers: deny

Modifier	Type	Description	Target Applicability
method	enumeration: acl, blackhole, sinkhole, blacklist, whitelist	Optional. When there is more than one way to perform the action, the method can be specified, if necessary.	openc2:network-traffic, openc2:product

Modifier	Type	Description	Target Applicability
where	enumeration: internal, perimeter	Optional. The general location within the enclave to perform the DENY action.	openc2:network-traffic

Below is a sample usage table for the **deny** action, depicting actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table. Sample of OpenC2 Commands: deny

	Description	Action	Target Target-Specifier	Actuator Actuator-Specifier	Modifier
1	Block traffic to/from specific IP address; suitable for coordinating across multiple enclaves and allowing enclaves to determine most appropriate response	deny	openc2:network-traffic (as required)		
2	Block traffic to/from specific IP address at all network firewalls	deny	openc2:network-traffic (as required)	network-firewall (optional)	
3	Block traffic at the network routers	deny	openc2:network-traffic (as required)	network-router (optional)	

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
4	Block network traffic inside the enclave	deny	openc2:network-traffic (as required)		where = internal
5	Block network traffic at the perimeter	deny	openc2:network-traffic (as required)		where = perimeter
6	Block network traffic by ACL	deny	openc2:network-traffic (as required)	network-router (optional)	method = acl
7	Block access to a bad external IP address by null routing at the network routers.	deny	openc2:network-traffic (as required)	network-router (optional)	method = blackhole

4.2.2 **contain**

The **contain** action stipulates the isolation of a file or process or entity such that it cannot modify or access assets or processes that support the business and/or operations of the enclave.

The **contain** action is a superset of currently used terms such as *isolate*, *quarantine*, or *sandbox*.

Table. Supported Targets and Actuators: contain

Target Type
openc2:device
openc2:file
openc2:network-traffic
openc2:process
openc2:user-account

Actuator Type
endpoint
network

Table. Modifiers: contain

Modifier	Type	Description	Target Applicability
where		Optional. The general location within the enclave to contain the target.	openc2:device, openc2:file, openc2:network-traffic, openc2:process, openc2:user-account

Below is a sample usage table for the **contain** action, depicting actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table. Sample of OpenC2 Commands: contain

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Quarantine a file, general	contain	openc2:file		
			(as required)		
2	Quarantine a file	contain	openc2:file	endpoint	where
			(as required)	(optional)	
3	Contain a user or group, general	contain	openc2:user-account		
			(as required)		
4	Contain network traffic to a honeynet, general	contain	openc2:network-traffic		
			(as required)		

4.2.3 **allow**

The **allow** action permits the access to or execution of a target.

An **allow** action is typically associated with something that was previously denied (e.g., **deny**, **contain**).

Table. Supported Targets and Actuators: allow

Target Type	Actuator Type
openc2:device openc2:file openc2:network-traffic openc2:process openc2:software openc2:url openc2:user-account	endpoint network network-firewall network-proxy network-router process process-aaa-service

Table. Modifiers: allow

Modifier	Type	Description	Target Applicability
permissions		Optional. Specific permissions to be granted to the user.	openc2:user-account
where	enumeration: internal, perimeter	Optional. The general location within the enclave to perform the ALLOW	openc2:network-traffic

Modifier	Type	Description	Target Applicability
		action.	

Below is a sample usage table for the **allow** action, depicting actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table. Sample of OpenC2 Commands: allow

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Unblock traffic to/from specific IP address; suitable for coordinating across multiple enclaves and allowing enclaves to determine most appropriate response	allow	openc2:network-traffic		
			(as required)		
2	Unblock traffic to/from specific IP address at all network firewalls	allow	openc2:network-traffic	network-firewall	
			(as required)	(optional)	
3	Unblock traffic at the network routers	allow	openc2:network-traffic	network-router	
			(as required)	(optional)	

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
4	Unblock network traffic inside the enclave	allow	openc2:network-traffic		where = internal
			(as required)		
5	Delay Machine Authentication	allow	openc2:device	process-aaa-service	start_time
			(as required)	(optional)	
6	Unquarantine a file	allow	openc2:file	endpoint	
			(as required)	(optional)	

4.3 Actions that Control Activities/Devices

These actions are used to execute some task, adjust configurations, set and update parameters etc. These actions typically change the state of the system.

4.3.1 **start**

The **start** action initiates a process, application, system or some other activity.

Table. Supported Targets and Actuators: start

Target Type
openc2:device openc2:disk-partition openc2:process openc2:software

Actuator Type
endpoint network process-virtualization-service

Table. Modifiers: start

Modifier	Type	Description	Target Applicability
method	enumeration: spawn		openc2:process

Below is a sample usage table for the **start** action, depicting actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table. Sample of OpenC2 Commands: start

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Start Process, general	start	openc2:process		
			(as required)		
2	Start Process	start	openc2:process	endpoint	
			(as required)	(optional)	
3	Start Process with Delay	start	openc2:process	endpoint	start_time
			(as required)	(optional)	
4	Spawn Process	start	openc2:process	endpoint	method = spawn
			(as required)	(optional)	

4.3.2 **stop**

The **stop** action halts a system or ends an activity.

The **stop** action is used to convey terms in current use such as shutdown, kill, and terminate. The **stop** action has nuances and options associated with it that are actuator specific. In the case where more than one type of **stop** action is applicable for a particular target and actuator, if practical, the default implementation of **stop** should be a graceful shutdown. Action modifiers are used to indicate immediate or atypical **stop** actions.

Table. Supported Targets and Actuators: stop

Target Type
openc2:device
openc2:disk-partition
openc2:process
openc2:user-account
openc2:user-session

Actuator Type
endpoint
network
process-aaa-service
process-virtualization-service

Table. Modifiers: stop

Modifier	Type	Description	Target Applicability
method	enumeration: graceful, immediate	Optional. When there is more than one way to perform the action, the method can be specified, if necessary.	All

Below is a sample usage table for the **stop** action, depicting actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table. Sample of OpenC2 Commands: stop

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Shutdown a system	stop	openc2:device (as required)	endpoint (optional)	[method = graceful]
2	Shutdown a system, immediate	stop	openc2:device (as required)	endpoint (optional)	method = immediate
3	Logoff User: Logoff all the sessions of a particular user from the machine	stop	openc2:user-account (as required)	endpoint (optional)	[method = graceful]
4	Stop a vm	stop	openc2:process (as required)	process-virtualization-service (optional)	[method = graceful]

4.3.3 **restart**

The **restart** action conducts a **stop** of a system or an activity followed by a **start** of a system or an activity.

A **restart** implies a graceful shutdown, maintenance of state, and a new configuration.

Table. Supported Targets and Actuators: restart

Target Type	Actuator Type
openc2:device	endpoint
openc2:process	process-virtualization-service

Table. Modifiers: restart

Modifier	Type	Description	Target Applicability
frequency	duration (RFC 3339)	Optional. The frequency at which to perform the action. The value is the requested time between execution events.	All
options		Additional options that specify how to restart	All

Below is a sample usage table for the **restart** action, depicting actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table. Sample of OpenC2 Commands: restart

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Restart device (system)	restart	openc2:device		
			(as required)		
2	Restart device (system) with different OS	restart	openc2:device		options, e.g., OS
			(as required)		
3	Restart VM	restart	openc2:process	process-virtualization-service	
			(as required)	(optional)	

4.3.4 **pause**

The **pause** action ceases a system or activity while maintaining state.

A **pause** remains in effect until a **resume** is issued, unless the **pause** action is accompanied by modifier for a time-interval.

Table. Supported Targets and Actuators: pause

Target Type	Actuator Type
openc2:device openc2:process	endpoint process-virtualization-service

Table. Modifiers: pause

Modifier	Type	Description	Target Applicability
duration	duration (RFC 3339)	Optional. The time to wait until returning to the previous state.	All
method	enumeration: sleep, hibernate, suspend	Optional. When there is more than one way to perform the action, the method can be specified, if necessary.	All

Below is a sample usage table for the **pause** action, depicting actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table. Sample of OpenC2 Commands: pause

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Pause device (system)	pause	openc2:device		[method = sleep]
			(as required)		
2	Hibernate device (system)	pause	openc2:device		method = hibernate
			(as required)		
3	Pause VM	pause	openc2:process	process-virtualization-service	
			(as required)	(optional)	
4	Pause a system or VM for a specified duration	pause	openc2:process		duration = <DURATION>
			(as required)		

4.3.5 **resume**

The **resume** action starts a system or activity from a paused state.

The **resume** action is only meaningful after a **pause** action was issued.

Table. Supported Targets and Actuators: resume

Target Type	Actuator Type
openc2:device openc2:process	endpoint process-virtualization-service

Table. Modifiers: resume

Modifier	Type	Description	Target Applicability
None to Date			

Below is a sample usage table for the **resume** action, depicting actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table. Sample of OpenC2 Commands: resume

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Resume device (system)	resume	openc2:device		
			(as required)		
2	Resume VM	resume	openc2:process	process-virtualization-service	
			(as required)	(optional)	

4.3.6 **cancel**

The **cancel** action invalidates a previously issued action.

The **cancel** action must be associated with a previously issued command through the `command_ref` modifier. This action is intended to stop an action that has not initiated or completed and is not intended to undo a completed action and return to a previous state. It can set the validity period to immediately end or it could define a future duration for which the action is valid.

Table. Supported Targets and Actuators: cancel

Target Type	Actuator Type
openc2:command	endpoint network process

Table. Modifiers: cancel

Modifier	Type	Description	Target Applicability
command_ref	command-id	The reference to the associated command that is to be cancelled.	openc2:Command
duration	duration (RFC 3339)	Optional. The period of time that an action is valid. If not present, the CANCEL operation should occur immediately.	openc2:Command

Below is a sample usage table for the **cancel** action, depicting actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table. Sample of OpenC2 Commands: cancel

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Cancel a previously issued command	cancel	openc2:command		command_ref = <CMD_ID>
			(as required)		
2	Cancel a previously issued command, directed to a specific actuator (endpoint)	cancel	openc2:command	endpoint	command_ref = <CMD_ID>
			(as required)	(optional)	

4.3.7 **set**

The **set** action changes a value, configuration, or state of a managed entity within an IT system.

Typically, the **set** action is specified by a configuration item such as a sensor setting or privilege level and the command will have specifiers. The **set** action is intended for specific individual changes to the entity and the parameters are communicated in the command and control channel.

Table. Supported Targets and Actuators: set

Target Type
openc2:artifact
openc2:file
openc2:process
openc2:user-account
openc2:windows-registry-key

Actuator Type
endpoint-workstation
network-firewall
network-hips
network-router
network-sensor
process-directory-service

Table. Modifiers: set

Modifier	Type	Description	Target Applicability
set_to		The value to set the target to.	All

Below is a sample usage table for the **set** action, depicting actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table. Sample of OpenC2 Commands: set

	Description	Action	Target Target-Specifier	Actuator Actuator-Specifier	Modifier
1	Set registry key value	set	openc2:windows-registry-key (as required)	endpoint-workstation (optional)	set_to
2	Set file permissions	set	openc2:file (as required)	process-directory-service (optional)	set_to
3	Set user rights	set	openc2:user-account (as required)	process-directory-service (optional)	set_to

4.3.8 **update**

The **update** action instructs the component to retrieve, install, process, and operate in accordance with a software update, reconfiguration, or some other update.

The settings, files, patches associated with an **update** action are typically retrieved out of band from the control channel. It is incumbent upon the OpenC2 compliant devices to include implementation details such as save, reboot, restart.

Table. Supported Targets and Actuators: update

Target Type
openc2:artifact openc2:file openc2:software openc2:windows-registry-key

Actuator Type
endpoint network-sensor process-anti-virus-scanner

Table. Modifiers: update

Modifier	Type	Description	Target Applicability
frequency	duration (RFC 3339)	Optional. The frequency at which to perform the action. The value is the requested time between execution events.	All
source		The source of the updated information.	All

Below is a sample usage table for the **update** action, depicting actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table. Sample of OpenC2 Commands: update

	Description	Action	Target Target-Specifier	Actuator Actuator-Specifier	Modifier
1	Install software	update	openc2:software (as required)	endpoint (optional)	
2	Install patch	update	openc2:software (as required)	endpoint (optional)	
3	Update signature file (anti-virus)	update	openc2:artifact (as required)	process-anti-virus-scanner (optional)	

4.3.9 **move**

The **move** action changes the location of a file, subnet, network, or, process.

The **move** action is distinct from **contain** in that **contain** implies a desired effect of isolation and **move** supports the more general case.

Table. Supported Targets and Actuators: move

Target Type	Actuator Type
openc2:artifact openc2:file	

Table. Modifiers: move

Modifier	Type	Description	Target Applicability
move_to	location	The location to move to	All

Below is a sample usage table for the **move** action, depicting actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table. Sample of OpenC2 Commands: move

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Move file/directory	move	openc2:file		move_to
			(as required)		

4.3.10 **redirect**

The **redirect** action changes the flow of traffic to a particular destination other than its original intended destination.

The **redirect** action includes the case of bypassing an intermediate point. The **redirect** action is distinct from **move** in that it encompasses the entire flow rather than a single instance, item or object. The **move** action supports the more atomic case.

Table. Supported Targets and Actuators: redirect

Target Type	Actuator Type
openc2:network-traffic openc2:url	network-router

Table. Modifiers: redirect

Modifier	Type	Description	Target Applicability
where		Optional. The location within the enclave to redirect the target. "where = null" will cancel previous redirection actions.	All

Below is a sample usage table for the **redirect** action, depicting actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table. Sample of OpenC2 Commands: redirect

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Redirect traffic to a honeypot; suitable for coordinating across multiple enclaves and allowing enclaves to determine most appropriate response	redirect	openc2:network-traffic		where
			(as required)		
2	Redirect traffic to a honeypot at a specific router	redirect	openc2:network-traffic	network-router	where
			(as required)		
3	Cancel traffic redirection; suitable for coordinating across multiple enclaves and allowing enclaves to determine most appropriate response	redirect	openc2:network-traffic		where = null
			(as required)		

4.3.11 **delete**

The **delete** action removes data and files.

Table. Supported Targets and Actuators: delete

Target Type
openc2:artifact
openc2:email-message
openc2:file

Actuator Type
endpoint
network-firewall
process-email-service

Table. Modifiers: delete

Modifier	Type	Description	Target Applicability
None to Date			

Below is a sample usage table for the **delete** action, depicting actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table. Sample of OpenC2 Commands: delete

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Delete file, inter-enclave	delete	openc2:file		
			(as required)		
2	Delete file, within an enclave	delete	openc2:file	endpoint	
			(as required)	(optional)	
3	Delete email, inter-enclave	delete	openc2:email-message		
			(as required)		
4	Delete email from exchange server	delete	openc2:email-message	process-email-service	
			(as required)	(optional)	

4.3.12 **snapshot**

The **snapshot** action records and stores the state of a target at an instant in time.

Table. Supported Targets and Actuators: snapshot

Target Type	Actuator Type
openc2:process	process-virtualization-service

Table. Modifiers: snapshot

Modifier	Type	Description	Target Applicability
None to Date			

Below is a sample usage table for the **snapshot** action, depicting actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table. Sample of OpenC2 Commands: snapshot

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Take a snapshot of a VM	snapshot	openc2:process (as required)	process-virtualization-service (optional)	

4.3.13 **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.

The **detonate** action is distinct from **contain** in that **detonate** includes an execution and analytic component rather than just isolation.

Table. Supported Targets and Actuators: detonate

Target Type
openc2:file openc2:url

Actuator Type
process-sandbox

Table. Modifiers: detonate

Modifier	Type	Description	Target Applicability
None to Date			

Below is a sample usage table for the **detonate** action, depicting actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table. Sample of OpenC2 Commands: detonate

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Acting sends the URL to be analyzed in a sandbox.	detonate	openc2:url	process-sandbox	
			(as required)	(optional)	
2	Acting sends the file to the Sandbox for detonation analysis.	detonate	openc2:file	process-sandbox	
			(as required)	(optional)	

4.3.14 **restore**

The **restore** action deletes and/or replaces files, settings, or attributes to return the system to an identical or similar known state.

The **restore** could impact the whole system or return the state of an application or program to its previous state.

Table. Supported Targets and Actuators: restore

Target Type
openc2:device

Actuator Type
process-remediation-service

Table. Modifiers: restore

Modifier	Type	Description	Target Applicability
restore_point		Required. The specific restore point to restore to.	All

Below is a sample usage table for the **restore** action, depicting actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table. Sample of OpenC2 Commands: restore

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Restore a device to a known restore point.	restore	openc2:device (as required)	process-remediation-service (optional)	restore_point

4.3.15 **save**

The **save** action commits data or system state to memory.

Table. Supported Targets and Actuators: save

Target Type	Actuator Type
openc2:email-message	endpoint
openc2:file	network-router
openc2:network-traffic	process-email-service

Table. Modifiers: save

Modifier	Type	Description	Target Applicability
save_to	location	The location to save to.	All

Below is a sample usage table for the **save** action, depicting actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table. Sample of OpenC2 Commands: save

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Save data	save	openc2:file	endpoint	save_to
			(as required)	(optional)	
2	Save an email message	save	openc2:email-message	process-email-service	save_to
			(as required)	(optional)	
3	Save a raw network packet	save	openc2:network-traffic	network-router	save_to
			(as required)	(optional)	

4.3.16 **throttle**

The **throttle** action adjusts the throughput of a data flow.

Table. Supported Targets and Actuators: throttle

Target Type	Actuator Type
openc2:network-traffic	network-router

Table. Modifiers: throttle

Modifier	Type	Description	Target Applicability
None to Date			

Below is a sample usage table for the **throttle** action, depicting actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table. Sample of OpenC2 Commands: throttle

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Limit bandwidth	throttle	openc2:network-traffic	network-router	
			(as required)	(optional)	

4.3.17 **delay**

The **delay** action stops or holds up an activity or data transmittal.

The period of time for the delay can be specified in a modifier to the **delay** action.

Table. Supported Targets and Actuators: delay

Target Type	Actuator Type
openc2:network-traffic	

Table. Modifiers: delay

Modifier	Type	Description	Target Applicability
delay	duration (RFC 3339)	Required. The time delay to add to a network connection.	All

Below is a sample usage table for the **delay** action, depicting actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table. Sample of OpenC2 Commands: delay

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Delay all traffic	delay	openc2:network-traffic		delay
			(as required)		

4.3.18 **substitute**

The **substitute** action replaces all or part of the data, content or payload in the least detectable manner.

The **substitute** action is used in cases where an attack is to be impeded or thwarted in an undetectable manner.

Table. Supported Targets and Actuators: substitute

Target Type
openc2:file openc2:network-traffic

Actuator Type
endpoint network-router

Table. Modifiers: substitute

Modifier	Type	Description	Target Applicability
options		Additional options that specify what to replace and replace with what.	All

Below is a sample usage table for the **substitute** action, depicting actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table. Sample of OpenC2 Commands: substitute

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Overwrite data	substitute	openc2:file	endpoint	options
			(as required)	(optional)	
2	Substitute traffic	substitute	openc2:network-traffic	network-router	options
			(as required)	(optional)	

4.3.19 **copy**

The **copy** action duplicates a file or data flow.

Table. Supported Targets and Actuators: copy

Target Type	Actuator Type
openc2:disk-partition openc2:file openc2:memory openc2:network-traffic	

Table. Modifiers: copy

Modifier	Type	Description	Target Applicability
copy_to	location	The location to copy to.	All

Below is a sample usage table for the **copy** action, depicting actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table. Sample of OpenC2 Commands: copy

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Copy a file	copy	openc2:file		copy_to
			(as required)		
2	Copy network traffic	copy	openc2:network-traffic		copy_to
			(as required)		

4.3.20 **sync**

The **sync** action synchronizes a sensor or actuator with other system components.

Table. Supported Targets and Actuators: sync

Target Type
openc2:device

Actuator Type
endpoint

Table. Modifiers: sync

Modifier	Type	Description	Target Applicability
frequency	duration (RFC 3339)	Optional. The frequency at which to perform the action. The value is the requested time between execution events.	All

Below is a sample usage table for the **sync** action, depicting actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table. Sample of OpenC2 Commands: sync

	Description	Action	Target Target-Specifier	Actuator Actuator-Specifier	Modifier
1	Synchronize an endpoint sensor or actuator to another device	sync	openc2:device (as required)	endpoint (optional)	

4.4 Sensor-Related Actions

These actions are used to control the activities of a sensor in terms of how to collect and provide the sensor data.

4.4.1 distill

The **distill** action tasks the sensor to send a summary or abstraction of the sensing information instead of the raw data feed.

The **distill** action reduces the amount of sensor data. The means of reduction or filtering is indicated by a specifier.

Table. Supported Targets and Actuators: distill

Target Type	Actuator Type
openc2:network-traffic	network-sensor

Table. Modifiers: distill

Modifier	Type	Description	Target Applicability
None to Date			

Below is a sample usage table for the **distill** action, depicting actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table. Sample of OpenC2 Commands: distill

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Filter	distill	openc2:network-traffic	network-sensor	
			(as required)		

4.4.2 **augment**

The **augment** action tasks the sensor to do a level of preprocessing or sense making prior to sending the sensor data.

The means of augmentation and the source of additional data are indicated by a specifier.

Table. Supported Targets and Actuators: augment

Target Type	Actuator Type
openc2:network-traffic	network-sensor

Table. Modifiers: augment

Modifier	Type	Description	Target Applicability
method	enumeration	The specific augmentation function to perform on the network traffic.	openc2:network-traffic

Below is a sample usage table for the **augment** action, depicting actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table. Sample of OpenC2 Commands: augment

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Preprocess network traffic, inter-enclave	augment	openc2:network-traffic		method
			(as required)		
2	Preprocess network traffic, within an enclave	augment	openc2:network-traffic	network-sensor	method
			(as required)	(optional)	

4.5 Effects-Based Actions

Effects-based actions are at a higher level of abstraction and focus on the desired impact rather than a command to execute specific tasks within an enclave. These actions enable coordinating actions, while permitting a local enclave to execute actions in accordance with its local policies and/or capabilities. .

Implementation of an effects-based action requires that the recipient enclave has a decision making capability because an effects-based action permits multiple possible responses.

4.5.1 **investigate**

The **investigate** action tasks the recipient enclave to aggregate and report information as it pertains to an anomaly.

Examples of actions resulting from a received **investigate** could include **scan** multiple machines, **quarantine** an endpoint, or **detonate** a file. These actions are determined by the enclave based on the results of sense-making/analytics and decision-making based on operational constraints and mission needs.

Table. Supported Targets and Actuators: investigate

Target Type	Actuator Type
openc2:device	
openc2:domain-name	
openc2:email-message	
openc2:file	
openc2:ipv4-addr	
openc2:network-traffic	
openc2:process	
openc2:software	
openc2:x509-certificate	

Table. Modifiers: investigate

Modifier	Type	Description	Target Applicability
None to Date			

Below is a sample usage table for the **investigate** action, depicting actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table. Sample of OpenC2 Commands: investigate

	Description	Action	Target Target-Specifier	Actuator Actuator-Specifier	Modifier
1	Investigate the specified IP address for malicious activities	investigate	openc2:ipv4-addr (as required)		[respond_to]
2	Investigate the specified device	investigate	openc2:device (as required)		[respond_to]
3	Investigate the specified domain	investigate	openc2:domain-name (as required)		[respond_to]
4	Investigate the specified email message	investigate	openc2:email-message (as required)		[respond_to]
5	Investigate the specified file(s)	investigate	openc2:file (as required)		[respond_to]

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
6	Investigate the specified hostname	investigate	openc2:domain-name		[respond_to]
			(as required)		

4.5.2 **mitigate**

The **mitigate** action tasks the recipient enclave to circumvent the problem without necessarily eliminating the vulnerability or attack point.

The **mitigate** action implies that the impacts to the enclave's operations should be minimized while addressing the issue. Examples of actions resulting from a received **mitigate** action could include **deny** a URL or process, **scan**, **redirect** traffic to honeypot, or **move**.

Table. Supported Targets and Actuators: mitigate

Target Type	Actuator Type
openc2:device openc2:domain-name openc2:email-message openc2:file openc2:ipv4-addr openc2:network-traffic openc2:process openc2:software openc2:x509-certificate	

Table. Modifiers: mitigate

Modifier	Type	Description	Target Applicability
None to Date			

Below is a sample usage table for the **mitigate** action, depicting actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table. Sample of OpenC2 Commands: mitigate

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Mitigate the specified malicious IP address	mitigate	openc2:ipv4-addr		[respond_to]
			(as required)		
2	Mitigate the specified infected device	mitigate	openc2:device		[respond_to]
			(as required)		
3	Mitigate the specified malicious email message	mitigate	openc2:email-message		[respond_to]
			(as required)		

4.5.3 **remediate**

The **remediate** action tasks the recipient enclave to eliminate the vulnerability or attack point.
A **remediate** action implies that addressing the issue is paramount.

Examples of actions resulting from a received **remediate** action could include **contain**, **quarantine** to a VLAN, **set** authorizations, or **update** patches.

Table. Supported Targets and Actuators: remediate

Target Type	Actuator Type
openc2:device openc2:domain-name openc2:email-message openc2:file openc2:ipv4-addr openc2:network-traffic openc2:process openc2:software openc2:x509-certificate	

Table. Modifiers: remediate

Modifier	Type	Description	Target Applicability
None to Date			

Below is a sample usage table for the **remediate** action, depicting actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table. Sample of OpenC2 Commands: remediate

	Description	Action	Target Target-Specifier	Actuator Actuator-Specifier	Modifier
1	Remediate the specified malicious email message	remediate	openc2:email-message (as required)		[respond_to]
2	Remediate the specified infected hostname	remediate	openc2:domain-name (as required)		[respond_to]

5. Example OpenC2 Use Case

TBSL