OpenC2 Extension for CACAO Version 1.0

Committee Specification Draft 01

7 August 2024

This stage:

https://docs.oasis-open.org/openc2/openc2-cacao-ext/v1.0/csd01/openc2-cacao-ext-v1.0-csd01.md
(Authoritative)
https://docs.oasis-open.org/openc2/openc2-cacao-ext/v1.0/csd01/openc2-cacao-ext-v1.0-csd01.html
https://docs.oasis-open.org/openc2/openc2-cacao-ext/v1.0/csd01/openc2-cacao-ext-v1.0-csd01.pdf

Previous stage:

N/A

Latest stage:

https://docs.oasis-open.org/openc2/openc2-cacao-ext/v1.0/openc2-cacao-ext-v1.0.md (Authoritative)
https://docs.oasis-open.org/openc2/openc2-cacao-ext/v1.0/openc2-cacao-ext-v1.0.html
https://docs.oasis-open.org/openc2/openc2-cacao-ext/v1.0/openc2-cacao-ext-v1.0.pdf

Technical Committee:

OASIS Open Command and Control (OpenC2) TC

Chairs:

Duncan Sparrell (duncan@sfractal.com), sFractal Consulting LLC
Michael Rosa (mjrosa@nsa.gov), National Security Agency

Editor:

David Lemire (david.lemire@hii-tsd.com), National Security Agency

Additional artifacts:

This prose specification is one component of a Work Product that also includes:

- JADN schema: `schemas/oc2-cacao-ext.jadn`

Related work:

This specification is related to:

- *Open Command and Control (OpenC2) Language Specification Version 2.0*. Edited by Toby Considine and Duncan Sparrell. Latest stage: [https://docs.oasis-open.org/openc2/oc2ls/v2.0/oc2ls-v2.0.html
- *CACAO-Security-Playbooks-v2.0*. Edited by Bret Jordan and Allan Thomson. Latest stage:https://docs.oasis-open.org/cacao/security-playbooks/v2.0/security-playbooks-v2.0.html

Abstract:

Collaborative Automated Course of Action Operations (CACAO) is a schema and taxonomy for cyber security playbooks. The CACAO specification describes how these playbooks can be created, documented, and shared in a structured and standardized way across organizational boundaries and technological solutions. This extension builds on existing CACAO v2.0 OpenC2 features to improve modularity and utilize the current OpenC2 Transfer Specifications for MQTT (v1.0) and HTTPS (v1.1).

Status:

Key words:

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] and [RFC8174] when, and only when, they appear in all capitals, as shown here.

Citation format:

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

[OpenC2-CACAO-ext-v1.0]

CACAO OpenC2 Extension Version 1.0. Edited by David Lemire. 07 August 2024. OASIS Committee Specification Draft 01. https://docs.oasis-open.org/openc2/openc2-cacao-ext/v1.0/csd01/openc2-cacao-ext-v1.0-csd01.html
Latest stage: https://docs.oasis-open.org/openc2/openc2-cacao-ext/v1.0/openc2-cacao-ext-v1.0.html

Notices

Table of Contents

1 Introduction

This section is non-normative.

Collaborative Automated Course of Action Operations (CACAO) is a schema and taxonomy for cyber security playbooks. The [CACAO Security Playbooks] specification describes how these playbooks can be created, documented, and shared in a structured and standardized way across organizational boundaries and technological solutions.

OpenC2 is a suite of specifications that enables command and control of cyber defense systems and components. OpenC2 typically uses a request-response paradigm where a Command is encoded by a Producer (managing application) and transferred to a Consumer (managed device or virtualized function) using a secure transfer protocol, and the Consumer can respond with status and any requested information.

This extension builds on existing CACAO v2.0 OpenC2 features to improve playbook modularity when using OpenC2 capabilities, and to utilize the current OpenC2 Transfer Specifications for [MQTT] and [HTTPS].

1.1 Glossary

1.1.1 Definitions of terms

The [OpenC2 Architecture] and [CACAO Security
Playbooks] specifications should be consulted
for the authoritative definition of terms used in this specification. A brief
overview of relevant concepts and associated terminology from those
specification is provided in Section 2.1.

1.1.2 Acronyms and abbreviations

Acronym	Expansion
CACAO	Collaborative Automated Course of Action Operations
CBOR	Concise Binary Object Representation
JADN	JSON Abstract Data Notation
JSON	Javascript Object Notation
OpenC2	Open Command and Control
SLPF	Stateless Packet Filtering
UUID	Universally Unique Identifier

1.1.3 Document conventions

The following color, font and font style conventions are used in this document:

- A fixed width font is used for all type names, property names, and literals.
- Property names are in bold style: **created_at**.
- All examples in this document are expressed in JSON. They are in fixed width font, with straight quotes, black text and a light shaded background, and 4-space indentation. JSON examples in this document are representations of JSON Objects. They should not be interpreted as string literals. The ordering of object keys is insignificant. Whitespace before or after JSON structural characters in the examples are insignificant [RFC8259].
- Parts of the example may be omitted for conciseness and clarity. These omitted parts are denoted with ellipses (...).
- All CACAO identifiers (i.e., <object-type>--<UUID>) shown in the examples are notional and for illustrative purposes, they do not represent real objects.

Example:

```
{
  "type": "openc2",
  "command_b64":
"ewogICJoZWFkZXJzIjogewogICAgInJlcXVlc3RfaWQiOiAiZDFhYzA0ODktZWQ1MS00MzQ1...",
  "agent": "mqtt-broker--7125c6f6-7f78-4a3d-8a43-f20d20632305",
  "step_variables": {
    "__mqtt-topics__:value": {
      "topic-array": ["oc2/cmd/ap/pf", "oc2/cmd/ap/edr"]
    }
  }
}
```

1.2 Schema

The schema for this AP is defined using a [JSON Abstract Data Notation
(JADN)] information model. The scope of the schema includes those

aspects of CACAO relevant to the extension described herein but does not constitute a complete CACAO IM.

2 Key Concepts & Vocabularies

2.1 Key Concepts

This section is non-normative.

The key concepts from CACAO and OpenC2 listed below are applicable to this specification. For additional information consult the [CACAO v2.0] and the [OpenC2 Architecture] Specifications, respectively.

- **CACAO Concepts**
    - **Workflow Step:** A CACAO playbook contains a with the processing logic organized in to a set of workflow steps.
    - **Action:** The type of CACAO workflow step that contains commands to be executed.
    - **Playbook-Action:** The type of CACAO workflow step that executes a separate named playbook from within the current playbook.
    - **Agents and Targets:** CACAO agents are entities that execute commands on or against CACAO targets.
- **OpenC2 Concepts**
    - **Command:** An OpenC2 action-target pair, plus other optional information, used to command an OpenC2 Consumer.
    - **Response:** An OpenC2 message sent from a Consumer to a Producer reporting on the outcome of processing a Command.
    - **Actuator Profile:** A tailored subset of the OpenC2 language plus any extensions that specifies the use of OpenC2 to command a particular function.
    - **Transfer Specification:** The description of how an existing standard transfer protocol (e.g., MQTT, HTTPS) is used to send and receive OpenC2 commands and responses.

Both OpenC2 and CACAO employ the term "target" but the meanings differ. In this extension specification, the CACAO target is used to integrate the OpenC2 Actuator Profile concept. The logical flow is as follows:

- An OpenC2 command is defined in a CACAO playbook `openc2` action step
- The `openc2` action step specifies a CACAO agent that supports the desired transfer protocol
- The `openc2` action step specifies a CACAO target that represents the AP that should process the command.

2.2 Producers and Consumers

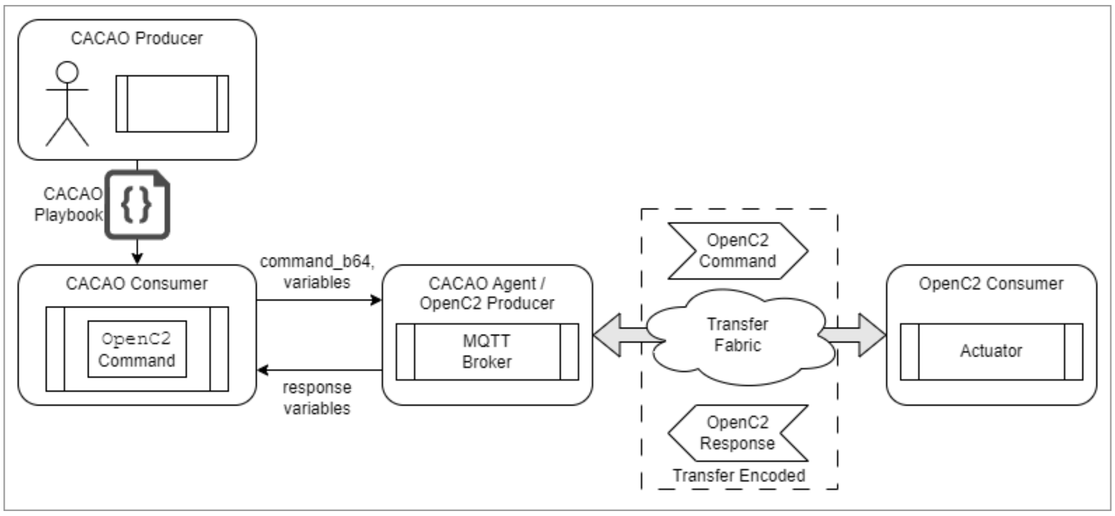
This section is non-normative.

Both OpenC2 and CACAO employ the terms "producer" and "consumer" but with different meanings. The following table identifies the relevant definitions, drawing on the CACAO v2.0 Specification and the OpenC2 Architecture Specification.

	Producer	Consumer
OpenC2	An OpenC2 Producer is a manager application that sends Commands.	An OpenC2 Consumer is a managed device/application that receives commands.
CACAO	A "CACAO 2.0 Producer" is any software that can create CACAO 2.0 content and conforms to the requirements of Section 11.1 of the CACAO Specification.	A "CACAO 2.0 Consumer" is any software that can consume CACAO 2.0 content and conforms to the requirements of Section 11.1 of the CACAO Specification.

Figure 2-1 illustrates how the concepts of producer and consumer apply when OpenC2 commands are incorporated into CACAO playbooks, illustrating an OpenC2 command invoked from a CACAO playbook action step, with the command sent and received via the MQTT protocol using a corresponding CACAO agent.

Figure 2-1: Producer and Consumer Relationships



2.3 CACAO Vocabulary Modifications

This section is non-normative.

CACAO employs the concept of vocabularies to "enhance interoperability by increasing the likelihood that different entities use the exact same string to represent the same concept". Some CACAO vocabularies are "open" (designated by `<vocabulary-type>-ov`), which means that they contain suggested values but that types that employ open vocabularies can be extended with additional values if needed. This specification proposes extensions to several open vocabularies from the CACAO specification:

- `command-type-ov` (CACAO Specification Section 5.2)
- `agent-target-type-ov` (CACAO Specification Section 7.2)
- `security-category-type-ov` (CACAO Specification Section 7.11.1)
- `variable-type-ov` (CACAO Specification Section 10.18.4)

The specific proposed extended values are:

- `command-type-ov` is extended with the type `openc2` (see Section 3.1)
    - Command type `openc2-http` is deprecated in favor of the non-transport specific `openc2` command type
- `agent-target-type-ov` "Devices and Equipment" vocabulary is extended with the following types:
    - `mqtt-broker` agent type for message transfer via MQTT (see Section 4.1.1)
    - `openc2-https` agent type for OpenC2 message transfer via HTTPS (see Section 4.1.2)
- `security-category-type-ov` is extended with the following types:
    - `openc2-consumer` (see Section 4.2)
- `variable-type-ov` is extended with the following types
    - `topic-list` to identify publish / subscribe topics to which a message should be published (see Section 5.1)

3 OpenC2 Commands In CACAO

This section describes the implementation of OpenC2 commands under CACAO, including the format and processing of an `openc2` command object, the handling of base64 encoding and decoding, and the invocation of OpenC2 via `openc2` command objects in a subordinate playbook.

This specification recommends deprecating CACAO's `openc2-http` command type in favor of the transport-neutral `openc2` command type defined here, in keeping with OpenC2's intent for the language to be defined in a transport-independent manner.

3.1 OpenC2 Command Action Step

The `openc2` command represents a command that is intended to be processed via an OpenC2 Consumer. The delivery of the command and specification of transfer mechanism and desired OpenC2 AP are handled by identifying appropriate CACAO agents and targets. The command type open vocabulary (`command-type-ov`) defined in Section 5.2 of [CACAO v2.0] is extended with the new value `openc2` :

Command Type	Description
`openc2`	An OpenC2 command to be transmitted to an OpenC2 Consumer via an OpenC2 transfer protocol.

This section defines the use of properties defined in the [CACAO v2.0] specification for an `openc2` action step.
Specifically, it addresses the content of:

- Workflow step common properties (CACAO Specification Section 4.1)
- Workflow action step properties (CACAO Specification Section 4.5)
- Workflow command object common properties (CACAO Specification Section 5.1)

The `command` and `headers` properties of the CACAO `openc2-http` command type *are not* used in the `openc2` command type defined here. The CACAO agents, targets, and variables described in Sections 4 and 5 of this specification provide mechanisms for selecting and controlling the transfer of OpenC2 command messages generated by an `openc2` command action step.

The table below defines how particular properties are addressed for an `openc2` command object. The table also identifies the level of CACAO playbook where each property is defined.

Level	Property Name	Data Type	Details
Command	**type** (required)	`string`	The value of this property **MUST** be `openc2`
Command	**command_b64** (required)	`string`	An OpenC2 command that is base64 encoded (see Section 4 of [RFC 4649]).
Workflow Action Step	**agent** (required)	`identifier`	The `agent` property of the workflow `action` type step **MUST** specify a suitable agent for OpenC2 message transfer
Workflow Common	**step_variables** (required)	`dictionary`	The common workflow `step_variables` property for an `openc2` command **MUST** specify a variable suitable for conveying OpenC2 command message destinations to the specified agent.

Usage Requirements

- When the `agent` is specified as an `mqtt-broker` (see Section 4.1.1) the `step_variables_` **MUST** include an `__mqtt-topics__` variable (see Section 5.1).

- When the `agent` is specified as an `oc2-http-api` agent (see Section 4.1.2) the `step_variables_` **MUST** include an `__http-endpoints__` variable (see Section 5.2).

> **To-Do:** Should `content_b64` be changed to `command_b64` for consistency with virtually all other CACAO command objects? Opened issue in CACAO repo; using `command_b64` for now.

Example 3.1 (OpenC2 Command, transfer via MQTT)

The IDs used in this example are notional and for illustrative purposes, they do not represent real objects.

```
{
  "type": "openc2",
  "command_b64":
"ewogICJoZWFkZXJzIjogewogICAgInJlcXVlc3RfaWQiOiAiZDFhYzA0ODktZWQ1MS00MzQ1 ...
B9CiAgfQp9",
  "agent": "mqtt-broker--7125c6f6-7f78-4a3d-8a43-f20d20632305",
  "step_variables": {
    "__mqtt-topics__:value": {
      "topic-array": ["oc2/cmd/ap/pf", "oc2/cmd/ap/edr"]
    }
  }
}
```

Example 3.2 (OpenC2 Command, transfer via HTTPS)

The IDs used in this example are notional and for illustrative purposes, they do not represent real objects.

```
{
  "type": "openc2",
  "command_b64":
"ewogICJoZWFkZXJzIjogewogICAgInJlcXVlc3RfaWQiOiAiZDFhYzA0ODktZWQ1MS00MzQ1 ...
B9CiAgfQp9",
  "agent": "oc2-http-api--5ceccd83-8052-4d12-8b42-e941647867c7",
  "step_variables": {
    "__http-endpoints__:value": {
      "ipv4": ["11.22.33.44", "55.66.77.88"]
    }
  }
}
```

The abbreviated content of the base64 command (`command_b64`) in both of the above examples is the encoded version of the OpenC2 content that is shown below (decoded version). The command content is shown as text for illustration purposes only.

```
{
  "headers": {
    "request_id": "d1ac0489-ed51-4345-9175-f3078f30afe5",
    "created": 1545257700000,
    "from": "oc2producer.company.net",
    "to": ["oc2consumer.company.net"]
  },
  "body": {
    "openc2": {
      "request": {
        "action": "deny",
        "target": {
          "ipv4_connection": {
            "protocol": "tcp",
            "src_addr": "1.2.3.4",
            "src_port": 10996,
            "dst_addr": "198.2.3.4",
            "dst_port": 80
          }
        },
        "args": {
          "start_time": 1534775460000,
          "duration": 500,
          "response_requested": "ack",
          "slpf": {
            "drop_process": "none"
          }
        },
        "profile": "slpf"
      }
    }
  }
}
```

3.2 Base64 Encoding and Decoding

CACAO uses base64 encoding, as defined in Section 4 of [RFC 4648], to
preserve the integrity of complex commands and scripts, such as the example
OpenC2 command shown in Section 3.1. The
default encoding for OpenC2 commands and responses is JSON, as defined in section 3.1.4 of the
[OpenC2 Language Specification]. The recommended conventions
for the handling of base64 encoding of OpenC2 commands and responses in the
context of a CACAO playbook being executed by a CACAO Consumer are:

- OpenC2 commands in JSON format will be base64 encoded by the CACAO Producer creating the playbook and stored as a string in the `command_b64` field of an OpenC2 command object.
- The base64-encoded content will be passed to the specified OpenC2 CACAO agent when the OpenC2 command object is executed.
- The OpenC2 CACAO agent will decode the base64-encoded content and re-encode in the appropriate transfer encoding (e.g., JSON, CBOR) for transfer to the Consumer identified by the specified OpenC2 CACAO target, incorporating any command content specified by CACAO variables when re-encoding the command.
- The mechanism for exchange of the the transfer-encoded command between agent and target is the responsibility of the CACAO Consumer executing the playbook.
- The OpenC2 CACAO agent will accept transfer-encoded responses from the OpenC2 CACAO target.

> **To-Do:** what happens to responses once they are accepted by the OpenC2 CACAO agent? Are they base64 encoded? Where do they go? How are they represented back to the CACAO Consumer to support any decision(s) that are dependent on the response(s)?

3.3 Invoking OpenC2 via Playbook Action Step

> **To-Do:** How much complexity is worthwhile here?
>
> - Selection of agent based on values in **mqtt-topics** or **http-endpoints**?
> - Selection of targets based on *something* that identifies the desired AP?

---

4 OpenC2 CACAO Agents and Targets

4.1 OpenC2 CACAO Agents

CACAO agents for OpenC2 correspond to OpenC2 transfer specifications. Each type of OpenC2 CACAO agent supports the use of one transfer protocol for sending OpenC2 commands and receiving OpenC2 responses.

4.1.1 MQTT Broker Agent

An `mqtt-broker` agent type supports publish / subscribe communications via the OASIS [MQTT v5] protocol. The CACAO `agent-target-type-ov` "Devices and Equipment" subcategory is extended as follows:

Type	Description
`mqtt-broker`	A publish/subscribe message transfer agent conforming to the OASIS MQTT v5.0 protocol.

The `mqtt-broker` agent is not specific to OpenC2 but when used for sending and receiving OpenC2 messages its use **MUST** conform to the [OpenC2 MQTT Transfer Specification]. In particular:

- Topics for message publication passed to this agent for transmitting OpenC2 messages **MUST** conform to the default topic structure specified in Section 2.2 of the OpenC2 MQTT Transfer Specification.

- A CACAO `mqtt-broker` agent in an environment using OpenC2 **MUST** subscribe to the response topics specified in Section 2.2 of the OpenC2 MQTT Transfer Specification.

The `__mqtt-topics__` variable (see Section 5.1) is used to pass the requested topic(s) for publishing a message to an `mqtt-broker` agent.

This type defines an MQTT Broker agent object and is used for messages to be transmitted via MQTT. In addition to the inherited properties, this section defines the following additional properties that are valid for this type.

Property Name	Data Type	Details
type (required)	`string`	The value of this property **MUST** be `mqtt-broker`
address (required)	`dictionary`	The key for each entry in the dictionary **MUST** be a string that uniquely identifies one or more address types. The key(s) MUST be one of the following values `dname` (domain name), `ipv4` , `ipv6` , `l2mac` , `vlan` , or `url` . The dictionary value associated with each key **MUST** be a `list` of `string` that contains the corresponding address(es) for that particular key type. The `address` dictionary for an `mqtt-broker` agent **MUST** specify only a single address for the broker to be used.
authentication_info (optional)	`identifier`	This property contains an ID reference to a CACAO `authentication-info` object that is stored at the Playbook level in the `authentication_info_definitions` property. The ID **MUST** reference a CACAO `authentication-info` object (see section 6 of the [CACAO v2.0 Specification]).
category (optional)	`list` of `open-vocab`	One or more identified categories of security infrastructure types that this agent represents (see section 7.11.1 of the [CACAO v2.0 Specification]). The value for this property **SHOULD** come from the `security-category-type-ov` vocabulary.

Example 4.1.1 (MQTT Broker Agent)

The IDs used in this example are notional and for illustrative purposes, they do not represent real objects.

```
"agent_definitions": {
  "mqtt-broker--7125c6f6-7f78-4a3d-8a43-f20d20632305": {
    "type": "mqtt-broker",
    "name": "mqtt.example.com",
    "description": "An MQTT pub/sub broker for company example dot com",
    "address": {
      "url": ["https://mqtt.example.com"]
    },
    "category": "server"
  }
}
```

> **To-Do:** Should we define a new `security-category-ov` entry `message-broker` or is `server` sufficient?

4.1.2 OpenC2 HTTP-API Agent

An `oc2-http-api` agent type supports point-to-point communications via the HTTP or HTTPS protocol. The CACAO `agent-target-type-ov` "Devices and Equipment" subcategory is extended as follows:

Type	Description
`oc2-http-api`	An agent capable of transferring OpenC2 commands to one or more specified endpoints per requirements of the OASIS [OpenC2 HTTPS Transfer Protocol Specification].

The `oc2-http-api` agent is an extension of the CACAO `http-api` agent to address the particular requirements for handling OpenC2 messages defined in the

[OpenC2 HTTPS Transfer Protocol Specification]. In particular:

- The preferred transfer protocols is HTTPS.
- The HTTP message **MUST** begin with the headers:
  - `POST /.well-known/openc2 HTTP/1.1`
  - `Content-type: application/openc2+json;version=1.0`
- The URL for destinations (i.e., OpenC2 consumers) **MUST** use the URI scheme specified in Section 3.2.2 of the [OpenC2 HTTPS Transfer Protocol Specification] (i.e., `https://<consumer address>/.well-known/openc2`).

The `__http-endpoints__` variable (see Section 5.2) is used to pass the desired destinations for transferring an OpenC2 message to an `oc2-http-api` agent.

This type defines an OpenC2 HTTP-API agent object and is used for messages to be transmitted via HTTP(S). In addition to the inherited properties, this section defines the following additional properties that are valid for this type.

Property Name	Data Type	Details
type (required)	`string`	The value of this property **MUST** be `oc2-http-api`
address (required)	`dictionary`	The destination(s) for transfer of this OpenC2 command. The values for `address` are taken from the `__http_endpoints__` variable
authentication_info (optional)	`identifier`	This property contains an ID reference to a CACAO `authentication-info` object that is stored at the Playbook level in the `authentication_info_definitions` property. The ID **MUST** reference a CACAO `authentication-info` object (see section 6 of the [CACAO v2.0 Specification]).
category (optional)	`list` of `open-vocab`	One or more identified categories of security infrastructure types that this agent represents (see section 7.11.1 of the [CACAO v2.0 Specification]). The value for this property **SHOULD** come from the `security-category-type-ov` vocabulary.

Example 4.1.1 (OpenC2 HTTP-API)

The IDs used in this example are notional and for illustrative purposes, they do not represent real objects.

```
"agent_definitions": {
  "oc2-http-api--5ceccd83-8052-4d12-8b42-e941647867c7": {
    "type": "oc2-http-api",
    "name": "p2p.example.com",
    "description": "An MQTT pub/sub broker for company example dot com",
    "address": "__http-endpoints__:value",
    "category": "server"
  }
}
```

> **To-Do:** Should we define a new `security-category-ov` entry `oc2-consumer` or is `server` sufficient?

4.2 OpenC2 CACAO Targets

OpenC2 CACAO Targets correspond to OpenC2 Actuator Profile (AP) specifications. An `openc2` command object **SHOULD** specify one or more CACAO targets to identify the OpenC2 APs to be invoked for the execution of the object's OpenC2 command.

An OpenC2 CACAO target **MUST** be of type `security-category` as defined in Section 7.11 of the [CACAO v2.0 Specification].
The CACAO `security-category-type-ov` is extended as follows:

Type	Description
`openc2-consumer`	A category of CACAO targets representing OpenC2 Consumers supporting one or more OpenC2 APs

The `category` value of an OpenC2 CACAO target **SHALL** be set to `openc2-consumer`.

The `security-category` target object is extended with a new property: `openc2-profile`. The resulting extended `security-category` target is structured as follows:

Property Name	Data Type	Details
type (required)	`string`	The value of this property **MUST** be `security-category`.
category (required)	`list` of `open-vocab`	The value for this property **MUST** include `openc2-consumer`.
openc2-profile (required)	`string`	The value for this property **SHOULD** be the "Property Name" of a registered OpenC2 AP.

The Property Names of registered OpenC2 APs are found in the [OpenC2 Namespace Registry].
For example an OpenC2 CACAO target for the Stateless Packet Filtering AP would specify the profile as follows:

```
"openc2-profile" : "slpf"
```

Example 4.2 (OpenC2 Target)

The IDs used in this example are notional and for illustrative purposes, they do not represent real objects.

```
"target_definitions": {
  "security-category--09b5b900-f333-41fd-9fdc-cb466e9b1f20": {
    "type": "security-category",
    "name": "OC2 Packet Filter",
    "category": [ "openc2-consumer" ],
    "openc2-profile" : "slpf"
  }
}
```

> **To-Do:** determine what, if anything, needs to be defined beyond the correlation of APs and CACAO Targets.

---

5 Standardized Playbook Variables

This section defines a set of standardized CACAO variables for use when invoking an MQTT broker or OpenC2 HTTP API agent to handle message transfer. These CACAO variables are playbook variables whose values can be set internally via an `openc2` command object or from a `playbook-action` step in a calling playbook and accessed by the appropriate agent.

A standardized CACAO variable is also defined for returning OpenC2 responses to the calling `openc2` action step for subsequent processing.

5.1 `__mqtt-topics__` Variable

The `__mqtt-topics__` variable is used to convey a list of MQTT topics onto which a message should be published. The `variable-type-ov` is extended as follows:

Vocabulary Value	Description	Examples
`topic-list`	An object containing a list of strings that identify one or more publish / subscribe topics to which a message should be published.	`"type": "topic-list",` `"value":

The `mqtt-broker` agent is general purpose. MQTT offers great flexibility regarding topic naming. The format of the topic names in the `__mqtt-topics__` value should be appropriate to the application. The [OpenC2 MQTT Transfer Specification] provides specific guidance regarding the use of MQTT topics for OpenC2 message transfer. When an `mqtt-broker` agent is employed for sending and receiving OpenC2 messages the topics specified as `__mqtt-topics__:value` **SHOULD** conform to the topic structure guidance in Section 2.2 of the [OpenC2 MQTT Transfer Specification]. Other users of the MQTT Broker CACAO agent and `__mqtt-topics__` variable for publish / subscribe messaging should apply their own corresponding guidance.

Example 5.1 (`__mqtt-topics__`)

```
{
  "type": "playbook",
  …,
  "playbook_variables": {
    "__mqtt-topics__": {
      "type": "topic-list",
      "description": "Provides a list of topics to publish a message via an MQTT
broker",
      "value": {
          "topic-array": ["oc2/cmd/ap/pf","oc2/cmd/ap/edr"]
      },
      "constant": false,
      "external": true
    }
  }
}
```

5.2 `__http-endpoints__` Variable

The `__http_endpoints__` variable is used to convey a list of endpoints to an OpenC2 command should be published.

The `variable-type-ov` for `__http-endpoints__` **MUST** be `dictionary`.

The value of `__http-endpoints__` **MUST** be a `dictionary` of address(es) as defined for the CACAO `http-api` agent object (section 7.8 of the [CACAO Playbooks] specification).

Example 5.2 (`__http-endpoints__`)

```
{
  "type": "playbook",
  …,
  "playbook_variables": {
    "__http-endpoints__": {
      "type": "dictionary",
      "description": "A list of endpoints for delivery of an OpenC2 command via
HTTP(S)",
      "value": {
        "url": ["https://oc2consumer.example.com"],
        "ipv4" : ["11.22.33.44", "55.66.77.88"]
      },
      "constant": false,
      "external": true
    }
  }
}
```

5.3 `__openc2-responses__` Variable

The `__openc2-responses` variable is used to aggregate the responses from one or more OpenC2 Consumers for return to the calling `openc2` command action step. The return of results from OpenC2 Consumer responses enables conditional processing by subsequent action steps in the CACAO playbook.

> To-Do: confirm this is a suitable `variable-type-ov` for this variable. Since it's an `-ov` a new type may be in order.

The `variable-type-ov` for `__openc2-responses__` **MUST** be `dictionary`.

> To-Do: develop more realistic response content for this example

Example 5.3 (`__openc2-response__`)

```
{
  "type": "playbook",
  …,
  "playbook_variables": {
    "__openc2-response__": {
      "type": "dictionary",
      "description": "A collection of responses for an OpenC2 command",
      "value": {
        "device1": { <response from Device1>},
        "device4": { <response from Device4>},
        "device9": { <response from Device9>}
      },
      "constant": false,
      "external": false
    }
  }
}
```

6 Conformance

(Note: The OASIS TC Process requires that a specification approved by the TC at the Committee Specification Public Review Draft, Committee Specification or OASIS Standard level must include a separate section, listing a set of numbered conformance clauses, to which any implementation of the specification must adhere in order to claim conformance to the specification (or any optional portion thereof). This is done by listing the conformance clauses

here.
For the definition of "conformance clause," see OASIS Defined Terms.

See "Guidelines to Writing Conformance Clauses":
http://docs.oasis-open.org/templates/TCHandbook/ConformanceGuidelines.html.

Remove this note before submitting for publication.)

Appendix A. References

This appendix contains the normative and informative references that are used in this document.

While any hyperlinks included in this appendix were valid at the time of publication, OASIS cannot guarantee their long-term validity.

A.1 Normative References

The following documents are referenced in such a way that some or all of their content constitutes requirements of this document.

(Reference sources:
For references to IETF RFCs, use the approved citation formats at:
http://docs.oasis-open.org/templates/ietf-rfc-list/ietf-rfc-list.html.
For references to W3C Recommendations, use the approved citation formats at:
http://docs.oasis-open.org/templates/w3c-recommendations-list/w3c-recommendations-list.html.
Remove this note before submitting for publication.)

[CACAO-Security-Playbooks-v2.0]

CACAO Security Playbooks Version 2.0. Edited by Bret Jordan and Allan Thomson. 27 November 2023. OASIS Committee Specification 01. https://docs.oasis-open.org/cacao/security-playbooks/v2.0/cs01/security-playbooks-v2.0-cs01.html. Latest version: https://docs.oasis-open.org/cacao/security-playbooks/v2.0/security-playbooks-v2.0.html.

[mqtt-v5.0]

MQTT Version 5.0. Edited by Andrew Banks, Ed Briggs, Ken Borgendale, and Rahul Gupta. 07 March 2019. OASIS Standard. https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html. Latest version: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html.

[OpenC2-HTTPS-v1.1]

Specification for Transfer of OpenC2 Messages via HTTPS Version 1.1. Edited by David Lemire. Latest stage: https://docs.oasis-open.org/openc2/open-impl-https/v1.1/open-impl-https-v1.1.html

[OpenC2-Lang-v1.1]

Open Command and Control (OpenC2) Language Specification Version 1.1. Edited by Duncan Sparrell and Toby Considine. Latest stage: https://docs.oasis-open.org/openc2/oc2ls/v1.1/oc2ls-v1.1.html

[OpenC2-MQTT-v1.0]

Specification for Transfer of OpenC2 Messages via MQTT Version 1.0. Edited by David Lemire. 19 November 2021. OASIS Committee Specification 01. https://docs.oasis-open.org/openc2/transf-mqtt/v1.0/cs01/transf-mqtt-v1.0-cs01.html. Latest stage: https://docs.oasis-open.org/openc2/transf-mqtt/v1.0/transf-mqtt-v1.0.html

[OpenC2-Namespaces]

OpenC2 Namespace Registry. https://github.com/oasis-tcs/openc2-oc2arch/blob/published/namespace-registry.md

[OpenC2-SLPF-v1.1]

Open Command and Control (OpenC2) Profile for Stateless Packet Filtering Version 1.1. Edited by Joe Brule, Duncan Sparrell, and Alex Everett. Latest stage: https://docs.oasis-open.org/openc2/oc2slpf/v1.1/oc2slpf-v1.1.html

[RFC2119]

Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, http://www.rfc-editor.org/info/rfc2119.

[RFC4648]

Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, https://www.rfc-editor.org/info/rfc4648.

[RFC8174]

Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, http://www.rfc-editor.org/info/rfc8174.

[RFC8259]

Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, https://www.rfc-editor.org/info/rfc8259.

A.2 Informative References

[OpenC2-Arch-v1.0]

Open Command and Control (OpenC2) Architecture Specification Version 1.0. Edited by Duncan Sparrell. 30 September 2022. OASIS Committee Specification 01. https://docs.oasis-open.org/openc2/oc2arch/v1.0/cs01/oc2arch-v1.0-cs01.html. Latest stage: https://docs.oasis-open.org/openc2/oc2arch/v1.0/oc2arch-v1.0.html.

[RFC3552]

Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, https://www.rfc-editor.org/info/rfc3552.

Appendix B. Safety, Security and Privacy Considerations

OpenC2, as a cyber defense automation tool, is high-value target for adversaries attempting to exploit an environment where it is used. Appendix B of the OpenC2 Architecture Specification [OpenC2-Arch-v1.0] discusses:

- Threats to OpenC2
- Applying security services to OpenC2 operations
- Network topology considerations for OpenC2 messages

Refer to that document for a review of these topics in the context of OpenC2.

Appendix B of the [CACAO v2.0] Specification
includes information regarding security and privacy considerations for CACAO playbook generation, consumption, and content sensitivity. Refer to that document for information regarding these topics in the context of CACAO.

Appendix C. Acknowledgments

Note: A Work Product approved by the TC must include a list of people who participated in the development of the Work Product. This is generally done by collecting the list of names in this appendix. This list shall be initially compiled by the Chair, and any Member of the TC may add or remove their names from the list by request. Remove this note before submitting for publication.

C.1 Special Thanks

Substantial contributions to this document from the following individuals are gratefully acknowledged:

Participant Name, Affiliation or "Individual Member"

C.2 Participants

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

OpenC2 TC Members:

First Name	Last Name	Company
Philippe	Alman	Something Networks
Alex	Amirnovman	Company B
Kris	Anderman	Mini Micro
Darren	Anstman	Big Networks

Appendix D. Revision History

Revision	Date	Editor	Changes Made
specname-v1.0-wd01	yyyy-mm-dd	Editor Name	Initial working draft

Appendix E. Use Cases and Examples

E.1 Use Cases

E.1.1 Multiple OpenC2 Consumers With Common Profile

E.1.2 OpenC2 Command With Response Requested

E.1.3 OpenC2 Message Transfer via MQTT

E.1.4 Extended OpenC2 Consumer Execution Times

E.2 Examples

This section presents example CACAO playbooks aligned with this extension
specification for notional OpenC2 command scenarios.

E.2.1 OpenC2 Single Consumer Command / Response via MQTT

In this example an OpenC2 command is sent to a single OpenC2 consumer using MQTT as the
transfer mechanism. Consistent with the [OpenC2 MQTT Transfer Specification],
the consumer is addressed as an individual device. This scenario requires:

- A CACAO `openc2` command object containing the OpenC2 command to be sent
- A CACAO agent defining an MQTT broker
- A CACAO `__mqtt-topics__` variable to convey the publication topic
- A CACAO `security-category` target to identify the OpenC2 actuator profile being invoked'
- A CACAO `__openc2-responses__` variable to return the consumer's response for any subsequent decision logic
- Knowledge of the OpenC2 consumer device address

This example assumes an OpenC2 consumer implementing the [*OpenC2 Profile for Stateless Packet Filtering*] (SLPF). A notional device identification of `oc2-slpf-consumer` is assumed for MQTT topic addressing and the OpenC2 command denies outbound FTP transfers (example A.1.2 in the SLPF AP). The OpenC2 command to be sent is:

```
{
  "action": "deny",
  "target": {
    "ipv4_connection": {
      "protocol": "tcp",
      "src_port": 21
    }
  },
  "args": {
    "response_requested": "ack",
    "slpf": {
      "drop_process": "false_ack",
      "direction": "egress"
    }
  },
  "profile": {
    "slpf": {}
  }
}
```

The following is an excerpt from the CACAO playbook showing the `openc2` action workflow step and the CACAO agent and target definitions. The `command_b64` property is truncated for presentation purposes.

```
"action--629b12de-f0e5-402e-944d-2c4df0883caa": {
  "name": "Example OpenC2 Action Step",
  "description": "Example openc2 action step based on the OpenC2 Extension for
CACAO Specification",
  "step_variables": {
    "__mqtt-topics__": {
      "type": "topic-list",
      "description": "example of MQTT topics IAW OC2 Ext. for CACAO Spec",
      "value":
      {
         "topic-array": ["oc2/cmd/device/oc2-slpf-consumer"]
       },
       "constant": false,
      "external": false
    },
    "__openc2-responses__": {
      "type": "dictionary",
      "description": "Captures responses returned from OpenC2 consumers",
      "constant": false,
      "external": false
    }
  },
  "on_completion": "end--4f6a186d-89da-4da7-9da4-0df89b223658",
  "type": "action",
  "commands": [
    {
      "type": "openc2",
      "description": "Example openc2 command",
      "command_b64": "ewogICJhY3Rpb24iOi ..."
    }
  ],
  "agent": "mqtt-broker--29ede420-29c1-4fa9-8d91-5ed0a26d6708",
  "targets": [
    "security-category--b0852f76-1c36-4f00-8dd7-bf433cdbb954"
  ]
},
"end--4f6a186d-89da-4da7-9da4-0df89b223658": {
  "type": "end"
},

"agent_definitions": {
  "mqtt-broker--29ede420-29c1-4fa9-8d91-5ed0a26d6708": {
    "type": "mqtt-broker",
    "name": "Example MQTT broker",
    "description": "Agent to provide an MQTT 5.0 broker",
    "location": {
      "name": "example-mqtt-broker",
      "network_details": "mqtt.example.com"
    }
  }
},

"target_definitions": {
  "security-category--b0852f76-1c36-4f00-8dd7-bf433cdbb954": {
    "type": "openc2-consumer",
    "category": ["openc2-consumer"],
    "name": "openc2 SLPF consumer",
    "openc2-profile":"slpf"
  }
}
```

E.2.2 OpenC2 Multiple Consumer Command / Response via MQTT

E.2.3 OpenC2 Command / Response via HTTPS

Appendix F. Notices