Open Command and Control (OpenC2) Profile for Software Bill of Materials (SBOM) Actuator Version 1.0

Working Document eventually to be Draft Committee Specification CS01

25 Feb 2020

This version:
TBD

Previous version:
TBD

Latest version:
TBD

Technical Committee:
OASIS Open Command and Control (OpenC2) TC

Chairs:

Joe Brule (jbrule@nsa.gov), National Security Agency
Duncan Sparrell (duncan@sfractal.com), sFractal Consulting LLC

Editors:

Duncan Sparrell (duncan@sfractal.com), sFractal Consulting

Abstract:

Open Command and Control (OpenC2) is a concise and extensible language to enable the command and control of cyber defense components, subsystems and/or systems in a manner that is agnostic of the underlying products, technologies, transport mechanisms or other aspects of the implementation.

A Software Bill of Materials (SBOM) is blah blah ...

This profile defines the Actions, Targets, Specifiers and Options that are consistent with the version 1.0 of the OpenC2 Language Specification ([OpenC2-Lang-v1.0]) in the context of the SBOM actuator profile.

Status:

This is a working draft. Eventually this paragraph will be deleted and it will say the following:

This document was last revised or approved by the OASIS Open Command and Control (OpenC2) TC on the above date. The level of approval is also listed above. Check the "Latest version" location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Technical Committee (TC) are listed at https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=openc2#technical.

TC members should send comments on this specification to the TC’s email list. Others should send comments to the TC's public comment list, after subscribing to it by following the instructions at the "Send A Comment" button on the TC's web page at https://www.oasis-open.org/committees/openc2/.

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

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

Citation format:

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

Copyright © OASIS Open 2020. All Rights Reserved.

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

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

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

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

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.
OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS’ procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The name "OASIS" is a trademark of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see https://www.oasis-open.org/policies-guidelines/trademark for above guidance.

Table of Contents

- 1 Introduction
  - 1.1 IPR Policy
  - 1.2 Terminology
  - 1.3 Normative References
  - 1.4 Non-Normative References
  - 1.5 Document Conventions
    - 1.5.1 Naming Conventions
    - 1.5.2 Font Colors and Style
  - 1.6 Overview
  - 1.7 Goal
  - 1.8 Purpose and Scope
- 2 OpenC2 Language Binding
  - 2.1 OpenC2 Command Components
    - 2.1.1 Actions
    - 2.1.2 Targets
    - 2.1.3 Command Arguments
    - 2.1.4 Actuator Specifiers
  - 2.2 OpenC2 Response Components
    - 2.2.1 Common Results
    - 2.2.2 SBOM Results
    - 2.2.3 Response Status Codes
  - 2.3 OpenC2 Commands
    - 2.3.1 Query
- 3 Conformance statements
1 Introduction

The content in this section is non-normative, except where it is marked normative.

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.

OpenC2 allows the application producing the commands to discover the set of capabilities supported by the managed devices. These capabilities permit the managing application to adjust its behavior to take advantage of the features exposed by the managed device. The capability definitions can be easily extended in a noncentralized manner, allowing standard and non-standard capabilities to be defined with semantic and syntactic rigor.

1.1 IPR Policy

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

1.2 Terminology

This section is normative.

- **Action**: The task or activity to be performed (e.g., 'deny').
• **Actuator:** The function performed by the Consumer that executes the Command (e.g., 'Stateless Packet Filtering').

• **Argument:** A property of a Command that provides additional information on how to perform the Command, such as date/time, periodicity, duration, etc.

• **Command:** A Message defined by an Action-Target pair that is sent from a Producer and received by a Consumer.

• **Consumer:** A managed device / application that receives Commands. Note that a single device / application can have both Consumer and Producer capabilities.

• **Message:** A content- and transport-independent set of elements conveyed between Consumers and Producers.

• **Producer:** A manager application that sends Commands.

• **Response:** A Message from a Consumer to a Producer acknowledging a Command or returning the requested resources or status to a previously received Command.

• **Specifier:** A property or field that identifies a Target or Actuator to some level of precision.

• **Target:** The object of the Action, i.e., the Action is performed on the Target (e.g., IP Address).

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 [RFC2119] and [RFC8174] when, and only when, they appear in all capitals, as shown here.

### 1.3 Normative References

needs work

[ RFC1123 ]


[ RFC2119 ]


[ RFC2780 ]


[ RFC4443 ]


[ RFC8174 ]

[RFC8259]


[OpenC2-Lang-v1.0]


1.4 Non-Normative References

needs work

[RFC3339]


[RFC4291]


[RFC6891]


[RFC5237]


[OpenC2-HTTPS-v1.0]


[ACD]


[IACD]
1.5 Document Conventions

1.5.1 Naming Conventions

- [RFC2119]/[RFC8174] key words (see Section 1.2) are in all uppercase.
- All property names and literals are in lowercase, except when referencing canonical names defined in another standard (e.g., literal values from an IANA registry).
- Words in property names are separated with an underscore (_), while words in string enumerations and type names are separated with a hyphen (-).
- The term "hyphen" used here refers to the ASCII hyphen or minus character, which in Unicode is "hyphen-minus", U+002D.

1.5.2 Font Colors and Style

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 (...).

Example:

redo example with sbom example

```json
{
   "action": "deny",
   "target": {
       "file": {
           "hashes": {
               "sha256": "22fe72a34f006ea67d26bb7004e2b6941b5c3953d43ae7ec24d41b1a928a6973"
           }
       }
   }
}
```

1.6 Overview
In general, there are two types of participants involved in the exchange of OpenC2 Messages, as depicted in Figure 1-1:

1. **Producers**: A Producer is an entity that creates Commands to provide instruction to one or more systems to act in accordance with the content of the Command. A Producer may receive and process Responses in conjunction with a Command.

2. **Consumers**: A Consumer is an entity that receives and may act upon a Command. A Consumer may create Responses that provide any information captured or necessary to send back to the Producer.

![OpenC2 Message Exchange Diagram](image)

**Figure 1-1. OpenC2 Message Exchange**

OpenC2 is a suite of specifications for Producers and Consumers to command and execute cyber defense functions. These specifications include the OpenC2 Language Specification, Actuator Profiles, and Transfer Specifications. The OpenC2 Language Specification and Actuator Profile specifications focus on the language content and meaning at the Producer and Consumer of the Command and Response while the transfer specifications focus on the protocols for their exchange.

- **The OpenC2 Architecture Specification ([oc2arch-v1.0-csprd01.md])** includes a comprehensive description of these components and the Language Specifications, Transfer Specifications, and Actuator Profiles that form their constituent parts.
- **The OpenC2 Language Specification ([OpenC2-Lang-v1.0])** provides the semantics for the essential elements of the language, the structure for Commands and Responses, and the schema that defines the proper syntax for the language elements that represents the Command or Response.
- **OpenC2 Actuator Profiles** specify the subset of the OpenC2 language relevant in the context of specific Actuator functions. Cyber defense components, devices, systems and/or instances may (in fact are likely to) implement multiple Actuator profiles. Actuator profiles extend the language by defining Specifiers that identify the Actuator to the required level of precision. Actuator Profiles may define Command Arguments and Targets that are relevant and/or unique to those Actuator functions.

- **OpenC2 Transfer Specifications** utilize existing protocols and standards to implement OpenC2 in specific environments. These standards are used for communications and security functions beyond the scope of the language, such as message transfer encoding, authentication, and end-to-end transport of OpenC2 Messages.

The OpenC2 Language Specification defines a language used to compose Messages for command and control of cyber defense systems and components. A Message consists of a header and a payload *(defined as a Message body in the OpenC2 Language Specification Version 1.0 and specified in one or more Actuator profiles)*.
The language defines two payload structures:

1. **Command**: An instruction from one system known as the Producer, to one or more systems, the Consumer(s), to act on the content of the Command.

2. **Response**: Any information sent back to the Producer as a result of the Command.

OpenC2 implementations integrate the related OpenC2 specifications described above with related industry specifications, protocols, and standards. Figure 1-2 depicts the relationships among OpenC2 specifications, and their relationships to other industry standards and environment-specific implementations of OpenC2. Note that the layering of implementation aspects in the diagram is notional, and not intended to preclude any particular approach to implementing the needed functionality (for example, the use of an application-layer message signature function to provide message source authentication and integrity).

![OpenC2 Documentation and Layering Model](image)

**Figure 1-2. OpenC2 Documentation and Layering Model**

OpenC2 is conceptually partitioned into four layers as shown in Table 1-1.

**Table 1-1. OpenC2 Protocol Layers**

<table>
<thead>
<tr>
<th>Layer</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function-Specific Content</td>
<td>Actuator Profiles (standard and extensions)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Common Content</td>
<td>Language Specification [OpenC2-Lang-v1.0]</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Message</td>
<td>Transfer Specifications ([OpenC2-HTTPS-v1.0], OpenC2-over-CoAP, ...)</td>
</tr>
<tr>
<td>Secure Transport</td>
<td>HTTPS, CoAP, MQTT, OpenDXL, ...</td>
</tr>
</tbody>
</table>

- The **Secure Transport** layer provides a communication path between the Producer and the Consumer. OpenC2 can be layered over any standard transport protocol.
- The **Message** layer provides a transfer- and content-independent mechanism for conveying Messages. A transfer specification maps transfer-specific protocol elements to a transfer-independent set of message elements consisting of content and associated metadata.
- The **Common Content** layer defines the structure of Commands and Responses and a set of common language elements used to construct them.
- The **Function-specific Content** layer defines the language elements used to support a particular cyber defense function. An Actuator profile defines the implementation conformance requirements for that function. Producers and Consumers will support one or more profiles.

The components of a Command are an Action (what is to be done), a Target (what is being acted upon), an optional Actuator (what is performing the command), and Command Arguments, which influence how the Command is to be performed. An Action coupled with a Target is sufficient to describe a complete Command. Though optional, the inclusion of an Actuator and/or Command Arguments provides additional precision to a Command.
The components of a Response are a numerical status code, an optional status text string, and optional results. The format of the results, if included, depend on the type of Response being transferred.

1.7 Goal

The goal of the OpenC2 Language Specification is to provide a language for inter-operating between functional elements of cyber defense systems. This language used in conjunction with OpenC2 Actuator Profiles and OpenC2 Transfer Specifications allows for vendor-agnostic cybertime response to attacks.

The Integrated Adaptive Cyber Defense (IACD) framework defines a collection of activities, based on the traditional OODA (Observe–Orient–Decide–Act) Loop [IACD]:

- Sensing: gathering of data regarding system activities
- Sense Making: evaluating data using analytics to understand what’s happening
- Decision Making: determining a course-of-action to respond to system events
- Acting: Executing the course-of-action

The goal of OpenC2 is to enable coordinated defense in cyber-relevant time between decoupled blocks that perform cyber defense functions. OpenC2 focuses on the Acting portion of the IACD framework; the assumption that underlies the design of OpenC2 is that the sensing/analytics have been provisioned and the decision to act has been made. This goal and these assumptions guide the design of OpenC2:

- **Technology Agnostic:** The OpenC2 language defines a set of abstract atomic cyber defense actions in a platform and implementation agnostic manner
- **Concise:** A Command is intended to convey only the essential information required to describe the action required and can be represented in a very compact form for communications-constrained environments
- **Abstract:** Commands and Responses are defined abstractly and can be encoded and transferred via multiple schemes as dictated by the needs of different implementation environments
- **Extensible:** While OpenC2 defines a core set of Actions and Targets for cyber defense, the language is expected to evolve with cyber defense technologies, and permits extensions to accommodate new cyber defense technologies.

1.8 Purpose and Scope

A ‘Software Bill of Materials’ (SBOM) is a blah blah. All devices containing software SHOULD have a SBOM but it may or may not be accessible via OpenC2 command. Reasons for having an SBOM blah blah ref NTIA docs. An actuator with the functionality to return an SBOM via OpenC2 MUST conform to this specification. The scope of this profile is limited to actuators with SBOM capability, herein referred to as SBOM.

This Actuator profile specifies the set of Actions, Targets, Specifiers, and Command Arguments that integrates SBOM functionality with the Open Command and Control (OpenC2) Command set. Through this Command set, cyber security orchestrators may gain visibility into and provide control over the SBOM functionality in a manner that is independent of the instance of the SBOM function.
All components, devices and systems that provide SBOM functionality will implement the OpenC2 Actions, Targets, Specifiers and Arguments identified as required in this document. Actions that are applicable, but not necessarily required, for SBOM will be identified as optional.

The purpose of this document is to:

- Identify the required and optional OpenC2 Actions for Actuators with SBOM functionality
- Identify the required and optional Target types for each Action in the SBOM class of Actuators
- Identify Actuator-Specifiers and Arguments for each Action/Target pair that are applicable and/or unique to the SBOM class of Actuators
- Annotate each Action/Target pair with a justification and example, and provide sample OpenC2 Commands to a SBOM with corresponding Responses

This SBOM profile:

- Does not define or implement Actions beyond those defined in Version 1.0 of the [OpenC2-Lang-v1.0]
- Is consistent with Version 1.0 of the OpenC2 Language Specification

Cyber defense systems that are utilizing OpenC2 may require the following components to implement the SBOM profile:

- OpenC2 Producers: Devices that send Commands, receive Responses, and manage the execution of Commands involving one or more SBOM or other Actuators with SBOM capability. The OpenC2 Producer needs a priori knowledge of which Commands the Actuator can process and execute, therefore must understand the profiles for any device that it intends to command
- OpenC2 Consumers: Devices or instances that provide stateless packet filtering functions. Typically these are Actuators that execute the cyber defense function, but could be orchestrators (i.e., a device or instance that forwards Commands to the Actuator)

Though cyber defense components, devices, systems and/or instances may implement multiple Actuator profiles, a particular OpenC2 Message may reference at most a single Actuator profile. The scope of this document is limited to SBOM.

This specification is organized into three major sections.

Section One (this section) provides a non-normative overview of the suite of specifications that realize OpenC2. This section provides references as well as defines the scope and purpose of this specification.

**Section Two** (normative) binds this particular profile to the OpenC2 Language Specification. Section Two enumerates the components of the language specification that are meaningful in the context of SBOM and defines components that are applicable to this distinct profile. Section Two also defines the Commands (i.e., the Action/Target pairs) that are permitted in the context of SBOM.

**Section Three** (normative) presents definitive criteria for conformance so that cyber security stakeholders can be assured that their products, instances and/or integrations are compatible with OpenC2.

**Annex A** (non-normative) provides multiple examples of Commands and associated Responses (JSON serialization) to facilitate development.
2. OpenC2 Language Binding

This section is normative

This section defines the set of Actions, Targets, Specifiers, and Arguments that are meaningful in the context of an SBOM. This section also describes the appropriate format for the status and properties of a Response frame. This section is organized into three major subsections; Command Components, Response Components and Commands.

Extensions to the Language Specification are defined in accordance with [OpenC2-Lang-v1.0], Section 3.1.5, where:

1. The unique name of the SBOM schema is oasis-open.org/openc2/v1.0/ap-sbom
2. The namespace identifier (nsid) referring to the SBOM schema is: sbom
3. The definitions of and conformance requirements for these types are contained in this document

2.1 OpenC2 Command Components

The components of an OpenC2 Command include Actions, Targets, Actuators and associated Arguments and Specifiers. Appropriate aggregation of the components will define a Command-body that is meaningful in the context of an SBOM.

This specification identifies the applicable components of an OpenC2 Command. The components of an OpenC2 Command include:

- Action: A subset of the Actions defined in the OpenC2 Language Specification that are meaningful in the context of a SBOM.
  - This profile SHALL NOT define Actions that are external to Version 1.0 of the OpenC2 Language Specification
  - This profile MAY augment the definition of the Actions in the context of a SBOM
  - This profile SHALL NOT define Actions in a manner that is inconsistent with version 1.0 of the OpenC2 Language Specification
- Target: A subset of the Targets and Target-Specifiers defined in Version 1.0 of the OpenC2 Language Specification that are meaningful in the context of SBOM and one Target (and its associated Specifier) that is defined in this specification
- Arguments: A subset of the Arguments defined in the Language Specification and a set of Arguments defined in this specification
- Actuator: A set of specifiers defined in this specification that are meaningful in the context of SBOM

2.1.1 Actions

Table 2.1.1-1 presents the OpenC2 Actions defined in version 1.0 of the Language Specification which are meaningful in the context of an SBOM. The particular Action/Target pairs that are required or are optional are presented in Section 2.3.

Table 2.1.1-1. Actions Applicable to SBOM

Type: Action (Enumerated)
### 2.1.2 Targets

Table 2.1.2-1 summarizes the Targets defined in Version 1.0 of the [OpenC2-Lang-v1.0] as they relate to SBOM functionality. Table 2.1.2-2 summarizes the Targets that are defined in this specification.

#### 2.1.2.1 Common Targets

Table 2.1.2-1 lists the Targets defined in the OpenC2 Language Specification that are applicable to SBOM. The particular Action/Target pairs that are required or are optional are presented in Section 2.3.

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>features</td>
<td>Features</td>
<td>A set of items such as Action/Target pairs, profiles versions, options that are supported by the Actuator. The Target is used with the query Action to determine an Actuator’s capabilities</td>
</tr>
<tr>
<td>10</td>
<td>sbom</td>
<td>Sbom</td>
<td>Properties of a SBOM</td>
</tr>
</tbody>
</table>

The semantics/requirements as they pertain to common targets:

- fill in if we have any

#### 2.1.2.2 SBOM Targets

The list of common Targets is extended to include the additional Targets defined in this section and referenced with the SBOM namespace.

Table 2.1.2-2. Targets Unique to SBOM

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1024</td>
<td>fillin</td>
<td>Rule-ID</td>
<td>Immutable identifier assigned when a rule is created. Identifies a rule to be deleted</td>
</tr>
</tbody>
</table>

update per https://github.com/oasis-tcs/openc2-usecases/tree/master/Cybercom-Plugfest/TestData/sbom

### 2.1.3 Command Arguments
Arguments provide additional precision to a Command by including information such as how, when, or where a Command is to be executed. Table 2.1.3-1 summarizes the Command Arguments defined in Version 1.0 of the [OpenC2-Lang-v1.0] as they relate to SBOM functionality. Table 2.1.3-2 summarizes the Command Arguments that are defined in this specification.

### 2.1.3.1 Common Arguments

Table 2.1.3-1 lists the Command Arguments defined in the [OpenC2-Lang-v1.0] that are applicable to SBOM.

#### Table 2.1.3-1. Command Arguments applicable to SBOM

*Type: Args (Map)*

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Type</th>
<th>#</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>response_requested</td>
<td>Response-Type</td>
<td>0..1</td>
<td>The type of Response required for the Action: none, ack, status, complete</td>
</tr>
</tbody>
</table>

Proposal to delete "none" as it doesn't make sense on either Query.

### 2.1.3.2 SBOM Arguments

The list of common Command Arguments is extended to include the additional Command Arguments defined in this section and referenced with the SBOM namespace.

#### Table 2.1.3-2. Command Arguments Unique to SBOM

*Type: Args (Map)*

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Type</th>
<th>#</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>N</td>
<td>N</td>
<td>0..1</td>
<td>need to add the sbom formats per plugfest examples</td>
</tr>
</tbody>
</table>

The semantics/requirements as they relate to SBOM arguments:

- fill in about the choice of formats and list order is preference order

### 2.1.4 Actuator Specifiers

An Actuator is the entity that provides the functionality and performs the Action. The Actuator executes the Action on the Target. In the context of this profile, the Actuator is the SBOM and the presence of one or more Specifiers further refine which Actuator(s) shall execute the Action.

Table 2.1.4-1 lists the Specifiers that are applicable to the SBOM Actuator. Annex A provides sample Commands with the use of Specifiers.

The Actuator Specifiers defined in this document are referenced under the SBOM namespace.

#### Table 2.1.4-1. SBOM Specifiers

*Type: Specifiers (Map)*
### 2.2 OpenC2 Response Components

Response messages originate from the Actuator as a result of a Command.

Responses associated with required Actions MUST be implemented. Implementations that include optional Actions MUST implement the RESPONSE associated with the implemented Action. Additional details regarding the Command and associated Response are captured in Section 2.3. Examples are provided in Annex A.

#### 2.2.1 Common Results

Table 2.2.1-1 lists the Response Results properties defined in the [OpenC2-Lang-v1.0](https://example.com) that are applicable to SBOM.

#### Table 2.2.1-1. Response Results Applicable to SBOM

*Type: Results (Map [1..*])*

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Type</th>
<th>#</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>versions</td>
<td>Version</td>
<td>0..*</td>
<td>List of OpenC2 language versions supported by this Actuator</td>
</tr>
<tr>
<td>2</td>
<td>profiles</td>
<td>ArrayOf(Nsid)</td>
<td>0..1</td>
<td>List of profiles supported by this Actuator</td>
</tr>
<tr>
<td>3</td>
<td>pairs</td>
<td>Action-Targets</td>
<td>0..*</td>
<td>List of targets applicable to each supported Action</td>
</tr>
<tr>
<td>4</td>
<td>rate_limit</td>
<td>Number</td>
<td>0..1</td>
<td>Maximum number of requests per minute supported by design or policy</td>
</tr>
</tbody>
</table>

#### 2.2.2 SBOM Results

The list of common Response properties is extended to include the additional Response properties defined in this section and referenced with the SBOM namespace.

#### Table 2.2.2-1. SBOM Results

*Type: OpenC2-Response (Map)*
### 2.2.3 Response Status Codes

Table 2.2.1-2 lists the Response Status Codes defined in the OpenC2 Language Specification that are applicable to SBOM.

**Table 2.2.1-2. Response Status Codes**

*Type: Status-Code (Enumerated.ID)*

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>102</td>
<td>Processing. Command received but action not necessarily complete.</td>
</tr>
<tr>
<td>200</td>
<td>OK.</td>
</tr>
<tr>
<td>400</td>
<td>Bad Request. Unable to process Command, parsing error.</td>
</tr>
</tbody>
</table>
| 500   | Internal Error. For "response_requested" value "complete", one of the following MAY apply:  
       * Cannot access file or path  
       * Rule number currently in use  
       * Rule not updated |
| 501   | Not implemented. For "response_requested" value "complete", one of the following MAY apply:  
       * Target not supported  
       * Option not supported  
       * Command not supported |

### 2.3 OpenC2 Commands

An OpenC2 Command consists of an Action/Target pair and associated Specifiers and Arguments. This section enumerates the allowed Commands and presents the associated Responses.

Table 2.3-1 defines the Commands that are valid in the context of the SBOM profile. An Action (the top row in Table 2.3-1) paired with a Target (the first column in Table 2.3-1) defines a valid Command. The subsequent subsections provide the property tables applicable to each OpenC2 Command.

**Table 2.3-1. Command Matrix**

<table>
<thead>
<tr>
<th></th>
<th>Allow</th>
<th>Deny</th>
<th>Query</th>
<th>Delete</th>
<th>Update</th>
</tr>
</thead>
<tbody>
<tr>
<td>features</td>
<td></td>
<td></td>
<td>valid</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sbom</td>
<td></td>
<td></td>
<td>valid</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 2.3-2 defines the Command Arguments that are allowed for a particular Command by the SBOM profile. A Command (the top row in Table 2.3-2) paired with an Argument (the first column in Table 2.3-2) defines an allowable combination. The subsection identified at the intersection of the Command/Argument provides details applicable to each Command as influenced by the Argument.

Table 2.3-2. Command Arguments Matrix

<table>
<thead>
<tr>
<th>Allow target</th>
<th>Deny target</th>
<th>Query features</th>
<th>Delete SBOM:rule_number</th>
<th>Update file</th>
</tr>
</thead>
<tbody>
<tr>
<td>response_requested</td>
<td>2.3.1</td>
<td>2.3.2</td>
<td>2.3.3.1</td>
<td>2.3.4.1</td>
</tr>
<tr>
<td>add stuff for sbom</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2.3.1 Query

The valid Target type, associated Specifiers, and Options are summarized in Section 2.3.1. Sample Commands are presented in Annex A.

2.3.3.1 Query features

The 'query features' Command MUST be implemented in accordance with Version 1.0 of the [OpenC2-Lang-v1.0].

2.3.3.1 Query sbom

The 'query sbom' Command MUST be implemented in accordance with fill-in-here

Refer to Annex A for sample Commands.

3 Conformance statements

This section is normative. This section identifies the requirements for twenty-two conformance profiles as they pertain to two conformance targets. The two conformance targets are OpenC2 Producers and OpenC2 Consumers (as defined in Section 1.8 of this specification).

3.1 Clauses Pertaining to the OpenC2 Producer Conformance Target

All OpenC2 Producers that are conformant to this specification MUST satisfy Conformance Clause 1 and MAY satisfy one or more of Conformance Clauses 2 through 11.

3.1.1 Conformance Clause 1: Baseline OpenC2 Producer

An OpenC2 Producer satisfies Baseline OpenC2 Producer conformance if:

- 3.1.1.1 MUST support JSON serialization of OpenC2 Commands that are syntactically valid in accordance with the property tables presented in Section 2.1
- 3.1.1.2 All serializations MUST be implemented in a manner such that the serialization validates against and provides a one-to-one mapping to the property tables in Section 2.1 of this specification
3.1.1.3 **MUST** support the use of a Transfer Specification that is capable of delivering authenticated, ordered, lossless and uniquely identified OpenC2 messages

3.1.1.4 **SHOULD** support the use of one or more published OpenC2 Transfer Specifications which identify underlying transport protocols such that an authenticated, ordered, lossless, delivery of uniquely identified OpenC2 messages is provided as referenced in Section 1 of this specification

3.1.1.5 **MUST** be conformant with Version 1.0 of the OpenC2 Language Specification

3.1.1.6 **MUST** implement the 'query features' Command in accordance with the normative text provided in Version 1.0 of the OpenC2 Language Specification

3.1.1.7 **MUST** implement the 'response_requested' Command Argument as a valid option for any Command

3.1.1.8 **MUST** conform to at least one of the following conformance clauses in this specification:
   - Conformance Clause 2
   - Conformance Clause 3
   - Conformance Clause 4
   - Conformance Clause 5

### 3.2 Clauses Pertaining to the OpenC2 Consumer Conformance Target

All OpenC2 Consumers that are conformant to this specification **MUST** satisfy Conformance Clause 12 and **MAY** satisfy one or more of Conformance Clauses 13 through 22.

#### 3.2.1 Conformance Clause 12: Baseline OpenC2 Consumer

An OpenC2 Consumer satisfies Baseline OpenC2 Consumer conformance if:

3.2.1.1 **MUST** support JSON serialization of OpenC2 Commands that are syntactically valid in accordance with the property tables presented in Section 2.1

3.2.1.2 All serializations **MUST** be implemented in a manner such that the serialization validates against and provides a one-to-one mapping to the property tables in Section 2.1 of this specification

3.2.1.3 **MUST** support the use of a Transfer Specification that is capable of delivering authenticated, ordered, lossless and uniquely identified OpenC2 messages

3.2.1.4 **SHOULD** support the use of one or more published OpenC2 Transfer Specifications which identify underlying transport protocols such that an authenticated, ordered, lossless, delivery of uniquely identified OpenC2 messages is provided as referenced in Section 1 of this specification

3.2.1.5 **MUST** be conformant with Version 1.0 of the OpenC2 Language Specification

3.2.1.6 **MUST** implement the 'query features' Command in accordance with the normative text provided in version 1.0 of the OpenC2 Language Specification

3.2.1.7 **MUST** implement the 'response_requested' Command Argument as a valid option for any Command
   - 3.2.1.7.1 All Commands received with a 'response_requested' argument set to 'none' **MUST** process the Command and **MUST NOT** send a Response. This criteria supersedes all other normative text as it pertains to Responses
   - 3.2.1.7.2 All Commands received without the 'response_requested' argument **MUST** process the Command and Response in a manner that is consistent with "response_requested"."complete"

3.2.1.8 **MUST** conform to at least one of the following conformance clauses in this specification:
Annex A: Sample Commands

This section is non-normative

Editor’s note – update per https://github.com/oasis-tcs/openc2-usecases/tree/master/Cybercom-Plugfest/TestData/sbom

This section will summarize and provide examples of OpenC2 Commands as they pertain to SBOM firewalls. The sample Commands will be encoded in verbose JSON, however other encodings are possible provided the Command is validated against the property tables defined in Section 2 of this specification. Examples of corresponding Responses are provided where appropriate.

The samples provided in this section are for illustrative purposes only and are not to be interpreted as operational examples for actual systems.

The following examples include Binary fields which are serialized in Base64url format. The examples show JSON-serialized Commands; the conversion of Base64url serialized values to Binary data and String display text is:

<table>
<thead>
<tr>
<th>Base64url</th>
<th>Binary</th>
<th>Display String</th>
</tr>
</thead>
<tbody>
<tr>
<td>AQIDBA</td>
<td>01020304</td>
<td>1.2.3.4</td>
</tr>
<tr>
<td>xgIDBA</td>
<td>c6020304</td>
<td>198.2.3.4</td>
</tr>
<tr>
<td>xjNkEQ</td>
<td>c6336411</td>
<td>198.51.100.17</td>
</tr>
</tbody>
</table>

The examples include Integer Date-Time fields; the conversion of Integer values to String display text is:

<table>
<thead>
<tr>
<th>Integer</th>
<th>Display String</th>
</tr>
</thead>
<tbody>
<tr>
<td>1534775460000</td>
<td>Monday, August 20, 2018 2:31:00 PM GMT, 2018-08-20T10:31:00-04:00</td>
</tr>
</tbody>
</table>

A.1 Deny and Allow

Deny and allow can be treated as mathematical complements of each other. Unless otherwise stated, the example Targets, Specifiers, Arguments and corresponding Responses are applicable to both Actions.

A.1.1 Deny a particular connection

Block a particular connection within the domain and do not send a host unreachable. Note, the "SBOM":{"drop_process"} argument does not apply to the allow Action.
A.1.2 Deny all outbound ftp transfers

Block all outbound ftp data transfers, send false acknowledgment. Note that the five-tuple is incomplete. Note that the response_requested field was not populated therefore will be 'complete'. Also note that the Actuator called out was SBOM with no additional Specifiers, therefore all endpoints that can execute the Command should. Note, the "SBOM":{"drop_process"} argument does not apply to the allow Action.
Responses:

Case One: the Actuator successfully issued the deny.

{
    "status": 200
}

Case Two: the Command failed due to a syntax error in the Command. Optional status text is ignored by the Producer, but may be added to provide error details for debugging or logging.

{
    "status": 400,
    "status_text": "Validation Error: Target: ip_coneception"
}

Case Three: the Command failed because an Argument was not supported.

{
    "status": 501
}

A.1.3 Block all inbound traffic from a particular source.

Block all inbound traffic from the specified ipv6 network and do not respond. In this case the ipv6_net Target and the direction argument was used. In this case only the perimeter filters should update the rule.

Command:

{
    "action": "deny",
    "target": {
    },
    "args": {
        "response_requested": "none",
        "SBOM": {
            "direction": "ingress"
        }
    },
    "actuator": {
        "SBOM": {
...
A.1.4 Permit ftp transfers to a particular destination.

Permit ftp data transfers to 3ffe:1900:4545:3::f8ff:fe21:67cf from any source. (Note that an actual application would also need to allow ftp-data (port 20) in order for transfers to be permitted).

Command:

```json
{
    "action": "allow",
    "target": {
        "ipv6_connection": {
            "protocol": "tcp",
            "dst_addr": "3ffe:1900:4545:3::f8ff:fe21:67cf",
            "src_port": 21
        }
    },
    "actuator": {
        "SBOM": {}
    }
}
```

In this case the Actuator returned a rule number associated with the allow.

Response:

```json
{
    "status": 200,
    "results": {
        "SBOM": {
            "rule_number": 1234
        }
    }
}
```

A.2 Delete Rule

Used to remove a firewall rule rather than issue an allow or deny to counteract the effect of an existing rule. Implementation of the 'delete SBOM:rule_number' Command is OPTIONAL.

In this case the rule number assigned in a previous allow will be removed (refer to the final example in Annex A.1).

Command:

```json
{
    "action": "delete",
    "target": {
        "ipv6_connection": {
            "protocol": "tcp",
            "dst_addr": "3ffe:1900:4545:3::f8ff:fe21:67cf",
            "src_port": 21
        }
    },
    "actuator": {
        "SBOM": {}
    }
}
```
A.3 Update file

Implementation of the Update Action is optional. Update is intended for the device to process new configuration files. The update Action is a compound Action in that all of the steps required for a successful update (such as download the new file, install the file, reboot etc.) are implied. File is the only valid Target type for Update.

Instructs the firewalls to acquire a new configuration file. Note that all network based firewalls will install the new update because no particular firewall was identified. Host based firewalls will not act on this because network firewalls were identified as the Actuator.

Command:

```
{
    "action": "update",
    "target": {
        "file": {
            "path": "\\\someshared-drive\somedirectory\configurations",
            "name": "firewallconfiguration.txt"
        }
    },
    "actuator": {
        "SBOM": {
            "named_group": "network"
        }
    }
}
```

Responses:

Successful update of the configuration

```
{
    "status": 200
}
```

This Actuator does not support the update file Command

```
{
    "status": 501,
}
This Actuator could not access the file

```
{
  "status": 500,
  "status_text": "Server error, Cannot access file"
}
```

### A.4 Query features

Implementation of query Openc2 is required. The query features Command is intended to enable the Openc2 Producer to determine the capabilities of the Actuator. The query features Command can also be used to check the status of the Actuator.

#### A.4.1 No query items set

This Command uses query features with no query items to verify that the Actuator is functioning.

**Command:**

```
{
  "action": "query",
  "target": {
    "features": []
  }
}
```

**Response:**

The Actuator is alive.

```
{
  "status": 200
}
```

#### A.4.2 Version of Language specification supported

This Command queries the Actuator to determine which version(s) of the language specification are supported. The language specifications use semantic versioning ("major.minor"); for each supported major version the Actuator need only report the highest supported minor version.

**Command:**

```
{
  "action": "query",
  "target": {
    "features": ["versions"]
  }
}
```
Response:
The Actuator supports language specification versions 1.0 - 1.3.

```json
{
   "status": 200,
   "results": {
      "versions": ["1.0"]
   }
}
```

A.4.3 Actuator profiles supported

This Command queries the Actuator to determine both the language versions and the profiles supported.

Command:

```json
{
   "action": "query",
   "target": {
      "features": ["versions", "profiles"]
   }
}
```

Response:
The Actuator device is apparently a smart front-door-lock for which an extension profile has been written. The device supports both the standard SBOM functions and whatever Commands are defined in the extension profile.

```json
{
   "status": 200,
   "results": {
      "versions": ["1.3"],
      "profiles": ["SBOM", "iot-front-door-lock"]
   }
}
```

A.4.4 Specific Commands Supported

This Command queries the Actuator to determine which Action/Target pairs are supported. Not all Targets are meaningful in the context of a specific Action, and although a Command such as "update ipv4_connection" may be syntactically valid, the combination does not specify an operation supported by the Actuator.

Command:

For each supported Action list the Targets supported by this Actuator.
Response:
The Actuator supports all Action/Target pairs shown in Table 2.3-1 - Command Matrix.

```
{
    "status": 200,
    "results": {
        "pairs": {
            "allow": ["ipv6_net", "ipv6_connection"],
            "deny": ["ipv6_net", "ipv6_connection"],
            "query": ["features"],
            "delete": ["SBOM:rule_number"],
            "update": ["file"]
        }
    }
}
```

Annex B: Acronyms

This section is non-normative

<table>
<thead>
<tr>
<th>Term</th>
<th>Expansion</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPR</td>
<td>Intellectual Property Rights</td>
</tr>
<tr>
<td>JADN</td>
<td>JSON Abstract Data Notation</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>OASIS</td>
<td>Organization for the Advancement of Structured Information Standards</td>
</tr>
<tr>
<td>RFC</td>
<td>Request for Comment</td>
</tr>
<tr>
<td>TC</td>
<td>Technical Committee</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifier</td>
</tr>
</tbody>
</table>

Annex C: Acknowledgments

This section is non-normative
The Actuator Profile Subcommittee was tasked by the OASIS Open Command and Control Technical Committee (OpenC2 TC) which at the time of this submission, had TBD members. The editors wish to express their gratitude to the members of the OpenC2 TC.

The following individuals are acknowledged for providing comments, suggested text and/or participation in the SBOM CSD ballots:

- TBD

Annex D: Revision History

This section is non-normative

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Editor</th>
<th>Changes Made</th>
</tr>
</thead>
<tbody>
<tr>
<td>Working Draft 01</td>
<td>tbd</td>
<td>Sparrell</td>
<td>Initial draft</td>
</tr>
</tbody>
</table>