Open Command and Control (OpenC2) Language Description Document

Version 1.0 – Release Candidate

22 August 2016

FOREWORD

The Open Command and Control Forum (OpenC2 or the Forum) supports the cyber defense community of interest. The Open Command and Control Forum promotes the global development and adoption of the OpenC2 language and reference material.

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 present its findings and artifacts to recognized standards bodies for the open 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, 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

TABLE OF CONTENTS (Cont.)

TABLE OF CONTENTS (Cont.)

LIST OF FIGURES

This page intentionally left blank.

LIST OF TABLES

LIST OF TABLES (Cont.)

LIST OF TABLES (Cont.)

LIST OF TABLES (Cont.)

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 lexicons and languages used in the interfaces and protocols necessary for machine-to-machine command and control communications in cyber relevant time 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 language and lexicon at a level of abstraction that will enable the coordination and execution of command and control of cyber defense components between domains and within a domain. It is expected that the OpenC2 language can be profiled (e.g., applicable commands, applicable values) by community groups for specific uses like Software Defined Networking.

1.2 Scope

The scope of this document is to create a set of terms that define the actions, 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, components, and modify actions to support a wide range of operational environments.

Future OpenC2 efforts will further refine the controlled vocabulary and define implementation approaches to facilitate interoperable machine to machine communications. These efforts will support the development and promulgation of reference implementations to demonstrate the use and flexibility of the OpenC2 language and promote the incorporation of OpenC2 in cyber defense solutions.

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). 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 will still need to rely on 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, targets, actuators, and modifiers.

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.

Appendix A, Example OpenC2 commands, contains example OpenC2 commands organized in tables by OpenC2 action. These example commands were based on use cases provided by government agencies, critical infrastructure, industry (e.g., security orchestrator, actuator, and sensor) and academia.

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 (i.e. connection oriented, connectionless, pub-sub, hub and spoke, etc.).

OpenC2 was designed to have a concise set of core actions that are extensible through attributes and modifiers to the language 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 are 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, authentication and/or information sharing are beyond the scope of OpenC2 and left to other standards such as STIX.

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) and specific targets and actuators.
- Ensure the OpenC2 command is independent of a message construct that provides transport, identifies priority/ quality of service, and supports security attributes.

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 OpenC2 commands (e.g., inventory of actuators controlled by the enclave, the local security policy, the communication paths and protocols available, and the command structure of the enclave).

Each domain or enclave 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.

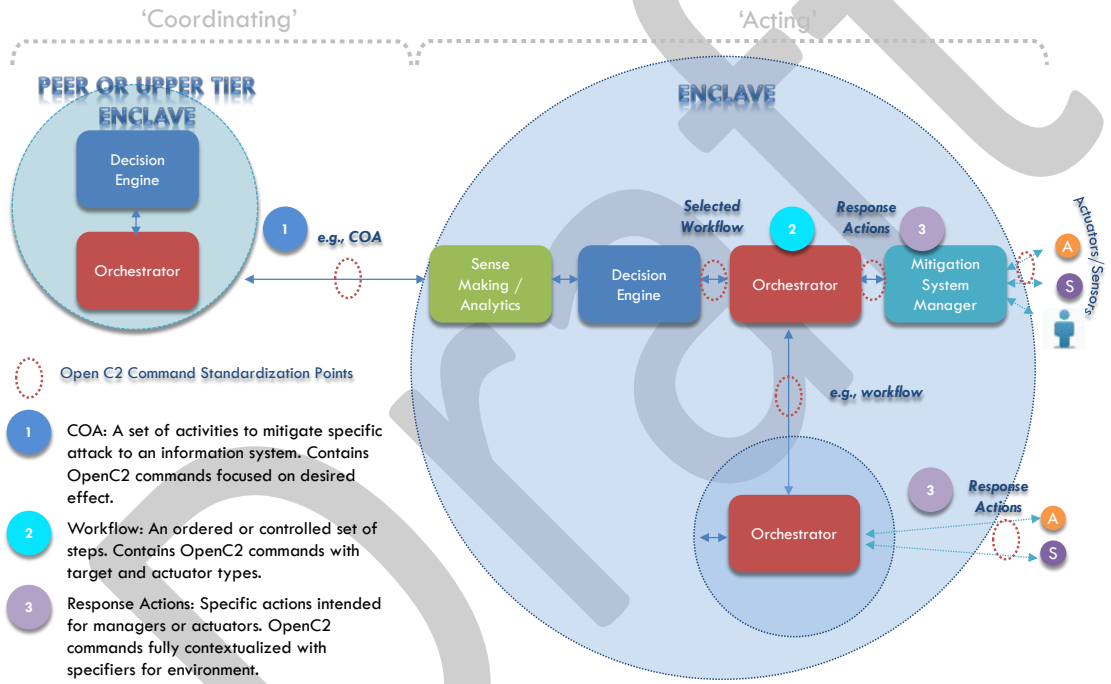


Figure 2-1. OpenC2 Deployment Environments

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 and unambiguous OpenC2 commands to communicate a responsive action between enterprises/enclaves will be more precise and more quickly actionable than a set of recommended steps within a text document (e.g., flash), which must be parsed, analyzed, interpreted, and executed. 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 detailed enough so that the need for specifiers and modifiers is limited.

3.2 Abstract Syntax

Conceptually, an OpenC2 command has the following form:

```
(
      ACTION (
            type = <ACTION_TYPE>
      ),
      TARGET (
            type = <TARGET_TYPE>,
            <target-specifier>
      ),
      ACTUATOR (
            type = <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 and whether they are required, optional or ignored in certain situations. Actual implementation approaches will leverage pre-existing conventions and notations such as XML, JSON, or Type-Length-Value delimitation.

The following table contains the description of 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').
type	Required. The ACTION type is the name of the action.
TARGET	Required for actions, not applicable for responses. 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 namespace.
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. The ACTUATOR type will be defined within the context of a namespace.

Field	Description
actuator-specifier	Optional. The specifier further describes a specific actuator, a list of actuators, or a class of actuators.
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 efficiency.

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 effect 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	
Block traffic to/from specific IP address [effects-based, no actuator specified]; suitable for inter-domain coordination	DENY	Network Connection Source and/or Destination IP Address		
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 Source and/or Destination IP Address	Network (any devices)	
Block traffic at network routers [specify type of network device actuator]; suitable within an enclave	DENY	Network Connection Source and/or Destination IP Address	Network.router (optional)	
Block traffic at specific network router; [specify identity of network router]; suitable within an enclave	DENY	Network Connection Source and/or Destination IP Address	Network.router Router identity	
Block access to bad external IP by null routing; [specify method of performing action]; suitable within an enclave	DENY	Network Connection Source and/or Destination IP Address	Network.router (optional)	Method= blackhole

3.2.1 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 are used.

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

3.2.2 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, and domains.

There will be only one TARGET type per OpenC2 command. By design, OpenC2 will support any data model, but for illustrative purposes OpenC2 TARGETs will reference CybOX objects to the greatest extent practical in this document. Data models will be identified by a namespace before the TARGET type name. Section 3.4 contains a compiled list and definitions of the TARGETs that support the OpenC2 language.

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

There will be only one ACTUATOR type per OpenC2 command. OpenC2 will leverage existing data models to the greatest extent practical (e.g., the Secure Automation and Configuration Management (SACM) working group, ISCM taxonomy). Section 3.5 contains a compiled list of actuators with definitions used to support OpenC2.

[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.

3.2.4 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.

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

3.2.5 Modifiers

"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. 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.

When present, modifiers are always associated with a specific action, however, some modifiers can generally be applied to more than one action or to all OpenC2 actions. A modifier is said to be "actuator-specific", "action-specific", or "universal" depending on the applicability of the modifier within the language.

The modifier can also be used to convey the need for additional status information about the execution of an action. Modifiers can be used to indicate whether the actuator should explicitly acknowledge receipt of the command, respond upon completion of the execution of the command, or provide some other status information. The requested status/information will be carried in a RESPONSE. Refer to Section 4.6.

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	Delay = duration
		Process Object Type	(optional)	
Quarantine a device	CONTAIN	Device	network	where (network segment, vlan)
		Device Object Type	(optional)	
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		

3.3 Actions

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.

3.3.1 Actions that Gather and Convey 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.

3.3.2 Actions that Control Permissions

These actions are used to control permissions and accesses. The permissions and accesses can be for person or non-person entities.

3.3.3 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 adjust configurations, set and update parameters, and modify attributes.

3.3.4 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.

3.3.5 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 task(s) within an enclave. The benefit of including effects-based actions is that it permits a higher level or peer enclave to coordinate actions, while still permitting a local enclave to optimize its workflow for its specific environment in order to achieve the desired result.

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.

3.3.6 Response and Alert

RESPONSE is used to provide data requested as a result of an action. The RESPONSE message will contain the requested data and have a reference to the action that initiated the response. ALERT is used to signal the occurrence of an event or error. It is an unsolicited message that does not reference a previously issued action.

237 **Table 3-5. Summary of Action Definitions**

ACTIONS THAT GATHER AND CONVEY 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.
GET	The GET action tasks an entity to retrieve a specific object.
NOTIFY	The NOTIFY action is used to direct an entity to send information to another entity.
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 something.
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 and process 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 an object (e.g., file, hyperlink) in a manner that isolates the object from assets that support the business or operations of the enclave.
RESTORE	The RESTORE action deletes and/or replaces files, settings, or attributes such that the state of the system is identical to its state at some previous time.
SAVE	The SAVE action commits data or system state to memory.
MODIFY	The MODIFY action augments, enhances, transforms, or changes some aspect of a system.
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.

238

239 **Table 3-5. Summary of Action Definitions (Cont.)**

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. Mitigate implies that the impacts to the enclave's operations should be minimized while addressing the issue.
REMEDIATE	The REMEDIATE action tasks the recipient enclave to eliminate the vulnerability or attack point. Remediate implies that addressing the issue is paramount.
RESPONSE AND ALERT	
RESPONSE	RESPONSE is used to provide any data requested as a result of an action. RESPONSE 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 RESPONSE can be the original requester of the action or to another recipient(s) designated in the modifier of the action.
ALERT	ALERT is used to signal the occurrence of an event.

240

241 3.4 Target Vocabulary

242 The TARGET is the object of the ACTION (or alternatively, the ACTION is performed on the TARGET).
243 OpenC2 will utilize pre-existing data models to provide the namespace for the TARGETs. This document
244 will reference the applicable CybOX objects in the OpenC2 TARGET namespace. However, the OpenC2
245 syntax supports custom or other data models. Refer to the following table for a summary of the OpenC2
246 TARGET Namespaces.

247 **Table 3-6. Target Namespace**

Type	Description	Options
namespace	Used to uniquely identify a set of names so there is no ambiguity; defines the context in which names are defined.	Choice of: • CybOX Version • OpenC2 • Custom

248 Targets include objects such as network connections, URLs, hashes, IP addresses, files, processes, and
249 domains. Refer to the following table for a summary of the supported OpenC2 TARGETs in the CybOX
250 2.1 Namespace (http://cybox.mitre.org/cybox-2).

251 **Table 3-7. Summary of Supported Targets**

Target Type	Description	Target Specifier
cybox:Address	The Address object is intended to specify a cyber address.	cybox:AddressObjectType: Address Value, VLAN Name, VLAN Num
cybox:Device	The Device object is intended to characterize a specific Device.	cybox:DeviceObjectType: Description, Device Type, Manufacturer, Model, Serial Number, Firmware Version, System Details
cybox:Disk	The Disk object is intended to characterize a disk drive.	cybox:DiskObjectType: Disk Name, Disk Size, Free Space, Partition List, Type
cybox:Disk_Partition	The Disk_Partition object is intended to characterize a single partition of a disk drive.	cybox:DiskPartitionObjectType: Created, Device Name, Mount Point, Partition ID, Partition Length, Partition Offset, Space Left, Space Used, Total Space, Type
cybox:Domain_Name	The Domain_Name object is intended to characterize network domain names.	cybox:DomainNameObjectType: Value
cybox:Email_Message	The Email_Message object is intended to characterize an individual email message.	cybox:EmailMessageObjectType: Header, Email Server, Raw Body, Raw Header, Attachments, Links
cybox:File	The File object is intended to characterize a generic file.	cybox:FileObjectType: File Name, File Path, Device Path, Full Path, File Extension, Size In Bytes, Magic Number, File Format, Hashes, Digital Signatures, Modified Time, Accessed Time, Created Time, File Attributes List, Permissions, User Owner, Packer List, Peak Entropy, Sym Links, Byte Runs, Extracted Features, Encryption Algorithm, Decryption Key, Compression Method, Compression Version, Compression Comment
cybox:Hostname	The Hostname object is intended to specify a particular network hostname.	cybox:HostNameObjectType: Hostname Value, Naming System
cybox:Memory	The Memory_Region object is intended to characterize generic memory objects.	cybox:MemoryObjectType: Hashes, Name, Memory Source, Region Size, Block Type, Region Start Address, Region End Address, Extracted Features
cybox:Network_Connection	The Network_Connection object is intended to represent a single network connection.	cybox:NetworkConnectionObjectType: Layer3 Protocol, Layer4 Protocol, Source Socket Address (IP Address/Port), Destination Socket Address, (IP Address/Port)

252

253 **Table 3-7. Summary of Supported Targets (Cont.)**

Target Type	Description	Target Specifier
cybox:Network_Flow	The Network_Flow_Object object provides a summary of network traffic, expressed as flows of multiple packets instead of individual packets, without the packet payload data (i.e. the actual data that was uploaded/downloaded to and from the Dest IP to Source IP as included in packet monitoring tools, such as Wireshark).	cybox:NetworkFlowObjectType: Network Flow Label (Src Socket Address, Dest Socket Address, IP Protocol), Unidirectional Flow Record, Bidirectional Flow Record
cybox:Network_Packet	The Network_Packet object provides the definition of a network packet based on the TCP/IP model/Internet protocol suite. In the TCP/IP stack, "packet" is generally defined as IP header plus payload, but we also include the LinkLayer from the OSI model, which defines the physical network interfaces and routing protocols. The application layer has not yet been defined.	cybox:NetworkPacket: Link Layer (Physical Interface, Logical Protocols), Internet Layer, Transport Layer
cybox:Network_Subnet	The Network_Subnet object is intended to characterize a generic system network subnet.	cybox:NetworkSubnetObjectType: Name, Description, Number Of IP Addresses, Routes
cybox:Port	The Port object is intended to characterize networking ports.	cybox:PortObjectType: Port Value, Layer4 Protocol
cybox:Process	The Process object is intended to characterize system processes.	cybox:ProcessObjectType: PID, Name, Creation Time, Parent PID, Child PID List, Image Info, Argument List, Environment Variable List, Kernel Time, Post List, Network Connection List, Start Time, Status, Username, User Time, Extracted Features
cybox:Product	The Product object is intended to characterize software or hardware products.	cybox:ProductObjectType: Edition, Language, Product, Update, Vendor, Version, Device Details
cybox:Socket_Address	The Socket_Address element is intended to characterize a single network socket address.	cybox:SocketAddressObjectType: IP Address, Hostname, Port
cybox:System	The System object is intended to characterize computer systems (as a combination of both software and hardware).	cybox:SystemObjectType: Available Physical Memory, BIOS Info, Date, Hostname, Local Time, Network Interface List, OS, Processor, Processor Architecture, System Time, Timezone DST, Timezone Standard, Total Physical Memory, Uptime, Username
cybox:URI	The URI object is intended to characterize Uniform Resource Identifiers (URI's).	cybox:URIObjectType Value

254

255 **Table 3-7. Summary of Supported Targets (Cont.)**

Target Type	Description	Target Specifier
cybox:User_Account	The User_Account object is intended to characterize generic user accounts.	cybox:UserAccountObjectType: Full Name, Group List, Home Directory, Last Login, Privilege List, Script Path, Username, User Password Age
cybox:User_Session	The User_Session object is intended to characterize user sessions.	cybox:UserSessionObjectType: Effective Group, Effective Group ID, Effective User, Effective User ID, Login Time, Logout Time
cybox:Volume	The Volume object is intended to characterize generic drive volumes.	cybox:VolumeObjectType: Name, Device Path, File System Type, Total Allocation Units, Sectors Per Allocation Unit, Bytes Per Sector, Actual Available Allocation Units, Creation Time, File System Flag List, Serial Number
cybox:Windows_Registry_Key	Windows_Registry_Key object characterizes windows registry objects, including Keys and Key/Value pairs. [Link](http://msdn.microsoft.com/en-us/library/windows/desktop/ms724871(v=vs.85).asp)	cybox:WindowsRegistryKeyObjectType: Key, Hive, Number Values, Values, Modified Time, Creator Username, Handle List, Number Subkeys, Subkeys, Byte Runs
cybox:Windows_Service	Windows_Service object is intended to characterize Windows services. [Link](http://msdn.microsoft.com/en-us/library/windows/desktop/ms685141(v=vs.85).aspx)	cybox:WindowsServiceObjectType: Description List, Display Name, Group Name, Service Name, Service DLL, Service DLL Certificate Issuer, Service DLL Certificate Subject, Service DLL Hashes, Service DLL Signature Description, Startup Command Line, Startup Type, Service Status, Service Type, Started As
cybox:X509_Certificate	X509_Certificate object represents a public key certificate for use in a public key infrastructure.	cybox:X509CertificateObjectType: Certificate, Raw Certificate, Certificate Signature
openc2:Command	The Command object is intended to characterize an OpenC2 command.	openc2:CommandObjectType: ID
openc2:Data	The Data object is intended to characterize the result of information gathering and publishing activities.	openc2:DataObjectType: Value, Attributes, Search
openc2:OpenC2	The OpenC2 object is a subset of the Data object that represents an Actuator's OpenC2 supported capabilities.	openc2:OpenC2ObjectType: Value, Attributes, Search

256

3.5 Actuator Vocabulary

An ACTUATOR is the entity that puts command and control into motion or action. The ACTUATOR executes the ACTION on the TARGET. Implementers of OpenC2 are encouraged to leverage existing standardized data models for ACTUATORs (e.g., IETF Security Automation and Continuous Monitoring, Information Security Continuous Monitoring (ISCM)) however this document will reference a namespace created by the OpenC2 Forum. Refer to the following table for a summary of the OpenC2 ACTUATOR Namespaces.

Table 3-8. Actuator Namespace

Type	Description	Options
namespace	Used to uniquely identify a set of names so there is no ambiguity; defines the context in which names are defined.	Choice of: • TBD: e.g., ISCM, SACM • OpenC2 • Custom

ACTUATORs fall into classes (e.g., endpoint device, network, services/processes, and human). Refer to the following table for a summary of supported OpenC2 ACTUATORs.

Table 3-9. Summary of Supported Actuators

Actuator Type	Description	Actuator Specifier
endpoint	Endpoint Device	
endpoint.digital-telephone-handset		
endpoint.laptop		
endpoint.pos-terminal		
endpoint.printer		
endpoint.sensor		
endpoint.server		
endpoint.smart-meter		
endpoint.smart-phone		
endpoint.tablet		
endpoint.workstation		
network	Network Platform	
network.bridge		
network.firewall		
network.gateway		
network.guard		
network.hips		
network.hub		
network.ids		
network.ips		
network.modem		
network.nic	Network Interface Card	
network.proxy		
network.router		
network.security_manager		
network.sense_making		
network.sensor		
network.switch		

Table 3-9. Summary of Supported Actuators (Cont.)

Actuator Type	Description	Actuator Specifier
network.vpn	VPN Concentrator/Appliance	
network.wap	Wireless Access Point	
process	Services/Processes	
process.aaa-server		
process.anti-virus-scanner		
process.connection-scanner		
process.directory-service		
process.dns-server		
process.email-service		
process.file-scanner		
process.location-service		
process.network-scanner		
process.remediation-service		
process.reputation-service		
process.sandbox		
process.virtualization-service		
process.vulnerability-scanner		

3.6 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. Modifiers can be used to indicate whether the actuator should explicitly acknowledge receipt of the command, respond upon completion of the execution of the command, or provide some other status information. 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, and are intended to provide additional details for action/target pairs. Action-specific modifiers are identified in the sections detailing out each action.

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
delay	duration	Optional. The time to wait before performing the action.	All
duration	duration	Optional. The period of time that an action is valid.	All
id	string	The unique identifier for the action.	All
response	ack, status	Optional. Indicate the type of response required for the action.	All
datetime	datetime	Optional. The specific date/time to initiate the action.	All

4. EXAMPLE OPENC2 USAGE

This section provides examples of OpenC2 commands corresponding to each OpenC2 action and its applicable targets, actuators, and modifiers. These examples are samples of OpenC2 commands, intended to show the usability and flexibility of the OpenC2 language. A fuller set of example usages can be found in Appendix A.

4.1 Actions that Gather and Convey 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.

This 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). SCAN commands are distinct from the QUERY in that SCAN implies an analytic while a QUERY implies a routine retrieval of data.

Table 4-1. Supported Targets and Actuators: SCAN

Target Type
cybox:Device
cybox:Disk
cybox:Disk_Partition
cybox:Domain_Name
cybox:Email_Message
cybox:File
cybox:Memory
cybox:Network_Connection
cybox:Network_Packet
cybox:Network_Subnet
cybox:Process
cybox:Product
cybox:System
cybox:URI
cybox:User_Account
cybox:User_Session
cybox:Volume

Actuator Type
network.sensor

303 The SCAN action accepts the following modifiers:

304 **Table 4-2. 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

305 Below is a sample of OpenC2 commands to perform a SCAN of targets, utilizing actuators at different
306 levels of specificity, qualified by modifiers to the action as appropriate.

307 **Table 4-3. Sample of OpenC2 Commands: SCAN**

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Scan a device for vulnerabilities	SCAN	cybox:Device	network.sensor	search = CVE
			cybox:DeviceObjectType	(optional)	
2	Scan email messages for malware	SCAN	cybox:Email_Message	network.sensor	search = malware signature
			cybox:EmailMessageObjectType	(optional)	
3	Scan network traffic for malicious activities	SCAN	cybox:Network_Connection	network.sensor	search = network signature
			cybox:NetworkConnectionObjectType	(optional)	

308

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.

This 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 4-4. Supported Targets and Actuators: LOCATE

Target Type
cybox:Address
cybox:Device
cybox:File
cybox:User_Account

Actuator Type
process.location-service

The LOCATE action accepts the following modifiers:

Table 4-5. Modifiers: LOCATE

Modifier	Type	Description	Target Applicability
None to date			

Below is a sample of OpenC2 commands to perform a LOCATE of targets, utilizing actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table 4-6. Sample of OpenC2 Commands: LOCATE

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Geolocate a device	LOCATE	cybox:Device	process.location-service	
			cybox:DeviceObjectType	(optional)	
2	Get location of an IP address	LOCATE	cybox:Address	process.location-service	
			cybox:AddressObjectType	(optional)	

4.1.3 QUERY

The QUERY action initiates a single request for information.

QUERY, 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. QUERY 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:Data. 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 4-7. Supported Targets and Actuators: QUERY

Target Type
openc2:Data
openc2:OpenC2

Actuator Type
endpoint
network.firewall
network.router
process.directory-service

The QUERY action accepts the following modifiers:

Table 4-8. Modifiers: QUERY

Modifier	Type	Description	Target Applicability
response		Where and how to direct the response to the query.	All

Below is a sample of OpenC2 commands to perform a QUERY of targets, utilizing actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table 4-9. Sample of OpenC2 Commands: QUERY

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	List all network connections	QUERY	openc2:Data	network.router	response
			openc2:DataObjectType	(optional)	
2	List running processes on a machine	QUERY	openc2:Data	endpoint	response
			openc2:DataObjectType	(optional)	
3	Request an Actuator's supported OpenC2 capabilities	QUERY	openc2:OpenC2	network.firewall	response
			openc2:OpenC2ObjectType	(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/sensor 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 4-10. Supported Targets and Actuators: REPORT

Target Type	Actuator Type
openc2:Data	TBSL

The REPORT action accepts the following modifiers:

Table 4-11. Modifiers: REPORT

Modifier	Type	Description	Target Applicability
frequency	duration	Optional. The frequency at which to perform the action. The value is the requested time between execution events.	All
report-to	cybox:AddressObjectType	Required. This modifier identifies where to send the report.	All

Below is a sample of OpenC2 commands to perform a REPORT of targets, utilizing actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table 4-12. Sample of OpenC2 Commands: REPORT

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Produce and send a report	REPORT	openc2:Data openc2:DataObjectType		report-to

4.1.5 GET

The GET action tasks an entity to retrieve a specific object.

The location of the object can be designated in the specifier of the TARGET. The entity typically (but not necessarily) retrieves the object outside of the command and control channel.

Table 4-13. Supported Targets and Actuators: GET

Target Type
cybox:Email_Message
cybox:File
cybox:Memory

Actuator Type
endpoint.workstation
network.sense_making

The GET action accepts the following modifiers:

Table 4-14. Modifiers: GET

Modifier	Type	Description	Target Applicability
None to Date			

Below is a sample of OpenC2 commands to perform a GET of targets, utilizing actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table 4-15. Sample of OpenC2 Commands: GET

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Acting gets the potentially malicious email, including URLs and attachments	GET	cybox:Email_Message	network.sense_making	
			cybox:EmailMessageObjectType		
2	Get process file	GET	cybox:File	endpoint.workstation	
			cybox:FileObjectType		

4.1.6 NOTIFY

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

NOTIFY is distinct from REPORT in that NOTIFY is used for time sensitive event notification and carries a sense of persistence.

Table 4-16. Supported Targets and Actuators: NOTIFY

Target Type
cybox:System
cybox:User_Account

Actuator Type
endpoint.server
process.email-service

The NOTIFY action accepts the following modifiers:

Table 4-17. Modifiers: NOTIFY

Modifier	Type	Description	Target Applicability
frequency	duration	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 of OpenC2 commands to perform a NOTIFY of targets, utilizing actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table 4-18. 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	cybox:User_Account cybox:UserAccountObjectType	process.email-service (optional)	message
2	Send a command to notify an external enclave	NOTIFY	cybox:System		message = acknowledge

4.2 Actions that Control Permissions

These actions are used to control permissions and accesses. The permissions and accesses can be for person or non-person entities.

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.

DENY is a superset of current terms such as BLOCK (network perimeter devices) and DENY (user, access to system, access to files).

Table 4-19. Supported Targets and Actuators: DENY

Target Type
cybox:Device
cybox:Network_Connection
cybox:Process
cybox:Product
cybox:URI
cybox:User_Account

Actuator Type
endpoint
network.firewall
network.proxy
network.router
process
process.aaa-server

The DENY action accepts the following modifiers:

Table 4-20. 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.	cybox:Network_Connection, cybox:Product
where	enumeration: internal, perimeter	Optional. The general location within the enclave to perform the DENY action.	cybox:Network_Connection

390 Below is a sample of OpenC2 commands to perform a DENY of targets, utilizing actuators at different
391 levels of specificity, qualified by modifiers to the action as appropriate.

392 **Table 4-21. Sample of OpenC2 Commands: DENY**

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Block traffic to/from specific IP address; suitable for coordinating across multiple enclaves and allowing enclaves to determine most appropriate response	DENY	cybox:Network_Connection		
			cybox:NetworkConnectionObjectType		
2	Block traffic to/from specific IP address at all network firewalls	DENY	cybox:Network_Connection	network.firewall	
			cybox:NetworkConnectionObjectType	(optional)	
3	Block traffic at the network routers	DENY	cybox:Network_Connection	network.router	
			cybox:NetworkConnectionObjectType	(optional)	
4	Block network traffic inside the enclave	DENY	cybox:Network_Connection		
5	Block network traffic at the perimeter	DENY	cybox:Network_Connection		
			cybox:NetworkConnectionObjectType		
6	Block network traffic by ACL	DENY	cybox:Network_Connection	network.router	
			cybox:NetworkConnectionObjectType	(optional)	
7	Block access to a bad external IP address by null routing at the network routers.	DENY	cybox:Network_Connection	network.router	
			cybox:NetworkConnectionObjectType (external IP address)	(optional)	

393

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 4-22. Supported Targets and Actuators: CONTAIN

Target Type
cybox:Device
cybox:File
cybox:Network_Connection
cybox:Process
cybox:User_Account

Actuator Type
endpoint
network

The CONTAIN action accepts the following modifiers:

Table 4-23. Modifiers: CONTAIN

Modifier	Type	Description	Target Applicability
where		Optional. The general location within the enclave to contain the target.	cybox:Device, cybox:File, cybox:Network_Connection, cybox:Process, cybox:User_Account

Below is a sample of OpenC2 commands to perform a CONTAIN of targets, utilizing actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table 4-24. Sample of OpenC2 Commands: CONTAIN

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Quarantine a file, general	CONTAIN	cybox:File		
			cybox:FileObjectType		
2	Quarantine a file	CONTAIN	cybox:File	endpoint	where
			cybox:FileObjectType	(optional)	
3	Contain a user or group, general	CONTAIN	cybox:User_Account		
			cybox:UserAccountType		
4	Contain network traffic to a honeynet, general	CONTAIN	cybox:Network_Connection		
			cybox:NetworkConnectionObjectType		

4.2.3 ALLOW

The ALLOW action permits the access to or execution of something.

An ALLOW action is typically associated with something that was previously denied (e.g., block, quarantine).

Table 4-25. Supported Targets and Actuators: ALLOW

Target Type
cybox:Device
cybox:File
cybox:Network_Connection
cybox:Process
cybox:Product
cybox:URI
cybox:User_Account

Actuator Type
endpoint
network.firewall
network.proxy
network.router
process.aaa-server

The ALLOW action accepts the following modifiers:

Table 4-26. Modifiers: ALLOW

Modifier	Type	Description	Target Applicability
delay	duration	Optional. The time to wait before performing the action.	cybox:Device, cybox:User_Account
permissions		Optional. Specific permissions to be granted to the user.	cybox:User_Account
where	enumeration: internal, perimeter	Optional. The general location within the enclave to perform the DENY action.	cybox:Network_Connection

Below is a sample of OpenC2 commands to perform an ALLOW of targets, utilizing actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

415 **Table 4-27. 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	cybox:Network_Connection		
			cybox:NetworkConnectionObjectType		
2	Unblock traffic to/from specific IP address at all network firewalls	ALLOW	cybox:Network_Connection	network.firewall	
			cybox:NetworkConnectionObjectType	(optional)	
3	Unblock traffic at the network routers	ALLOW	cybox:Network_Connection	network.router	
			cybox:NetworkConnectionObjectType	(optional)	
4	Unblock network traffic inside the enclave	ALLOW	cybox:Network_Connection		where = internal
			cybox:NetworkConnectionObjectType		
5	Delay Machine Authentication	ALLOW	cybox:Device	process.aaa-server	delay = <TIME>
			cybox:DeviceObjectType	(optional)	
6	Unquarantine a file	ALLOW	cybox:File	endpoint	
			cybox:FileObjectType	(optional)	

416

4.3 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 adjust configurations, set and update parameters, and modify attributes.

4.3.1 START

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

Table 4-28. Supported Targets and Actuators: START

Target Type
cybox:Disk_Partition
cybox:Process
cybox:Product
cybox:System

Actuator Type
endpoint
network
process.virtualization-service

The START action accepts the following modifiers:

Table 4-29. Modifiers: START

Modifier	Type	Description	Target Applicability
delay	duration	Optional. The time to wait before performing the action.	All
method	enumeration: spawn		cybox:Process

Below is a sample of OpenC2 commands to perform a START of targets, utilizing actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table 4-30. Sample of OpenC2 Commands: START

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Start Process, general	START	cybox:Process		
			cybox:ProcessObjectType		
2	Start Process	START	cybox:Process	endpoint	
			cybox:ProcessObjectType	(optional)	
3	Start Process with Delay	START	cybox:Process	endpoint	delay
			cybox:ProcessObjectType	(optional)	
4	Spawn Process	START	cybox:Process	endpoint	method = spawn
			cybox:ProcessObjectType	(optional)	

4.3.2 STOP

The STOP action halts a system or ends an activity.

The STOP OpenC2 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 will be a graceful shutdown. Action modifiers are used to indicate immediate or atypical STOP actions.

Table 4-31. Supported Targets and Actuators: STOP

Target Type
cybox:Device
cybox:Disk_Partition
cybox:Process
cybox:System
cybox:User_Account
cybox:User_Session
cybox:Windows_Service

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

The STOP action accepts the following modifiers:

Table 4-32. 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 of OpenC2 commands to perform a STOP of targets, utilizing actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table 4-33. Sample of OpenC2 Commands: STOP

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Shutdown a system	STOP	cybox:Device	endpoint	[method = graceful]
			cybox:DeviceObjectType	(optional)	
2	Shutdown a system, immediate	STOP	cybox:Device	endpoint	method = immediate
			cybox:DeviceObjectType	(optional)	
3	Logoff User: Logoff all the sessions of a particular user from the machine	STOP	cybox:User_Account	endpoint	[method = graceful]
			cybox:UserAccountObjectType	(optional)	
4	Stop a vm	STOP	cybox:System	process.virtualization-service	[method = graceful]
			cybox:SystemObjectType	(optional)	

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 4-34. Supported Targets and Actuators: RESTART

Target Type
cybox:Process
cybox:System

Actuator Type
endpoint
process.virtualization-service

The RESTART action accepts the following modifiers:

Table 4-35. Modifiers: RESTART

Modifier	Type	Description	Target Applicability
delay	duration	Optional. The time to wait before performing the action.	All
frequency	string	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 of OpenC2 commands to perform a RESTART of targets, utilizing actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table 4-36. Sample of OpenC2 Commands: RESTART

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Restart device (system)	RESTART	cybox:System		
			cybox:SystemObjectType		
2	Restart device (system) with different OS	RESTART	cybox:System		options, e.g., OS
			cybox:SystemObjectType		
3	Restart VM	RESTART	cybox:System	process.virtualization-service	
			cybox:SystemObjectType	(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 4-37. Supported Targets and Actuators: PAUSE

Target Type
cybox:Process
cybox:System

Actuator Type
endpoint
process.virtualization-service

The PAUSE action accepts the following modifiers:

Table 4-38. Modifiers: PAUSE

Modifier	Type	Description	Target Applicability
duration	duration	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 of OpenC2 commands to perform a PAUSE of targets, utilizing actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table 4-39. Sample of OpenC2 Commands: PAUSE

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Pause device (system)	PAUSE	cybox:System		[method = sleep]
			cybox:SystemObjectType		
2	Hibernate device (system)	PAUSE	cybox:System		method = hibernate
			cybox:SystemObjectType		
3	Pause VM	PAUSE	cybox:System	process.virtualization-service	
			cybox:SystemObjectType	(optional)	
4	Pause a system or VM for a specified duration	PAUSE	cybox:System		duration = duration
			cybox:SystemObjectType		

4.3.5 RESUME

The RESUME action starts a system or activity from a paused state.

RESUME is only meaningful after a PAUSE command.

Table 4-40. Supported Targets and Actuators: RESUME

Target Type
cybox:Process
cybox:System

Actuator Type
endpoint
process.virtualization-service

The RESUME action accepts the following modifiers:

Table 4-41. Modifiers: RESUME

Modifier	Type	Description	Target Applicability
None to Date			

Below is a sample of OpenC2 commands to perform a RESUME of targets, utilizing actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table 4-42. Sample of OpenC2 Commands: RESUME

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Resume device (system)	RESUME	cybox:System		
			cybox:SystemObjectType		
2	Resume VM	RESUME	cybox:System	process.virtualization-service	
			cybox:SystemObjectType	(optional)	

4.3.6 CANCEL

The CANCEL action invalidates a previously issued action.

CANCEL must be associated with a previously issued command through the “command-ref” modifier. This action should not be considered an undo. It can set the validity period to immediately end or it could define a future duration for which the action is valid.

Table 4-43. Supported Targets and Actuators: CANCEL

Target Type
openc2:Command

Actuator Type
endpoint
network
process

The CANCEL action accepts the following modifiers:

Table 4-44. Modifiers: CANCEL

Modifier	Type	Description	Target Applicability
command-ref	string	The reference to the associated command that is to be cancelled.	openc2:Command
duration	duration	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 of OpenC2 commands to perform a CANCEL of targets, utilizing actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table 4-45. 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 = command reference
			openc2:CommandObjectType		
2	Cancel a previously issued command, directed to a specific actuator (endpoint)	CANCEL	openc2:Command	endpoint	command-ref = command reference
			openc2:CommandObjectType	(optional)	

4.3.7 SET

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

Typically this action is specified by a configuration item such as a sensor setting or privilege level and the command will have specifiers. SET commands are intended for specific individual changes to the entity and the parameters are communicated in the C2 channel.

Table 4-46. Supported Targets and Actuators: SET

Target Type
cybox:File
cybox:Process
cybox:System
cybox:User_Account
cybox:Windows_Registry_Key
openc2:Data

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

The SET action accepts the following modifiers:

Table 4-47. Modifiers: SET

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

Below is a sample of OpenC2 commands to perform a SET of targets, utilizing actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table 4-48. Sample of OpenC2 Commands: SET

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Set registry key value	SET	cybox:Windows_Registry_Key	endpoint.workstation	set-to
			cybox:WindowsRegistryKeyObjectType	(optional)	
2	Set file permissions	SET	cybox:File	process.directory-service	set-to
			cybox:FileObjectType	(optional)	
3	Set user rights	SET	cybox:User_Account	process.directory-service	set-to
			cybox:UserAccountObjectType	(optional)	

4.3.8 UPDATE

The UPDATE action instructs the component to retrieve and process 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. UPDATE actions typically do not need to include details such as reboot or restart. It is incumbent upon the OpenC2 compliant devices to include implementation details. Modifiers such as 'immediate' and specifiers such as the path for the software may be added.

Table 4-49. Supported Targets and Actuators: UPDATE

Target Type
cybox:Device
cybox:Product

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

The UPDATE action accepts the following modifiers:

Table 4-50. Modifiers: UPDATE

Modifier	Type	Description	Target Applicability
frequency	duration	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 of OpenC2 commands to perform an UPDATE of targets, utilizing actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table 4-51. Sample of OpenC2 Commands: UPDATE

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Install software	UPDATE	cybox:Product	endpoint	source
			cybox:ProductObjectType	(optional)	
2	Install patch	UPDATE	cybox:Product	endpoint	source
			cybox:ProductObjectType	(optional)	
3	Update signature file (anti-virus)	UPDATE	cybox:Product	process.anti-virus-scanner	source
			cybox:ProductObjectType	(optional)	

4.3.9 MOVE

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

MOVE is distinct from CONTAIN in that CONTAIN implies a desired effect of isolation and MOVE supports the more general case.

Table 4-52. Supported Targets and Actuators: MOVE

Target Type
cybox:File
openc2:Data

Actuator Type
TBSL

The MOVE action accepts the following modifiers:

Table 4-53. Modifiers: MOVE

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

Below is a sample of OpenC2 commands to perform a MOVE of targets, utilizing actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table 4-54. Sample of OpenC2 Commands: MOVE

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Move file/directory	MOVE	cybox:File cybox:FileObjectType		move-to

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. REDIRECT is distinct from MOVE in that it encompasses the entire flow rather than a single instance, item or object. MOVE supports the more atomic case.

Table 4-55. Supported Targets and Actuators: REDIRECT

Target Type
cybox:Network_Connection
cybox:URI

Actuator Type
network.router

The REDIRECT action accepts the following modifiers:

Table 4-56. 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 of OpenC2 commands to perform a REDIRECT of targets, utilizing actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table 4-57. 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	cybox:Network_Connection		where
			cybox:NetworkConnectionObjectType		
2	Redirect traffic to a honeypot at a specific router	REDIRECT	cybox:Network_Connection	network.router	where
			cybox:NetworkConnectionObjectType		
3	Cancel traffic redirection; suitable for coordinating across multiple enclaves and allowing enclaves to determine most appropriate response	REDIRECT	cybox:Network_Connection		where = null
			cybox:NetworkConnectionObjectType		

4.3.11 DELETE

The DELETE action removes data and files.

Table 4-58. Supported Targets and Actuators: DELETE

Target Type
cybox:Email_Message
cybox:File
openc2:Data

Actuator Type
endpoint
network.firewall
process.email-service

The DELETE action accepts the following modifiers:

Table 4-59. Modifiers: DELETE

Modifier	Type	Description	Target Applicability
None to Date			All

Below is a sample of OpenC2 commands to perform a DELETE of targets, utilizing actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table 4-60. Sample of OpenC2 Commands: DELETE

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Delete file, inter-enclave	DELETE	cybox:File cybox:FileObjectType		
2	Delete file, within an enclave	DELETE	cybox:File cybox:FileObjectType	endpoint (optional)	
3	Delete email, inter-enclave	DELETE	cybox:Email_Message cybox:EmailMessageObjectType		
4	Delete email from exchange server	DELETE	cybox:Email_Message cybox:EmailMessageObjectType	process.email-service (optional)	

4.3.12 SNAPSHOT

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

Table 4-61. Supported Targets and Actuators: SNAPSHOT

Target Type	Actuator Type
cybox:System	process.virtualization-service

The SNAPSHOT action accepts the following modifiers:

Table 4-62. Modifiers: SNAPSHOT

Modifier	Type	Description	Target Applicability
None to Date			

Below is a sample of OpenC2 commands to perform a SNAPSHOT of targets, utilizing actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table 4-63. Sample of OpenC2 Commands: SNAPSHOT

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Take a snapshot of a VM	SNAPSHOT	cybox:System cybox:SystemObjectType	process.virtualization-service (optional)	

4.3.13 DETONATE

The DETONATE action executes and observes the behavior of an object (e.g., file, hyperlink) in a manner that isolates the object from assets that support the business or operations of the enclave.

DETONATE is distinct from CONTAIN in that DETONATE includes an execution and analytic component rather than just isolation.

Table 4-64. Supported Targets and Actuators: DETONATE

Target Type
cybox:File
cybox:URI

Actuator Type
process.sandbox

The DETONATE action accepts the following modifiers:

Table 4-65. Modifiers: DETONATE

Modifier	Type	Description	Target Applicability
None to Date			

Below is a sample of OpenC2 commands to perform a DETONATE of targets, utilizing actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table 4-66. 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	cybox:URI	process.sandbox	
			cybox:URIObjectType	(optional)	
2	Acting sends the file to the Sandbox for detonation analysis.	DETONATE	cybox:File	process.sandbox	
			cybox:FileObjectType	(optional)	

4.3.14 RESTORE

The RESTORE action deletes and/or replaces files, settings, or attributes such that the state of the system is identical to its state at some previous time.

The RESTORE could impact the whole system or return the state of an application or program to its previous state. RESTORE can be used to undo a previous action.

Table 4-67. Supported Targets and Actuators: RESTORE

Target Type	Actuator Type
cybox:Device	process.remediation-service

The RESTORE action accepts the following modifiers:

Table 4-68. Modifiers: RESTORE

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

Below is a sample of OpenC2 commands to perform a RESTORE of targets, utilizing actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table 4-69. Sample of OpenC2 Commands: RESTORE

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Restore a device to a known restore point.	RESTORE	cybox:Device	process.remediation-service	restore-point
			cybox:DeviceObjectType	(optional)	

4.3.15 SAVE

The SAVE action commits data or system state to memory.

Table 4-70. Supported Targets and Actuators: SAVE

Target Type
cybox:Email_Message
cybox:File
cybox:Network_Packet

Actuator Type
endpoint
network.router
process.email-service

The SAVE action accepts the following modifiers:

Table 4-71. Modifiers: SAVE

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

Below is a sample of OpenC2 commands to perform a SAVE of targets, utilizing actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table 4-72. Sample of OpenC2 Commands: SAVE

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Save data	SAVE	cybox:File cybox:FileObjectType	endpoint (optional)	save-to
2	Save an email message	SAVE	cybox:Email_Message cybox:EmailMessageObjectType	process.email-service (optional)	save-to
3	Save a raw network packet	SAVE	cybox:Network_Packet cybox:NetworkPacketObjectType	network.router (optional)	save-to

4.3.16 MODIFY

The MODIFY action augments, enhances, transforms, or changes some aspect of a system.

MODIFY is used to change the attributes or behavior of some system element without stopping it or removing it from the system. MODIFY is a superset of commands such as set and rename.

Table 4-73. Supported Targets and Actuators: MODIFY

Target Type
cybox:Device
cybox:File
cybox:Process
cybox:Product
cybox:System
cybox:User_Account

Actuator Type
endpoint
process.directory-service

The MODIFY action accepts the following modifiers:

Table 4-74. Modifiers: MODIFY

Modifier	Type	Description	Target Applicability
modify-to		The new value resulting from the modification	All

Below is a sample of OpenC2 commands to perform a MODIFY of targets, utilizing actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table 4-75. Sample of OpenC2 Commands: MODIFY

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Modify a file	MODIFY	cybox:File cybox:FileObjectType		modify-to
2	Modify a device's configuration	MODIFY	cybox:Device cybox:DeviceObjectType	endpoint (optional)	modify-to
3	Modify user account privileges	MODIFY	cybox:User_Account cybox:UserAccountObjectType	process.directory-service (optional)	modify-to

4.3.17 THROTTLE

The THROTTLE action adjusts the throughput of a data flow.

Table 4-76. Supported Targets and Actuators: THROTTLE

Target Type	Actuator Type
cybox:Network_Connection	network.router

The THROTTLE action accepts the following modifiers:

Table 4-77. Modifiers: THROTTLE

Modifier	Type	Description	Target Applicability
None to Date			

Below is a sample of OpenC2 commands to perform a THROTTLE of targets, utilizing actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table 4-78. Sample of OpenC2 Commands: THROTTLE

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Limit bandwidth	THROTTLE	cybox:Network_Connection cybox:NetworkConnectionObjectType	network.router (optional)	

4.3.18 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 4-79. Supported Targets and Actuators: DELAY

Target Type	Actuator Type
cybox:Network_Connection	TBSL

The DELAY action accepts the following modifiers:

Table 4-80. Modifiers: DELAY

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

Below is a sample of OpenC2 commands to perform a DELAY of targets, utilizing actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table 4-81. Sample of OpenC2 Commands: DELAY

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Delay all traffic	DELAY	cybox:Network_Connection cybox:NetworkConnectionObjectType		delay

4.3.19 SUBSTITUTE

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

SUBSTITUTE is used in cases where an attack is to be impeded or thwarted in an undetectable manner.

Table 4-82. Supported Targets and Actuators: SUBSTITUTE

Target Type
cybox:File
cybox:Network_Connection

Actuator Type
endpoint
network.router

The SUBSTITUTE action accepts the following modifiers:

Table 4-83. Modifiers: SUBSTITUTE

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

Below is a sample of OpenC2 commands to perform a SUBSTITUTE of targets, utilizing actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table 4-84. Sample of OpenC2 Commands: SUBSTITUTE

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Overwrite data	SUBSTITUTE	cybox:File	endpoint	options
			cybox:FileObjectType	(optional)	
2	Substitute traffic	SUBSTITUTE	cybox:Network_Connection	network.router	options
			cybox:NetworkConnectionObjectType	(optional)	

4.3.20 COPY

The COPY action duplicates a file or data flow.

Table 4-85. Supported Targets and Actuators: COPY

Target Type
???
cybox:Disk_Partition
cybox:File
cybox:Memory
cybox:Network_Connection
cybox:Network_Flow

Actuator Type
TBSL

The COPY action accepts the following modifiers:

Table 4-86. Modifiers: COPY

Modifier	Type	Description	Target Applicability
copy-to		The location to copy to.	All
where		Optional. The system to copy from.	cybox:Disk_Partition, cybox:Memory

Below is a sample of OpenC2 commands to perform a COPY of targets, utilizing actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table 4-87. Sample of OpenC2 Commands: COPY

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Copy a file	COPY	cybox:File		where, copy-to
			cybox:FileObjectType		
2	Copy network traffic	COPY	cybox:Network_Connection		copy-to
			cybox:NetworkConnectionObjectType		

4.3.21 SYNC

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

Table 4-88. Supported Targets and Actuators: SYNC

Target Type	Actuator Type
cybox:Device	endpoint

The SYNC action accepts the following modifiers:

Table 4-89. Modifiers: SYNC

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

Below is a sample of OpenC2 commands to perform a SYNC of targets, utilizing actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table 4-90. Sample of OpenC2 Commands: SYNC

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Synchronize an endpoint sensor or actuator to another device	SYNC	cybox:Device	endpoint	
			cybox:DeviceObjectType	(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 4-91. Supported Targets and Actuators: DISTILL

Target Type	Actuator Type
cybox:Network_Connection	network.sensor

The DISTILL action accepts the following modifiers:

Table 4-92. Modifiers: DISTILL

Modifier	Type	Description	Target Applicability
None to Date			

Below is a sample of OpenC2 commands to perform a DISTILL of targets, utilizing actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table 4-93. Sample of OpenC2 Commands: DISTILL

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Filter	DISTILL	cybox:Network_Connection	network.sensor	
			cybox:NetworkConnectionObjectType		

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 4-94. Supported Targets and Actuators: AUGMENT

Target Type	Actuator Type
cybox:Network_Connection	network.sensor

The AUGMENT action accepts the following modifiers:

Table 4-95. Modifiers: AUGMENT

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

Below is a sample of OpenC2 commands to perform an AUGMENT of targets, utilizing actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table 4-96. Sample of OpenC2 Commands: AUGMENT

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Preprocess network traffic, inter-enclave	AUGMENT	cybox:Network_Connection		method
			cybox:NetworkConnectionObjectType		
2	Preprocess network traffic, within an enclave	AUGMENT	cybox:Network_Connection	network.sensor	method
			cybox:NetworkConnectionObjectType	(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 task(s) within an enclave. The benefit of including effects-based actions is that it permits a higher level or peer enclave to coordinate actions, while still permitting a local enclave to optimize its workflow for its specific environment in order to achieve the desired result.

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 OpenC2 command 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 4-97. Supported Targets and Actuators: INVESTIGATE

Target Type
cybox:Address
cybox:Device
cybox:Domain_Name
cybox:Email_Message
cybox:File
cybox:Hostname
cybox:Network_Connection
cybox:Port
cybox:Process
cybox:Product
cybox:System
cybox:X509_Certificate

Actuator Type
TBSL

The INVESTIGATE action accepts the following modifiers:

Table 4-98. Modifiers: INVESTIGATE

Modifier	Type	Description	Target Applicability
report-to	cybox:AddressObjectType	Optional. If requested, this modifier identifies where to report the results of the investigation.	All

691 Below is a sample of OpenC2 commands to perform an INVESTIGATE of targets, utilizing actuators at
692 different levels of specificity, qualified by modifiers to the action as appropriate.

693 **Table 4-99. Sample of OpenC2 Commands: INVESTIGATE**

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Investigate the specified IP address for malicious activities	INVESTIGATE	cybox:Address		[report-to]
			cybox:AddressObjectType		
2	Investigate the specified device	INVESTIGATE	cybox:Device		[report-to]
			cybox:DeviceObjectType		
3	Investigate the specified domain	INVESTIGATE	cybox:Domain_Name		[report-to]
			cybox:DomainNameObjectType		
4	Investigate the specified email message	INVESTIGATE	cybox:Email_Message		[report-to]
			cybox:EmailMessageObjectType		
5	Investigate the specified file(s)	INVESTIGATE	cybox:File		[report-to]
			cybox:FileObjectType		
6	Investigate the specified hostname	INVESTIGATE	cybox:Hostname		[report-to]
			cybox:HostnameObjectType		

694

4.5.2 MITIGATE

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

Mitigate implies that the impacts to the enclave's operations should be minimized while addressing the issue.

Examples of actions resulting from a received MITIGATE OpenC2 command could include deny a URL or process, scan, redirect traffic to honeypot, or move.

Table 4-100. Supported Targets and Actuators: MITIGATE

Target Type
cybox:Address
cybox:Device
cybox:Email_Message
cybox:File
cybox:Hostname
cybox:Network_Connection
cybox:Process
cybox:Product
cybox:System
cybox:X509_Certificate

Actuator Type
TBSL

The MITIGATE action accepts the following modifiers:

Table 4-101. Modifiers: MITIGATE

Modifier	Type	Description	Target Applicability
None to Date			

Below is a sample of OpenC2 commands to perform a MITIGATE of targets, utilizing actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table 4-102. Sample of OpenC2 Commands: MITIGATE

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Mitigate the specified malicious IP address	MITIGATE	cybox:Address cybox:AddressObjectType		[report-to]
2	Mitigate the specified infected device	MITIGATE	cybox:Device cybox:DeviceObjectType		[report-to]
3	Mitigate the specified malicious email message	MITIGATE	cybox:Email_Message cybox:EmailMessageObjectType		[report-to]

4.5.3 REMEDIATE

The REMEDIATE action tasks the recipient enclave to eliminate the vulnerability or attack point.

Remediate implies that addressing the issue is paramount.

Examples of actions resulting from a received REMEDIATE OpenC2 command could include contain/quarantine to a VLAN, set authorizations, redirect URL to quarantine portal, get new configuration, or update patches.

Table 4-103. Supported Targets and Actuators: REMEDIATE

Target Type
cybox:Address
cybox:Device
cybox:Email_Message
cybox:File
cybox:Hostname
cybox:Network_Connection
cybox:Process
cybox:Product
cybox:System
cybox:X509_Certificate

Actuator Type
TBSL

The REMEDIATE action accepts the following modifiers:

Table 4-104. Modifiers: REMEDIATE

Modifier	Type	Description	Target Applicability
None to Date			

Below is a sample of OpenC2 commands to perform a REMEDIATE of targets, utilizing actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table 4-105. Sample of OpenC2 Commands: REMEDIATE

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Remediate the specified malicious email message	REMEDIATE	cybox:Email_Message		[report-to]
			cybox:EmailMessageObjectType		
2	Remediate the specified infected hostname	REMEDIATE	cybox:Hostname		[report-to]
			cybox:HostnameObjectType		

4.6 Response and Alert

RESPONSE is used to provide data requested as a result of an action. The RESPONSE message will contain the requested data and have a reference to the action that initiated the response. ALERT is used to signal the occurrence of an event or error. It is an unsolicited message that does not reference a previously issued action.

4.6.1 RESPONSE

RESPONSE is used to provide any data requested as a result of an action. RESPONSE 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 RESPONSE can be the original requester of the action or to another recipient(s) designated in the modifier of the action.

The RESPONSE action accepts the following modifiers:

Table 4-106. Modifiers: RESPONSE

Modifier	Type	Description	Target Applicability
command-ref		The reference to the associated command that is in response to.	N/A
type	enumeration: acknowledgement, status, query	The type of response.	N/A
value		The value of the response.	N/A

Below is a sample of OpenC2 commands to perform a RESPONSE of targets, utilizing actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table 4-107. Sample of OpenC2 Commands: RESPONSE

	Description	Action	Modifier
1	Acknowledge the receipt of an action	RESPONSE	type = acknowledge, command-ref = command reference
2	Signal completion of an action	RESPONSE	type = status, value = complete, command-ref = command reference
3	Provide the status of an action	RESPONSE	type = status, value = current, command-ref = command reference

4.6.2 ALERT

ALERT is used to signal the occurrence of an event.

The ALERT action accepts the following modifiers:

Table 4-108. Modifiers: ALERT

Modifier	Type	Description	Target Applicability
type	enumeration	The type of alert.	N/A
value		Additional data associated with the alert.	N/A

Below is a sample of OpenC2 commands to perform an ALERT of targets, utilizing actuators at different levels of specificity, qualified by modifiers to the action as appropriate.

Table 4-109. Sample of OpenC2 Commands: ALERT

	Description	Action	Modifier
1	An actuator sends an alert as the result of some condition.	ALERT	type, value
2	A sensor sends an alert as the result of some condition.	ALERT	type, value

This page intentionally left blank.

5. EXAMPLE OPENC2 USE CASE: MITIGATE EVIL DOMAIN

5.1 Description

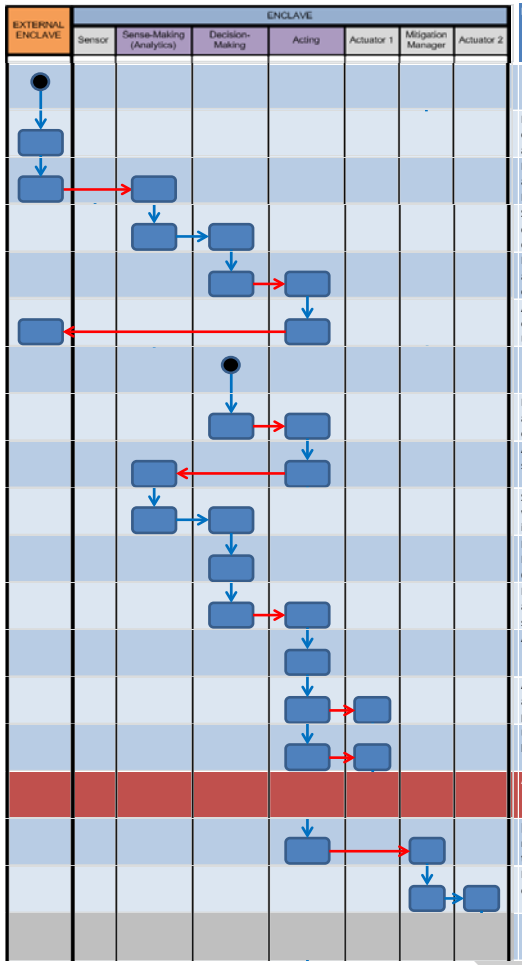
A cyber threat analyst reviews structured and unstructured information regarding cyber threat activity from a variety of manual or automated input sources. The analyst sets out to understand the nature of relevant threats, identify them, and fully characterize them such that all of the relevant knowledge of the threat can be fully expressed and evolved over time. This relevant knowledge includes threat-related actions, behaviors, capabilities, intents, and attributed actors. From this understanding and characterization, the analyst at an upper tier determines that a domain is “evil”. The upper tier notifies lower level enclaves of the need to mitigate against this evil domain.

5.2 Stakeholders/Goals

Upper Tier shares threat intelligence with lower tier. Lower tier acts on the shared threat intelligence.

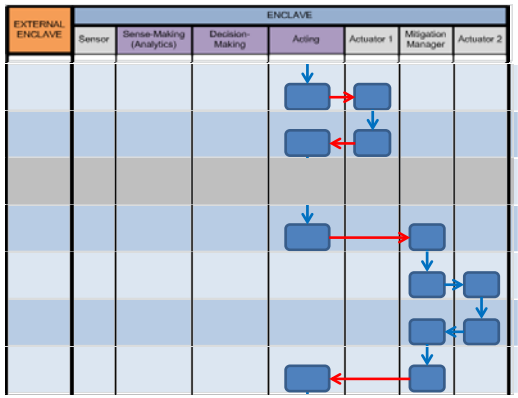
5.3 Preconditions

Upper and lower tier have pre-established a trust relationship.



Description	Step	ACTION	TARGET	TARGET-SPECIFIER	ACTUATOR (Sensor, Actuator, Mitigation Manager)	ACTUATOR-SPECIFIER	MODIFIER
	1						
External Enclave determines that there is a need to coordinate with other networks for the mitigation against an evil domain.	2						
External Enclave sends the COA containing an action to mitigate against the evil domain to the Enclave.	3	MITIGATE	cybox:Domain_Name	Value = "mymalware.com"			
Sense-Making at the Enclave provides the COA containing an action to mitigate against the evil domain to Decision-Making.	4						
Decision-Making commands Acting to send an application level Acknowledgement for the receipt of the COA.	5	NOTIFY	cybox:System	cybox:SystemObjectType			message = acknowledge
Acting sends the Acknowledgement for the receipt of the COA to the External Enclave. <Dependency: UUID for command from External Enclave>	6	RESPONSE					type = acknowledge, command = MITIGATE reference
	7						
Decision-Making commands Acting to request additional Sense-Making analytics and environmental conditions.	8	SCAN	cybox:Domain_Name	Value = "mymalware.com"			
Acting requests additional Sense-Making to support the Decision-Making process.	9	SCAN	cybox:Domain_Name	Value = "mymalware.com"	network.sense_making	(specifier)	
Sense-Making at the Enclave combines the COA with additional analytic results and provides the information to Decision-Making.	10						
Decision-Making utilizes the results of Sense-Making to determine the response to the received COA.	11						
Decision-Making commands Acting to execute an action plan comprised of one or more mitigation steps.	12	START	cybox:Process	Name = "MITIGATE Plan XX"			
Acting contextualized the mitigation steps.	13						
Acting notifies an authorized user of the new action plan. Acting awaits approval before proceeding.	14	NOTIFY	cybox:User_Account	Username	endpoint.server	(specifier)	message = "approval request"
Upon approval, Acting commands a firewall to block packets.	15	DENY	cybox:Network_Connection	Source Socket Address: IP Address = "8.8.8.8", Port = [80, 443])	network.firewall	(specifier)	
ALTERNATIVE ACTIONS							
[Alternative] Acting commands host remediation manager to update all endpoints' browser settings to block the evil domain.	16	DENY	cybox:Domain_Name	Value = "mymalware.com"	process.remediation-service	(specifier)	
Mitigation Manager commands the update of each endpoint's browser settings.	17						

Figure 5-1. Scenario Diagram: Mitigate Evil Domain



Description	Step	ACTION	TARGET	TARGET-SPECIFIER	ACTUATOR (Sensor, Actuator, Mitigation Manager)	ACTUATOR-SPECIFIER	MODIFIER
[Alternative] Acting issues a 'block' command so that any access to the domain will be null routed or redirected to an internal DNS server(Perimeter	18	DENY	cybox:Network_Connection	Source Socket Address: IP Address = "8.8.8.8", Port = [80, 443])	network.router	(specifier)	method = "sinkhole"
Actuator notifes Acting of the status of the execution of the OpenC2 command (e.g., completed, failed)	19	RESPONSE				type = status, value = complete, command = DENY reference	
[Alternative] Acting issues a command to block access to a bad external IP so that any access to the ip will be denied (Perimeter Blocking using	20	DENY	cybox:Network_Connection	Source Socket Address: IP Address = "8.8.8.8", Port = [80, 443])	network.router	(specifiy perimeter routers)	method = "acl"
Mitigation Manager commands the update of the ACL at each perimeter firewall (vendor specific)	21						
Actuator notifes the Mitigation Manager of the status of the execution of the command (vendor specific)	22						
Mitigation Manager notifies Acting of the status of the execution of the OpenC2 command (e.g., completed, failed)	23	RESPONSE				type = status, value = complete, command = DENY reference	

Figure 5-1. Scenario Diagram: Mitigate Evil Domain (Cont.)

APPENDIX A EXAMPLE OPENC2 COMMANDS

A.1 SCAN

Table A-1. Example Actions: SCAN

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Scan a device for vulnerabilities	SCAN	cybox:Device	network.sensor	search = CVE
			cybox:DeviceObjectType	(optional)	
2	Scan email messages for malware	SCAN	cybox:Email_Message	network.sensor	search = malware signature
			cybox:EmailMessageObjectType	(optional)	
3	Scan network traffic for malicious activities	SCAN	cybox:Network_Connection	network.sensor	search = network signature
			cybox:NetworkConnectionObjectType	(optional)	
4	Scan a disk for vulnerabilities	SCAN	cybox:Disk	network.sensor	search
			cybox:DiskObjectType	(optional)	
5	Scan a disk partition for malware	SCAN	cybox:Disk_Partition	network.sensor	search
			cybox:DiskPartitionObjectType	(optional)	
6	Scan a domain for malicious activities	SCAN	cybox:Domain_Name	network.sensor	search
			cybox:DomainNameObjectType	(optional)	
7	Scan files for malware	SCAN	cybox:File	network.sensor	search
			cybox:FileObjectType	(optional)	
8	Scan memory for malicious activities	SCAN	cybox:Memory	network.sensor	search
			cybox:MemoryObjectType	(optional)	
9	Scan network packets for malicious activities	SCAN	cybox:Network_Packet	network.sensor	search
			cybox:NetworkPacketObjectType	(optional)	
10	Scan a subnet for vulnerabilities or malicious activities	SCAN	cybox:Network_Subnet	network.sensor	search
			cybox:NetworkSubnetObjectType	(optional)	
11	Scan a process for malicious activities	SCAN	cybox:Process	network.sensor	search
			cybox:ProcessObjectType	(optional)	
12	Scan a software product for vulnerabilities or malicious activities	SCAN	cybox:Product	network.sensor	search
			cybox:ProductObjectType	(optional)	
13	Scan a system for vulnerabilities or malicious activities	SCAN	cybox:System	network.sensor	search
			cybox:SystemObjectType	(optional)	
14	Scan a URL for malicious activities	SCAN	cybox:URI	network.sensor	search
			cybox:URIObjectType	(optional)	
15	Scan a user for malicious activities	SCAN	cybox:User_Account	network.sensor	search
			cybox:UserAccountObjectType	(optional)	
16	Scan a user session for vulnerabilities or malicious activities	SCAN	cybox:User_Session	network.sensor	search
			cybox:UserSessionObjectType	(optional)	
17	Scan a volume for malware	SCAN	cybox:Volume	network.sensor	search
			cybox:VolumeObjectType	(optional)	

A.2 LOCATE

Table A-2. Example Actions: LOCATE

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Geolocate a device	LOCATE	cybox:Device	process.location-service	
			cybox:DeviceObjectType	(optional)	
2	Get location of an IP address	LOCATE	cybox:Address	process.location-service	
			cybox:AddressObjectType	(optional)	
3	Get location of a user	LOCATE	cybox:User_Account	process.location-service	
			cybox:UserAccountObjectType	(optional)	
4	Get a logical location of a file	LOCATE	cybox:File	process.location-service	
			cybox:FileObjectType	(optional)	

A.3 QUERY

Table A-3. Example Actions: QUERY

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	List all network connections	QUERY	openc2:Data	network.router	response
			openc2:DataObjectType	(optional)	
2	List running processes on a machine	QUERY	openc2:Data	endpoint	response
			openc2:DataObjectType	(optional)	
3	Request an Actuator's supported OpenC2 capabilities.	QUERY	openc2:OpenC2	network.firewall	response
			openc2:OpenC2ObjectType	(optional)	
4	Get attributes of a user	QUERY	openc2:Data	process.directory-service	response
			openc2:DataObjectType	(optional)	
5	List all alerts configured on the device	QUERY	openc2:Data	endpoint	response
			openc2:DataObjectType	(optional)	
6	List all endpoint applications/sensors configured on the device	QUERY	openc2:Data	endpoint	response
			openc2:DataObjectType	(optional)	
7	Get current running configuration of the device	QUERY	openc2:Data	endpoint	response
			openc2:DataObjectType	(optional)	

A.4 REPORT

Table A-4. Example Actions: REPORT

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Produce and send a report	REPORT	openc2:Data		report-to
			openc2:DataObjectType		

A.5 GET

Table A-5. Example Actions: GET

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Acting gets the potentially malicious email, including URLs and attachments.	GET	cybox:Email_Message	network.sense_making	
			cybox:EmailMessageObjectType		
2	Get process file	GET	cybox:File	endpoint.workstation	
			cybox:FileObjectType		
3	Get process dump	GET	cybox:Memory		
			cybox:MemoryObjectType		
4	Get configuration file	GET	cybox:File		
			cybox:FileObjectType		

A.6 NOTIFY

Table A-6. Example Actions: NOTIFY

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Notify security officer to report compliance with change of configuration	NOTIFY	cybox:User_Account	process.email-service	message
			cybox:UserAccountObjectType	(optional)	
2	Send a command to notify an external enclave.	NOTIFY	cybox:System		message = acknowledge
			cybox:SystemObjectType		
3	Send a command to notify an authorized user to request approval.	NOTIFY	cybox:User_Account	endpoint.server	message
			cybox:UserAccountObjectType		

A.7 DENY

Table A-7. Example Actions: DENY

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Block traffic to/from specific IP address; suitable for coordinating across multiple enclaves and allowing enclaves to determine most appropriate response	DENY	cybox:Network_Connection cybox:NetworkConnectionObjectType		
2	Block traffic to/from specific IP address at all network firewalls	DENY	cybox:Network_Connection cybox:NetworkConnectionObjectType	network.firewall (optional)	
3	Block traffic at the network routers	DENY	cybox:Network_Connection cybox:NetworkConnectionObjectType	network.router (optional)	
4	Block network traffic inside the enclave	DENY	cybox:Network_Connection cybox:NetworkConnectionObjectType		where = internal
5	Block network traffic at the perimeter	DENY	cybox:Network_Connection cybox:NetworkConnectionObjectType		where = perimeter
6	Block network traffic by ACL	DENY	cybox:Network_Connection cybox:NetworkConnectionObjectType	network.router (optional)	method = acl
7	Block access to a bad external IP address by null routing at the network routers.	DENY	cybox:Network_Connection cybox:NetworkConnectionObjectType (external IP address)	network.router (optional)	method = blackhole
8	Block access to/from suspicious internal IP address by null routing at the network routers	DENY	cybox:Network_Connection cybox:NetworkConnectionObjectType (internal IP address)	network.router (optional)	method = blackhole
9	Block network traffic at the perimeter routers	DENY	cybox:Network_Connection cybox:NetworkConnectionObjectType	network.router (specify perimeter routers)	
10	Block access to suspicious external IP address by redirecting external DNS queries to an internal DNS server	DENY	cybox:Network_Connection cybox:NetworkConnectionObjectType		method = sinkhole
11	Block traffic to/from specific IP address at all endpoints' firewalls	DENY	cybox:Network_Connection cybox:NetworkConnectionObjectType	process (specify endpoint and firewall application)	
12	Block malicious URL (blacklist domain); suitable for coordinating across multiple enclaves and allowing enclaves to determine most appropriate response	DENY	cybox:URI cybox:URIObjectType		

22 **Table A-7. Example Actions: DENY (Cont.)**

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
13	Block malicious URL at proxy server	DENY	cybox:URI	network.proxy	
			cybox:URIObjectType	(optional)	
14	Block malicious URL at all network firewalls	DENY	cybox:URI	network.firewall	
			cybox:URIObjectType	(optional)	
15	Block malicious URL at all endpoint firewalls	DENY	cybox:URI	process	
			cybox:URIObjectType	(specify endpoint and firewall application)	
16	Block malicious URL at all endpoint browsers	DENY	cybox:URI	process	
			cybox:URIObjectType	(optional)	
17	Block system application; suitable for coordinating across multiple enclaves and allowing enclaves to determine most appropriate response	DENY	cybox:Product		
			cybox:ProductObjectType		
18	Block system application from executing at endpoint with certain characteristics or specific endpoint(s)	DENY	cybox:Product	endpoint	
			cybox:ProductObjectType	(specify based on endpoint characteristics)	
19	Block system application from executing by application white listing	DENY	cybox:Product	endpoint	method = whitelist
			cybox:ProductObjectType	(optional)	
20	Deny Device Access (Infected Host)	DENY	cybox:Device	process.aaa-server	
			cybox:DeviceObjectType	(optional)	
21	Block Process	DENY	cybox:Process	endpoint	
			cybox:ProcessObjectType	(optional)	
22	Block Process by Domain	DENY	cybox:Process	endpoint	
			cybox:ProcessObjectType (specify process by domain)	(optional)	
23	Deny user access to the system; suitable for coordinating across multiple enclaves	DENY	cybox:User_Account		
			cybox:UserAccountObjectType		

23

A.8 CONTAIN

Table A-8. Example Actions: CONTAIN

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Quarantine a file, general	CONTAIN	cybox:File		
			cybox:FileObjectType		
2	Quarantine a file	CONTAIN	cybox:File	endpoint	where
			cybox:FileObjectType	(optional)	
3	Contain a user or group, general	CONTAIN	cybox:User_Account		
			cybox:UserAccountType		
4	Contain network traffic to a honeynet, general	CONTAIN	cybox:Network_Connection		
			cybox:NetworkConnectionObjectType		
5	Isolate a process, general	CONTAIN	cybox:Process		
			cybox:ProcessObjectType		
6	Isolate a process	CONTAIN	cybox:Process	endpoint	where
			cybox:ProcessObjectType	(optional)	
7	Quarantine a device, general	CONTAIN	cybox:Device		
			cybox:DeviceObjectType		
8	Quarantine a device	CONTAIN	cybox:Device	network	where (network segment, vlan)
			cybox:DeviceObjectType	(optional)	
9	Contain a user or group	CONTAIN	cybox:User_Account	network	
			cybox:UserAccountType	(optional)	
10	Contain network traffic to a honeynet	CONTAIN	cybox:Network_Connection	network	
			cybox:NetworkConnectionObjectType	(optional)	

A.9 ALLOW

Table A-9. Example Actions: 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	cybox:Network_Connection		
			cybox:NetworkConnectionObjectType		
2	Unblock traffic to/from specific IP address at all network firewalls	ALLOW	cybox:Network_Connection	network.firewall	
			cybox:NetworkConnectionObjectType	(optional)	
3	Unblock traffic at the network routers	ALLOW	cybox:Network_Connection	network.router	
			cybox:NetworkConnectionObjectType	(optional)	
4	Unblock network traffic inside the enclave	ALLOW	cybox:Network_Connection		where = internal
			cybox:NetworkConnectionObjectType		
5	Delay Machine Authentication	ALLOW	cybox:Device	process.aaa-server	delay = <duration>
			cybox:DeviceObjectType	(optional)	
6	Unquarantine a file	ALLOW	cybox:File	endpoint	
			cybox:FileObjectType	(optional)	
7	Unblock network traffic at the perimeter	ALLOW	cybox:Network_Connection		where = perimeter
			cybox:NetworkConnectionObjectType		
8	Unblock network traffic at the perimeter routers	ALLOW	cybox:Network_Connection	network.router	
			cybox:NetworkConnectionObjectType	(specify perimeter routers)	
9	Unblock traffic to/from specific IP address at all endpoints' firewalls	ALLOW	cybox:Network_Connection	process	
			cybox:NetworkConnectionObjectType	(specify endpoint and firewall application)	
10	Unblock URL (blacklist domain); suitable for coordinating across multiple enclaves and allowing enclaves to determine most appropriate response	ALLOW	cybox:URI		
			cybox:URIObjectType		
11	Unblock URL at proxy server	ALLOW	cybox:URI	network.proxy	
			cybox:URIObjectType	(optional)	
12	Unblock URL at all network firewalls	ALLOW	cybox:URI	network.firewall	
			cybox:URIObjectType	(optional)	
13	Unblock URL at all endpoint firewalls	ALLOW	cybox:URI	process	
			cybox:URIObjectType	(specify endpoint and firewall application)	
14	Unblock URL at all endpoint browsers	ALLOW	cybox:URI	process	
			cybox:URIObjectType	(optional)	

Table A-9. Example Actions: ALLOW (Cont.)

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
15	Unblock system application; suitable for coordinating across multiple enclaves and allowing enclaves to determine most appropriate response	ALLOW	cybox:Product cybox:ProductObjectType		
16	Unblock system application from executing at endpoint with certain characteristics or specific endpoint(s)	ALLOW	cybox:Product cybox:ProductObjectType	endpoint (specify based on endpoint characteristics)	
17	Authenticate Machine	ALLOW	cybox:Device cybox:DeviceObjectType	process.aaa-server (optional)	
18	Unblock Process	ALLOW	cybox:Process cybox:ProcessObjectType	endpoint (optional)	
19	Unblock Process by Domain	ALLOW	cybox:Process cybox:ProcessObjectType (specify process by domain)	endpoint (optional)	
20	Authenticate user; suitable for coordinating across multiple enclaves	ALLOW	cybox:User_Account cybox:UserAccountObjectType		
21	Delay user authentication; suitable for coordinating across multiple enclaves	ALLOW	cybox:User_Account cybox:UserAccountObjectType		delay = <duration>
22	Grant User Access to Specific System	ALLOW	cybox:User_Account cybox:UserAccountObjectType	process.aaa-server (optional)	permissions
23	Unquarantine a file, general	ALLOW	cybox:File cybox:FileObjectType		
24	Release a process from isolation, general	ALLOW	cybox:Process cybox:ProcessObjectType		
25	Release a process from isolation	ALLOW	cybox:Process cybox:ProcessObjectType	endpoint (optional)	
26	Unquarantine a device, general	ALLOW	cybox:Device cybox:DeviceObjectType		
27	Unquarantine a device	ALLOW	cybox:Device cybox:DeviceObjectType	network (optional)	

A.10 START

Table A-10. Example Actions: START

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Start Process, general	START	cybox:Process		
			cybox:ProcessObjectType		
2	Start Process	START	cybox:Process	endpoint	
			cybox:ProcessObjectType	(optional)	
3	Start Process with Delay	START	cybox:Process	endpoint	delay
			cybox:ProcessObjectType	(optional)	
4	Spawn Process	START	cybox:Process	endpoint	method = spawn
			cybox:ProcessObjectType	(optional)	
5	Execute Command	START	cybox:Process	endpoint	
			cybox:ProcessObjectType	(optional)	
6	Start an Application	START	cybox:Product	endpoint	
			cybox:ProductObjectType	(optional)	
7	Start a device	START	cybox:System	network	
			cybox:SystemObjectType	(optional)	
8	Start a virtual machine	START	cybox:System	process.virtualization-service	
			cybox:SystemObjectType	(optional)	
9	Activates the system partitions of a machine	START	cybox:Disk_Partition	endpoint	
			cybox:DiskPartitionObjectType	(optional)	

A.11 STOP

Table A-11. Example Actions: STOP

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Shutdown a system	STOP	cybox:Device	endpoint	method = graceful
			cybox:DeviceObjectType	(optional)	
2	Shutdown a system, immediate	STOP	cybox:Device	endpoint	method = immediate
			cybox:DeviceObjectType	(optional)	
3	Logoff User: Logoff all the sessions of a particular user from the machine	STOP	cybox:User_Account	endpoint	method = graceful
			cybox:UserAccountObjectType	(optional)	
4	Stop a vm	STOP	cybox:System	process.virtualization-service	method = graceful
			cybox:SystemObjectType	(optional)	
5	Terminate a process, general	STOP	cybox:Process		
			cybox:ProcessObjectType		
6	Terminate a process	STOP	cybox:Process	endpoint	
			cybox:ProcessObjectType	(optional)	
7	Stop service	STOP	cybox:Windows_Service	endpoint	
			cybox:WindowsServiceObjectType	(optional)	
8	Terminate a session	STOP	cybox:User_Session	process.aaa-server	
			cybox:UserSessionObjectType	(optional)	
9	Shutdown a system, general	STOP	cybox:Device		
			cybox:DeviceObjectType		
10	Disable Device	STOP	cybox:Device	network	method = disable
			cybox:DeviceObjectType	(optional)	
11	Deactivate Partition: Deactivates the system partitions of a machine. Disallows booting from the specified partition	STOP	cybox:Disk_Partition	endpoint	
			cybox:DiskPartitionObjectType	(optional)	
12	Logoff User: Logoff all the sessions of a particular user, general	STOP	cybox:User_Account		
			cybox:UserAccountObjectType		
13	Logoff User: Logoff all the sessions of a particular user from the machine, immediate	STOP	cybox:User_Account	endpoint	method = immediate
			cybox:UserAccountObjectType	(optional)	
14	Stop a vm, immediate	STOP	cybox:System	process.virtualization-service	method = immediate
			cybox:SystemObjectType	(optional)	

A.12 RESTART

Table A-12. Example Actions: RESTART

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Restart device (system)	RESTART	cybox:System		
			cybox:SystemObjectType		
2	Restart device (system) with different OS	RESTART	cybox:System		options, e.g., OS
			cybox:SystemObjectType		
3	Restart VM	RESTART	cybox:System	process.virtualization-service	
			cybox:SystemObjectType	(optional)	
4	Restart process	RESTART	cybox:Process	endpoint	
			cybox:ProcessObjectType	(optional)	

A.13 PAUSE

Table A-13. Example Actions: PAUSE

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Pause device (system)	PAUSE	cybox:System		method = sleep
			cybox:SystemObjectType		
2	Hibernate device (system)	PAUSE	cybox:System		method = hibernate
			cybox:SystemObjectType		
3	Pause VM	PAUSE	cybox:System	process.virtualization-service	
			cybox:SystemObjectType	(optional)	
4	Pause a system or VM for a specified duration	PAUSE	cybox:System		duration = <duration>
			cybox:SystemObjectType		
5	Pause process	PAUSE	cybox:Process	endpoint	
			cybox:ProcessObjectType	(optional)	
6	Pause a process for a specified duration	PAUSE	cybox:Process	endpoint	duration = <duration>
			cybox:ProcessObjectType	(optional)	

A.14 RESUME

Table A-14. Example Actions: RESUME

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Resume device (system)	RESUME	cybox:System		
			cybox:SystemObjectType		
2	Resume VM	RESUME	cybox:System	process.virtualization-service	
			cybox:SystemObjectType	(optional)	
3	Resume process	RESUME	cybox:Process	endpoint	
			cybox:ProcessObjectType	(optional)	

A.15 CANCEL

Table A-15. Example Actions: CANCEL

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Cancel a previously issued command	CANCEL	openc2:Command		command-ref = command reference
			openc2:CommandObjectType		
2	Cancel a previously issued command, directed to a specific actuator (endpoint)	CANCEL	openc2:Command	endpoint	command-ref = command reference
			openc2:CommandObjectType	(optional)	
3	Cancel a previously issued command, directed to a specific actuator (network)	CANCEL	openc2:Command	network	command-ref = command reference
			openc2:CommandObjectType	(optional)	
4	Cancel a previously issued command, directed to a specific actuator (process)	CANCEL	openc2:Command	process	command-ref = command reference
			openc2:CommandObjectType	(optional)	

A.16 SET

Table A-16. Example Actions: SET

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Set registry key value	SET	cybox:Windows_Registry_Key	endpoint.workstation	set-to
			cybox:WindowsRegistryKeyObjectType	(optional)	
2	Set file permissions	SET	cybox:File	process.directory-service	set-to
			cybox:FileObjectType	(optional)	
3	Set user rights	SET	cybox:User_Account	process.directory-service	set-to
			cybox:UserAccountObjectType	(optional)	
4	Set password policy	SET	cybox:System	process.directory-service	set-to
			cybox:SystemObjectType	(optional)	
5	Set auditing policy	SET	cybox:System		set-to
			cybox:SystemObjectType		
6	Set registry permissions	SET	cybox:Windows_Registry_Key	endpoint.workstation	set-to
			cybox:WindowsRegistryKeyObjectType	(optional)	
7	Set service permissions	SET	cybox:Process		set-to
			cybox:ProcessObjectType		
8	Set group policy (computer, user)	SET	cybox:System	endpoint.workstation	set-to
			cybox:SystemObjectType	(optional)	
9	Set user settings (remediate per user instead of per computer)	SET	cybox:User_Account	process.directory-service	set-to
			cybox:UserAccountObjectType	(optional)	
10	Change a specific value in a config file	SET	openc2:Data		set-to
			openc2:DataObjectType	(optional)	
11	Change firewall rule	SET	openc2:Data	network.firewall	set-to
			openc2:DataObjectType	(optional)	
12	Change HIPS rule	SET	openc2:Data	network.hips	set-to
			openc2:DataObjectType	(optional)	
13	Change network device rule	SET	openc2:Data	network.router	set-to
			openc2:DataObjectType	(optional)	
14	Acting quarantines the infected Host by commanding Directory Services to set the Host's security group. (No return requested.)	SET	cybox:System	process.directory-service	set-to
			cybox:SystemObjectType	(optional)	
15	Acting commands Directory Services to return the Host to the active group. (No return requested.)	SET	cybox:System	process.directory-service	set-to
			cybox:SystemObjectType	(optional)	

Table A-16. Example Actions: SET (Cont.)

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
16	[Alternative] Mitigation Manager sends an OpenC2 command containing the configuration update (signatures)	SET	openc2:Data	network.sensor	set-to
			openc2:DataObjectType	(optional)	
17	Change system ou	SET	openc2:Data		set-to
			openc2:DataObjectType		
18	Set system attribute	SET	openc2:Data		set-to
			openc2:DataObjectType		
19	Set/reset password	SET	cybox:User_Account		set-to
			cybox:UserAccountObjectType		
20	Change machine settings	SET	openc2:Data		set-to
			openc2:DataObjectType		
21	Change desktop settings	SET	openc2:Data		set-to
			openc2:DataObjectType		
22	Change device IP	SET	openc2:Data		set-to
			openc2:DataObjectType		
23	Change device MAC	SET	openc2:Data		set-to
			openc2:DataObjectType		
24	Change sensor sample rate	SET	openc2:Data		set-to
			openc2:DataObjectType		
25	Limit connections to process	SET	openc2:Data		set-to
			openc2:DataObjectType		

A.17 UPDATE

Table A-17. Example Actions: UPDATE

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Install software	UPDATE	cybox:Product	endpoint	source
			cybox:ProductObjectType	(optional)	
2	Install patch	UPDATE	cybox:Product	endpoint	source
			cybox:ProductObjectType	(optional)	
3	Update signature file (anti-virus)	UPDATE	cybox:Product	process.anti-virus-scanner	source
			cybox:ProductObjectType	(optional)	
4	Update sensor's signatures	UPDATE	cybox:Product	network.sensor	source
			cybox:ProductObjectType	(optional)	
5	Load Machine Settings	UPDATE	cybox:Device	endpoint	source
			cybox:DeviceObjectType	(optional)	
6	Synchronize Machine	UPDATE	cybox:Device	endpoint	source
			cybox:DeviceObjectType	(optional)	
7	Update Registry	UPDATE	cybox:Device	endpoint	source
			cybox:DeviceObjectType	(optional)	
8	Load File(s)	UPDATE	cybox:Device	endpoint	source
			cybox:DeviceObjectType	(optional)	

A.18 MOVE

Table A-18. Example Actions: MOVE

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Move file/directory	MOVE	cybox:File		move-to
			cybox:FileObjectType		
2	Fork: Copy and redirect data to more than one destination	MOVE	openc2:Data		move-to
			openc2:DataObjectType		

A.19 REDIRECT

Table A-19. Example Actions: 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	cybox:Network_Connection		where
			cybox:NetworkConnectionObjectType		
2	Redirect traffic to a honeypot at a specific router	REDIRECT	cybox:Network_Connection	network.router	where
			cybox:NetworkConnectionObjectType		
3	Cancel traffic redirection; suitable for coordinating across multiple enclaves and allowing enclaves to determine most appropriate response	REDIRECT	cybox:Network_Connection		where = null
			cybox:NetworkConnectionObjectType		
4	Cancel traffic redirection at a specific router	REDIRECT	cybox:Network_Connection	network.router	where = null
			cybox:NetworkConnectionObjectType		
5	In order to investigate a suspicious user/endpoint, an investigator would want to issue a 'redirect' command so that the endpoint's traffic is redirected to an intrusion detection system where alerts will be fired as signatures are matched	REDIRECT	cybox:Network_Connection		
			cybox:NetworkConnectionObjectType		
6	In order to enable self-remediation of a user's endpoint, the investigator would want to redirect all URLs to a quarantine portal so that remediation services can be accessed (URL redirection for self-service remediation)	REDIRECT	cybox:URI	network.router	where
			cybox:URIObjectType		

A.20 DELETE

Table A-20. Example Actions: DELETE

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Delete file, inter-enclave	DELETE	cybox:File		where
			cybox:FileObjectType		
2	Delete file, within an enclave	DELETE	cybox:File	endpoint	
			cybox:FileObjectType	(optional)	
3	Delete email, inter-enclave	DELETE	cybox:Email_Message		
			cybox:EmailMessageObjectType		
4	Delete email from exchange server	DELETE	cybox:Email_Message	process.email-service	
			cybox:EmailMessageObjectType	(optional)	
5	Delete firewall rule	DELETE	openc2:Data	network.firewall	
			openc2:DataObjectType	(optional)	
6	Delete srp	DELETE	openc2:Data		
			openc2:DataObjectType		

A.21 SNAPSHOT

Table A-21. Example Actions: SNAPSHOT

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Take a snapshot of a VM	SNAPSHOT	cybox:System	process.virtualization-service	
			cybox:SystemObjectType	(optional)	

A.22 DETONATE

Table A-22. Example Actions: DETONATE

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Acting sends the URL to be analyzed in a sandbox.	DETONATE	cybox:URI	process.sandbox	
			cybox:URIObjectType	(optional)	
2	Acting sends the file to the Sandbox for detonation analysis.	DETONATE	cybox:File	process.sandbox	
			cybox:FileObjectType	(optional)	
3	Acting sends the attachments to be analyzed in a sandbox.	DETONATE	cybox:File	process.sandbox	
			cybox:FileObjectType	(optional)	

A.23 RESTORE

Table A-23. Example Actions: RESTORE

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Restore a device to a known restore point.	RESTORE	cybox:Device	process.remediation-service	restore-point
			cybox:DeviceObjectType	(optional)	

A.24 SAVE

Table A-24. Example Actions: SAVE

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Save data	SAVE	cybox:File	endpoint	save-to
			cybox:FileObjectType	(optional)	
2	Save an email message	SAVE	cybox:Email_Message	process.email-service	save-to
			cybox:EmailMessageObjectType	(optional)	
3	Save a raw network packet	SAVE	cybox:Network_Packet	network.router	save-to
			cybox:NetworkPacketObjectType	(optional)	

A.25 MODIFY

Table A-25. Example Actions: MODIFY

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Modify a file	MODIFY	cybox:File		modify-to
			cybox:FileObjectType		
2	Modify a device's configuration	MODIFY	cybox:Device	endpoint	modify-to
			cybox:DeviceObjectType	(optional)	
3	Modify user account privileges	MODIFY	cybox:User_Account	process.directory-service	modify-to
			cybox:UserAccountObjectType	(optional)	
4	Modify data within a process	MODIFY	cybox:Process		modify-to
			cybox:ProcessObjectType		
5	Modify data within a software product	MODIFY	cybox:Product	endpoint	modify-to
			cybox:ProductObjectType	(optional)	
6	Modify a system's configuration	MODIFY	cybox:System		modify-to
			cybox:SystemObjectType		

A.26 THROTTLE

Table A-26. Example Actions: THROTTLE

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Limit bandwidth	THROTTLE	cybox:Network_Connection	network.router	
			cybox:NetworkConnectionObjectType	(optional)	

A.27 DELAY

Table A-27. Example Actions: DELAY

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Delay all traffic	DELAY	cybox:Network_Connection		delay
			cybox:NetworkConnectionObjectType		

A.28 SUBSTITUTE

Table A-28. Example Actions: SUBSTITUTE

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Overwrite data	SUBSTITUTE	cybox:File	endpoint	options
			cybox:FileObjectType	(optional)	
2	Substitute traffic	SUBSTITUTE	cybox:Network_Connection	network.router	options
			cybox:NetworkConnectionObjectType	(optional)	

A.29 COPY

Table A-29. Example Actions: COPY

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Copy a file	COPY	cybox:File		where, copy-to
			cybox:FileObjectType		
2	Copy network traffic	COPY	cybox:Network_Connection		copy-to
			cybox:NetworkConnectionObjectType		
3	Copy netflow information related to particular ip address	COPY	cybox:Network_Flow		copy-to
			cybox:NetworkFlowObjectType		
4	Copy the full contents of a disk partition	COPY	cybox:Disk_Partition		where, copy-to
			cybox:DiskPartitionObjectType		
5	Copy the full contents of a system's memory	COPY	cybox:Memory		where, copy-to
			cybox:MemoryObjectType		

A.30 SYNC

Table A-30. Example Actions: SYNC

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Synchronize an endpoint sensor or actuator to another device	SYNC	cybox:Device	endpoint	
			cybox:DeviceObjectType	(optional)	

A.31 DISTILL

Table A-31. Example Actions: DISTILL

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Filter	DISTILL	cybox:Network_Connection	network.sensor	
			cybox:NetworkConnectionObjectType		
2	Reduce	DISTILL	cybox:Network_Connection	network.sensor	
			cybox:NetworkConnectionObjectType		
3	Flatten	DISTILL	cybox:Network_Connection	network.sensor	
			cybox:NetworkConnectionObjectType		
4	Specify Block of IP addresses to capture sensing data from	DISTILL	cybox:Network_Connection	network.sensor	
			cybox:NetworkConnectionObjectType		

A.32 AUGMENT

Table A-32. Example Actions: AUGMENT

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Preprocess network traffic, inter-enclave	AUGMENT	cybox:Network_Connection		method
			cybox:NetworkConnectionObjectType		
2	Preprocess network traffic, within an enclave	AUGMENT	cybox:Network_Connection	network.sensor	method
			cybox:NetworkConnectionObjectType	(optional)	

A.33 INVESTIGATE

Table A-33. Example Actions: INVESTIGATE

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Investigate the specified IP address for malicious activities	INVESTIGATE	cybox:Address cybox:AddressObjectType		[report-to]
2	Investigate the specified device	INVESTIGATE	cybox:Device cybox:DeviceObjectType		[report-to]
3	Investigate the specified domain	INVESTIGATE	cybox:Domain_Name cybox:DomainNameObjectType		[report-to]
4	Investigate the specified email message	INVESTIGATE	cybox:Email_Message cybox:EmailMessageObjectType		[report-to]
5	Investigate the specified file(s)	INVESTIGATE	cybox:File cybox:FileObjectType		[report-to]
6	Investigate the specified hostname	INVESTIGATE	cybox:Hostname cybox:HostnameObjectType		[report-to]
7	Investigate the specified network traffic	INVESTIGATE	cybox:Network_Connection cybox:NetworkConnectionObjectType		[report-to]
8	Investigate the specified port for malicious activities	INVESTIGATE	cybox:Port cybox:PortObjectType		[report-to]
9	Investigate the specified process	INVESTIGATE	cybox:Process cybox:ProcessObjectType		[report-to]
10	Investigate the specified software product	INVESTIGATE	cybox:Product cybox:ProductObjectType		[report-to]
11	Investigate the specified system	INVESTIGATE	cybox:System cybox:SystemObjectType		[report-to]
12	Investigate the specified certificate	INVESTIGATE	cybox:X509_Certificate cybox:X509CertificateObjectType		[report-to]

105 ## A.34 MITIGATE

106 **Table A-34. Example Actions: MITIGATE**

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Mitigate the specified malicious IP address	MITIGATE	cybox:Address cybox:AddressObjectType		[report-to]
2	Mitigate the specified infected device	MITIGATE	cybox:Device cybox:DeviceObjectType		[report-to]
3	Mitigate the specified malicious email message	MITIGATE	cybox:Email_Message cybox:EmailMessageObjectType		[report-to]
4	Mitigate the specified malicious file(s)	MITIGATE	cybox:File cybox:FileObjectType		[report-to]
5	Mitigate the specified infected hostname	MITIGATE	cybox:Hostname cybox:HostnameObjectType		[report-to]
6	Mitigate the specified malicious network traffic	MITIGATE	cybox:Network_Connection cybox:NetworkConnectionObjectType		[report-to]
7	Mitigate the specified malicious process	MITIGATE	cybox:Process cybox:ProcessObjectType		[report-to]
8	Mitigate the specified malicious software product	MITIGATE	cybox:Product cybox:ProductObjectType		[report-to]
9	Mitigate the specified infected system	MITIGATE	cybox:System cybox:SystemObjectType		[report-to]
10	Mitigate the specified compromised certificate	MITIGATE	cybox:X509_Certificate cybox:X509CertificateObjectType		[report-to]

107

A.35 REMEDIATE

Table A-35. Example Actions: REMEDIATE

	Description	Action	Target	Actuator	Modifier
			Target-Specifier	Actuator-Specifier	
1	Remediate the specified malicious email message	REMEDIATE	cybox:Email_Message cybox:EmailMessageObjectType		[report-to]
2	Remediate the specified infected hostname	REMEDIATE	cybox:Hostname cybox:HostnameObjectType		[report-to]
3	Remediate the specified malicious IP address	REMEDIATE	cybox:Address cybox:AddressObjectType		[report-to]
4	Remediate the specified infected device	REMEDIATE	cybox:Device cybox:DeviceObjectType		[report-to]
5	Remediate the specified malicious file(s)	REMEDIATE	cybox:File cybox:FileObjectType		[report-to]
6	Remediate the specified malicious network traffic	REMEDIATE	cybox:Network_Connection cybox:NetworkConnectionObjectType		[report-to]
7	Remediate the specified malicious process	REMEDIATE	cybox:Process cybox:ProcessObjectType		[report-to]
8	Remediate the specified malicious software product	REMEDIATE	cybox:Product cybox:ProductObjectType		[report-to]
9	Remediate the specified infected system	REMEDIATE	cybox:System cybox:SystemObjectType		[report-to]
10	Remediate the specified compromised certificate	REMEDIATE	cybox:X509_Certificate cybox:X509CertificateObjectType		[report-to]

A.36 RESPONSE

Table A-36. Example Actions: RESPONSE

	Description	Action	Modifier
1	Acknowledge the receipt of an action	RESPONSE	type = acknowledge, command-ref = command reference
2	Signal completion of an action	RESPONSE	type = status, value = complete, command-ref = command reference
3	Provide the status of an action	RESPONSE	type = status, value = current, command-ref = command reference

A.37 ALERT

Table A-37. Example Actions: ALERT

	Description	Action	Modifier
1	An actuator sends an alert as the result of some condition.	ALERT	type, value
2	A sensor sends an alert as the result of some condition.	ALERT	type, value

118 This page intentionally left blank