# CACAO Security Playbooks Version 1.0

## Committee Specification 02

## 23 June 2021

**This stage:**

https://docs.oasis-open.org/cacao/security-playbooks/v1.0/cs02/security-playbooks-v1.0-cs02.docx (Authoritative)
https://docs.oasis-open.org/cacao/security-playbooks/v1.0/cs02/security-playbooks-v1.0-cs02.html
https://docs.oasis-open.org/cacao/security-playbooks/v1.0/cs02/security-playbooks-v1.0-cs02.pdf

**Previous stage:**

https://docs.oasis-open.org/cacao/security-playbooks/v1.0/csd03/security-playbooks-v1.0-csd03.docx (Authoritative)
https://docs.oasis-open.org/cacao/security-playbooks/v1.0/csd03/security-playbooks-v1.0-csd03.html
https://docs.oasis-open.org/cacao/security-playbooks/v1.0/csd03/security-playbooks-v1.0-csd03.pdf

**Latest stage:**

https://docs.oasis-open.org/cacao/security-playbooks/v1.0/security-playbooks-v1.0.docx (Authoritative)
https://docs.oasis-open.org/cacao/security-playbooks/v1.0/security-playbooks-v1.0.html
https://docs.oasis-open.org/cacao/security-playbooks/v1.0/security-playbooks-v1.0.pdf

**Technical Committee:**

OASIS Collaborative Automated Course of Action Operations (CACAO) for Cyber Security TC

**Chairs:**

Bret Jordan (jordan.oasisopen@gmail.com), Individual
Allan Thomson (atcyber1000@gmail.com), Individual

**Editors:**

Bret Jordan (jordan.oasisopen@gmail.com), Individual
Allan Thomson (atcyber1000@gmail.com), Individual

**Related Work:**

This document is related to:

- *Playbook Requirements Version 1.0*. Edited by Bret Jordan and Allan Thomson. 01 April 2020. Latest version: https://docs.oasis-open.org/cacao/playbook-requirements/v1.0/playbook-requirements-v1.0.html.
- CACAO Introduction Version 01. Edited by Bret Jordan, Allan Thomson, and Jyoti Verma. Latest version: https://tools.ietf.org/html/draft-jordan-cacao-introduction-01.

### Abstract:

To defend against threat actors and their tactics, techniques, and procedures organizations need to identify, create, document, and test detection, investigation, prevention, mitigation, and remediation steps. These steps, when grouped together form a cyber security playbook that can be used to protect organizational systems, networks, data, and users.

This specification defines the schema and taxonomy for collaborative automated course of action operations (CACAO) security playbooks and how these playbooks can be created, documented, and shared in a structured and standardized way across organizational boundaries and technological solutions.


### Status:

This document was last revised or approved by the OASIS Collaborative Automated Course of Action Operations (CACAO) for Cyber Security TC on the above date. The level of approval is also listed above. Check the "Latest stage" 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=cacao#technical.

TC members should send comments on this document 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/cacao/.



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.

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

### Citation format:

When referencing this document, the following citation format should be used:

**[CACAO-Security-Playbooks-v1.0]**

*CACAO Security Playbooks Version 1.0*. Edited by Bret Jordan and Allan Thomson. 23 June 2021. OASIS Committee Specification 02. https://docs.oasis-open.org/cacao/security-playbooks/v1.0/cs02/security-playbooks-v1.0-cs02.html. Latest stage: https://docs.oasis-open.org/cacao/security-playbooks/v1.0/security-playbooks-v1.0.html.

### Notices:

# Table of Contents

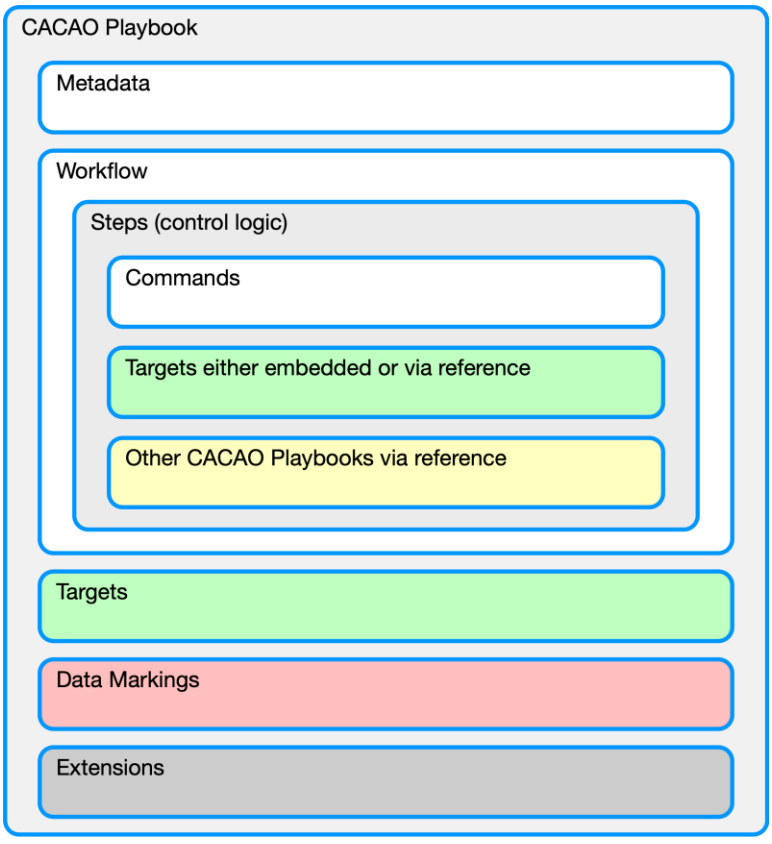
# 1 Introduction

To defend against threat actors and their tactics, techniques, and procedures organizations need to identify, create, document, and test detection, investigation, prevention, mitigation, and remediation steps. These steps, when grouped together form a cyber security playbook that can be used to protect organizational systems, networks, data, and users.

This specification defines the schema and taxonomy for collaborative automated course of action operations (CACAO) security playbooks and how these playbooks can be created, documented, and shared in a structured and standardized way across organizational boundaries and technological solutions.

## 1.1 Overview of Structure and Object Types

This specification defines the following classes of objects: playbooks (section 3), workflow steps (section 4), commands (section 5), targets (section 6), extensions (section 7), and data markings (section 8).



## 1.2 Playbook

A CACAO playbook is a workflow for security orchestration containing a set of steps to be performed based on a logical process and may be triggered by an automated or manual event or observation. A playbook provides guidance on how to address a certain security event, incident, problem, attack, or compromise. A playbook may be defined in one system by one or more authors, but the playbook may be executed in an operational environment where the systems and users of those systems have different authentication and authorizations. A playbook may also reference or include other playbooks in such a manner that allows composition from smaller, more specific functional playbooks similar to how software application development leverages modular libraries of common functions shared across different applications.

## 1.3 Executable Playbook

An executable playbook is intended to be immediately actionable in an organization's security infrastructure without requiring modification or updates to the workflow and commands.

## 1.4 Playbook Template

A playbook template provides examples of actions related to a particular security incident, malware, vulnerability or other security operation. A template playbook will not be immediately executable by a receiving organization but may inform their own executable playbook for their specific environment or organization.

## 1.5 Integrations

To enable integration within existing tools, CACAO security playbooks can reference and be referenced by other cybersecurity operational tools, including systems that may support cyber threat intelligence (CTI). This enables organizations to not only know and understand threats, behaviors, and associated intelligence, but also know what they could potentially do in response to a threat or behavior.

## 1.6 Related Standards

In some cases this specification may define references to schemas or constructs from other standards. This allows CACAO to use other standards without having to redefine those schemas or constructs within CACAO itself.

## 1.7 Vocabularies

Some properties in this specification use defined vocabularies. These vocabularies can be either open or closed. An open vocabulary allows implementers to use additional values beyond what is currently defined in the specification. However, if a similar value is already in the vocabulary, that value **MUST** be used. A closed vocabulary is effectively an enumeration and **MUST** be used as defined.

Vocabularies defined in this specification enhance interoperability by increasing the likelihood that different entities use the exact same string to represent the same concept, thereby making comparison and correlation easier.

## 1.8 Document Conventions

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

- The `Consolas` font is used for all type names, property names and literals.
  - type names are in red with a light red background – `string`
  - property names are in bold style – **`description`**
  - literals (values) are in blue with a blue background – `investigation`
- In a property table, if a common property is being redefined in some way, then the background is dark grey.
- All examples in this document are expressed in JSON. They are in `Consolas 9-point font`, with straight quotes, black text and a light grey background, and using 2-space indentation. JSON examples in this document are representations of JSON objects [RFC8259]. They should not be interpreted as string literals. The ordering of 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 the ellipses (...).
- The term "hyphen" is used throughout this document to refer to the ASCII hyphen or minus character, which in Unicode is "hyphen-minus", U+002D.

## 1.9 Changes From Earlier Versions

N/A

## 1.10 Glossary

**CACAO** - Collaborative Automated Course of Action Operations
**CTI** - Cyber Threat Intelligence
**JSON** - JavaScript Object Notation as defined in [RFC7493] and [RFC8259]
**MTI** - Mandatory To Implement
**STIX** - Structured Threat Information Expression
**TLP** - Traffic Light Protocol

# 2 Core Concepts

CACAO standardizes the definition and use of two important concepts often used by organizations protecting themselves or the broader ecosystem they connect with.

- Action
  - The first concept represents every security activity in an organization referred to as a **security action**, or just **action.** Those actions may represent an activity to investigate, prevent, mitigate or remediate a specific security state that has either occurred or the organization is taking action to ensure the security state never occurs.
- Playbook
  - The second concept introduced by CACAO, defines a **playbook** consisting of one or more security actions combined into a sequence or algorithmically-defined use.

Actions typically represent a single security operation, whereas playbooks may represent multiple security operations that cover multiple aspects of the security lifecycle of the organization.

## 2.2 Playbook Types

This section defines the playbook types that are used in this specification.

### 2.2.1 Notification Playbook

A playbook that is primarily focused on the orchestration steps required to notify and disseminate information and other playbooks about a security event, incident, or other threat. For example, a notification playbook can be used to notify multiple entities about an attack and disseminate other playbooks to detect and mitigate it as quickly as possible.

### 2.2.2 Detection Playbook

A playbook that is primarily focused on the orchestration steps required to detect a known security event, other known or expected security-relevant activity, or for threat hunting.

### 2.2.3 Investigation Playbook

A playbook that is primarily focused on the orchestration steps required to investigate what a security event, incident, or other security-relevant activity has caused. Investigation playbooks will likely inform other subsequent actions upon completion of the investigation.

### 2.2.4 Prevention Playbook

A playbook that is primarily focused on the orchestration steps required to prevent a known or expected security event, incident, or threat from occurring. Prevention playbooks are often designed and deployed as part of best practices to safeguard organizations from known and perceived threats and behaviors associated with suspicious activity.

### 2.2.5 Mitigation Playbook

A playbook that is primarily focused on the orchestration steps required to mitigate a security event or incident that has occurred when remediation is not initially possible. Organizations often choose to mitigate a security event or incident until they can actually remediate it. Mitigation playbooks are designed to reduce or limit the impact of suspicious or confirmed malicious activity. For example, a mitigation playbook can be used to quarantine affected users/devices/applications from the network temporarily to prevent additional problems. Mitigation usually precedes remediation, after which the mitigation actions are reversed.

### 2.2.6 Remediation Playbook

A playbook that is primarily focused on the orchestration steps required to remediate, resolve, or fix the resultant state of a security event or incident, and return the system, device, or network back to a nominal operating state. Remediation playbooks can fix affected assets by selectively correcting problems due to malicious activity by reverting the system or network to a known good state.

### 2.2.7 Attack Playbook

A playbook that is primarily focused on the orchestration steps required to execute a penetration test or attack simulation to test or verify security controls or identify vulnerabilities within an organization's environment. This is often represented by a penetration test that is used to verify how security systems or other systems respond to various aspects of the test or attack.

## 2.3 Playbook Creator

The playbook creator is the entity (e.g., person, system, organization, or instance of a tool) that generates the identifier for the **`id`** property of the playbook. Playbook creators are represented as STIX 2.1 [STIX-v2.1] Identity objects. The creator's ID is captured in the **`created_by`** property. If that property is omitted, the creator is either unknown or wishes to remain anonymous.

Entities that re-publish an object from another entity without making any changes to the object, and thus maintaining the original **`id`**, are not considered the object creator and **MUST NO**T change the **`created_by`** property. An entity that accepts objects and republishes them with modifications, additions, or omissions **MUST** create a new **`id`** for the object as they are now considered the object creator of the new object for purposes of versioning.

## 2.4 Versioning

Versioning is the mechanism that playbook creators use to manage a playbook's lifecycle, including when it is created, updated, or revoked. This section describes the versioning process and normative rules for performing versioning and revocation. Playbooks are versioned using the **`created`**, **`modified`**, and **`revoked`** properties (see section 3.1).

Playbooks **MAY** be versioned in order to update, add, or remove information. A version of a playbook is identified uniquely by the combination of its **`id`** and **`modified`** properties. The first version of a playbook **MUST** have the same timestamp for both the **`created`** and **`modified`** properties. More recent values of the **`modified`** property indicate later versions of the playbook. Implementations **MUST** consider the version of the playbook with the most recent **`modified`** value to be the most recent version of the

playbook. For every new version of a playbook, the **modified** property **MUST** be updated to represent the time that the new version was created. This specification does not define how to handle a consumer receiving two objects that are different, but have the same **id** and **modified** timestamp. This specification does not address how implementations should handle versions of the object that are not current.

Playbooks have a single object creator, the entity that generates the **id** for the object and creates the first version. The object creator **SHOULD** (but not necessarily will) be identified in the **created_by** property of the object. Only the object creator is permitted to create new versions of a playbook. Producers other than the object creator **MUST NOT** create new versions of that object using the same **id**. If a producer other than the object creator wishes to create a new version, they **MUST** instead create a new playbook with a new **id**. They **SHOULD** additionally populate the **derived-from** property to relate their new playbook to the original playbook that it was derived from.

Every representation (each time the object version is serialized and shared) of a version of a playbook (identified by the playbook's **id** and **modified** properties) **MUST** always have the same set of properties and the same values for each property. If a property has the same value as the default, it **MAY** be omitted from a representation, and this does not represent a change to the object. In order to change the value of any property, or to add or remove properties, the **modified** property **MUST** be updated with the time of the change to indicate a new version.

Playbooks can also be revoked, which means that they are no longer considered valid by the object creator. As with issuing a new version, only the object creator is permitted to revoke a playbook. A value of `true` in the **revoked** property indicates that a playbook (including the current version and all past versions) has been revoked. Revocation is permanent. Once an object is marked as revoked, later versions of that object **MUST NOT** be created. Changing the **revoked** property to indicate that an object is revoked is an update to the object, and therefore its **modified** property **MUST** be updated at the same time. This specification does not address how implementations should handle revoked data.

### 2.4.1 Versioning Timestamps

There are two timestamp properties used to indicate when playbooks were created and modified: **created** and **modified**. The **created** property indicates the time the first version of the playbook was created. The **modified** property indicates the time the specific version of the playbook was updated. The **modified** time **MUST NOT** be earlier than the **created** time. This specification does not address the specifics of how implementations should determine the value of the creation and modification times for use in the **created** and **modified** properties (e.g., one system might use when the playbook is first added to the local database as the creation time, while another might use the time when the playbook is first distributed).

### 2.4.2 New Version or New Object?

Eventually an implementation will encounter a case where a decision must be made regarding whether a change is a new version of an existing playbook or is different enough that it is a new playbook. This is generally considered a data quality problem and therefore this specification does not provide any normative text.

However, to assist implementers and promote consistency across implementations, some general rules are provided. Any time a change indicates a material change to the meaning of the playbook, a new

playbook with a different **`id`** **SHOULD** be used. A material change is any change that the playbook creator believes substantively changes the meaning or functionality of the playbook. These decisions are always made by the playbook creator. The playbook creator should also think about relationships to the playbook from other data when deciding if a change is material. If the change would invalidate the usefulness of relationships to the playbook, then the change is considered material and a new playbook **`id`** **SHOULD** be used.

## 2.5 Data Markings

Data markings represent restrictions, permissions, and other guidance for how playbooks can be used and shared. For example, playbooks may be shared with the restriction that it must not be re-shared, or that it must be encrypted at rest. In CACAO, data markings are specified using the data marking object and are applied via the **`markings`** property on the playbook object. These markings apply to all objects and elements included in the playbook.

Changes to the **`markings`** property (and therefore the markings applied to the object) are treated the same as changes to any other properties on the object and follow the same rules for versioning.

Multiple markings can be added to the same playbook. Some data markings or trust groups have rules about which markings override other markings or which markings can be additive to other markings. This specification does not define rules for how multiple markings applied to the same playbook should be interpreted.

## 2.6 Signing Playbooks

The process of signing CACAO Playbook data is similar to what is defined in JWS Clear Text JSON Signature Option (JWS/CT) and uses the concept of detached mode as described in JWS [RFC7515] Appendix F just without the JWS [RFC7515] header. The CACAO signature design supports both including the signature in the playbook itself and storing or releasing the signature separately as a detached signature. When signing a CACAO Playbook, the signer **MUST NOT** include any existing signatures in the playbook data, meaning no counter signing. Adding a signature to a playbook does not constitute a revision or change to the playbook and as such, the **`modified`** `timestamp` **MUST NOT** be updated. See section A.2 in the appendix for a detailed example.

### 2.6.1 Requirements

The algorithm used for creating CACAO digital signatures **MUST** come from one of the following options that are defined in JWA [RFC7518] section 3.1 and [RFC8037] section 3.1 and **SHOULD** be either `ES256` or `RS256` as defined in JWA.

`RS256`, `RS384`, `RS512`, `ES256`, `ES384`, `ES512`, `PS256`, `PS384`, `PS512`, `Ed25519`, `Ed448`

NOTE: Unlike RFC8037 [RFC8037] this specification requires explicit Ed* algorithm names instead of "EdDSA".

While JWA [RFC7518] section 3.1 defines the following symmetric algorithms: `HS256`, `HS384`, `HS512`, those algorithms **MUST NOT** be used as CACAO playbooks are intended to be shared across systems and organizational boundaries that would not allow the sharing of symmetric keys.

### 2.6.2 Signing Steps

The steps involved in signing a CACAO playbook are as follows (see Appendix A.2 for more details):

Step 1.0: Create or receive a JSON playbook object to sign
Step 1.1: Remove existing signature objects contained in the playbook's signatures property before computing the hash
Step 1.2: Create JCS [RFC8785] canonical version of the playbook from step 1.1
Step 1.3: Create SHA256 (in hex) of canonical version of playbook from step 1.2
Step 1.4: Create base64URL.encoded version of the SHA256 hash from step 1.3 and remove any padding
Step 2.0: Create a signature object and set the SHA256 string property to the string value of the b64 hash of the playbook from step 1.4
Step 2.1: Create JCS canonical version of signature from step 2.0
Step 2.2: Create base64URL.encoded version of the JCS signature from step 2.1
Step 3.0: Sign the data from step 2.2 using the algorithm defined in the signature object and base64URL.encode it.
Step 4.0: Append the new b64 digital signature from step 3.0 to the signatures property (with existing signatures, if any) of the playbook itself.

In pseudo code this would look like:

- PlaybookHash = base64URL.encode(sha256(jcs(<playbook data - existing signatures>)))
- SignatureObject = Create signature object and set the SHA256 string value to PlaybookHash
- DigitalSignature = base64URL.encode(sign(base64URL.encode(jcs(<SignatureObject>))))
- Playbook Data = Append DigitalSignature to Playbook.Signatures[]

# 3 Playbooks

CACAO playbooks are made up of five parts; playbook metadata, the workflow logic, a list of targets, a list of extensions, and a list of data markings. Playbooks **MAY** refer to other playbooks in the workflow, similar to how programs refer to function calls or modules that comprise the program.

## 3.1 Playbook Properties

| Property Name | Rq | Data Type | Details |
|---|---|---|---|
| **type** | Y | `string` | The value of this property **MUST** be `playbook` or `playbook-template`. See section 1.2, section 1.3, and section 1.4 respectively, for information about executable playbooks and playbook templates. |
| **spec_version** | Y | `string` | The version of the specification used to represent this playbook. The value of this property **MUST** be "1.0" to represent the version of this specification. |
| **id** | Y | `identifier` | A value that uniquely identifies the playbook. All playbooks with the same id are considered different versions of the same playbook and the version of the playbook is identified by its modified property. |
| **name** | Y | `string` | A simple name for this playbook. This name is not guaranteed or required to be unique. |
| **description** | | `string` | More details, context, and possibly an explanation about what this playbook does and tries to accomplish. Producers **SHOULD** populate this property. |
| **playbook_types** | Y | `list` of type `string` | A list of playbook types that specifies the operational functions this playbook addresses.<br><br>The values for this property **MUST** come from the `playbook-type` vocabulary. |
| **created_by** | Y | `identifier` | An ID that represents the entity that created this playbook. The ID **MUST** represent a STIX 2.1+ identity object. |
| **created** | Y | `timestamp` | The time at which this playbook was originally created. The creator can use any time it deems most appropriate as the time the playbook was created, but it **MUST** be precise to the nearest millisecond (exactly three digits after the decimal place in seconds). The created property **MUST NOT** be changed when creating a new version of the object. |
| **modified** | Y | `timestamp` | The time that this particular version of the playbook |

| | | | |
|---|---|---|---|
| | | | was last modified. The creator can use any time it deems most appropriate as the time that this version of the playbook was modified, but it **MUST** be precise to the nearest millisecond (exactly three digits after the decimal place in seconds). The modified property **MUST** be later than or equal to the value of the created property. |
| **revoked** | | `boolean` | A boolean that identifies if the playbook creator deems that this playbook is no longer valid. The default value is "false". |
| **valid_from** | | `timestamp` | The time from which this playbook is considered valid and the steps that it contains can be executed. More detailed information about time frames **MAY** be applied in the **workflow**.<br><br>If omitted, the playbook is valid at all times or until the timestamp defined by **valid_until**. |
| **valid_until** | | `timestamp` | The time at which this playbook should no longer be considered a valid playbook to be executed.<br><br>If the **valid_until** property is omitted, then there is no constraint on the latest time for which the playbook is valid.<br><br>This property **MUST** be greater than the timestamp in the **valid_from** property if the **valid_from** property is defined. |
| **derived_from** | | `list` of type `identifier` | The ID of one or more playbooks that this playbook was derived from.<br><br>The ID **MUST** represent a CACAO playbook object. |
| **priority** | | `integer` | A positive integer that represents the priority of this playbook relative to other defined playbooks.<br><br>Priority is a subjective assessment by the producer based on the context in which the playbook can be shared. Marketplaces and sharing organizations **MAY** define rules on how priority should be assessed and assigned. This property is primarily to allow such usage without requiring addition of a custom field for such practices.<br><br>If specified, the value of this property **MUST** be between 0 and 100.<br><br>When left blank this means unspecified. A value of 0 means specifically undefined. Values range from 1, the highest priority, to a value of 100, the lowest. |
| **severity** | | `integer` | A positive integer that represents the seriousness of |

| | | | |
|---|---|---|---|
| | | | the conditions that this playbook addresses. This is highly dependent on whether it's an incident (in which cases the severity can be mapped to the incident category) or a response to a threat (in which case the severity would likely be mapped to the severity of threat faced or captured by threat intelligence).<br><br>Marketplaces and sharing organizations **MAY** define additional rules for how this property should be assigned.<br><br>If specified, the value of this property **MUST** be between 0 and 100.<br><br>When left blank this means unspecified. A value of 0 means specifically undefined. Values range from 1, the lowest severity, to a value of 100, the highest. |
| **impact** | | `integer` | A positive integer that represents the impact the *playbook* has on the organization, not what triggered the playbook in the 1st place such as a threat or an incident. For example, a purely investigative playbook that is non-invasive would have a low impact value (1) whereas a playbook that makes firewall changes, IPS changes, moves laptops to quarantine....etc would have a higher impact value. If specified, the value of this property **MUST** be between 0 and 100.<br><br>When left blank this means unspecified. A value of 0 means specifically undefined. Values range from 1, the lowest impact, to a value of 100, the highest. |
| **labels** | | `list` of type `string` | An optional set of terms, labels, or tags associated with this playbook. The values may be user, organization, or trust-group defined and their meaning is outside the scope of this specification. |
| **external_references** | | `list` of type `external-reference` | An optional list of external references for this playbook or content found in this playbook. |
| **features** | | `playbook-features` | An optional property that contains a list of features that are enabled for this playbook. |
| **markings** | | `list` of type `identifier` | An optional list of data marking objects that apply to this playbook. In some cases, though uncommon, data markings themselves may be marked with sharing or handling guidance. In this case, this property **MUST NOT** contain any references to the same data marking object (i.e., it cannot contain any circular references).<br><br>The IDs **MUST** represent a CACAO data marking |

| | | | |
|---|---|---|---|
| | | | object. |
| **playbook_variables** | | `dictionary` of type `variable` | This property contains the variables that can be used within this playbook or within workflow steps, commands, and targets defined within this playbook. See section 9.13 for information about referencing variables.<br><br>The key for each entry in the dictionary **MUST** be a string that uniquely identifies the variable. The value for each key **MUST** be a CACAO `variable` data type (see section 9.13). |
| **workflow_start** | | `identifier` | The first workflow step included in the **`workflow`** property that **MUST** be executed when starting the workflow.<br><br>The ID **MUST** represent a CACAO workflow step object. |
| **workflow_exception** | | `identifier` | The workflow step invoked whenever a playbook exception condition occurs.<br><br>The ID **MUST** represent a CACAO workflow step object. |
| **workflow** | | `dictionary` | The **`workflow`** property contains the processing logic for the playbook as workflow steps.<br><br>The key for each entry in the dictionary **MUST** be an `identifier` that uniquely identifies the workflow step. The id **MUST** use the object type of "step" (see section 9.8 for more information on identifiers). The value for each key **MUST** be a CACAO workflow step object (see section 4). |
| **targets** | | `dictionary` | A dictionary of targets that can be referenced from workflow steps found in the **`workflow`** property.<br><br>The key for each entry in the dictionary **MUST** be an `identifier` that uniquely identifies the target. The id **MUST** use the object type of "target" (see section 9.8 for more information on identifiers). The value for each key **MUST** be a CACAO target object (see section 6). |
| **extension_definitions** | | `dictionary` | A dictionary of extension definitions that are referenced from workflow steps found in the **`workflow`** property.<br><br>The key for each entry in the dictionary **MUST** be an `identifier` that uniquely identifies the extension. The id **MUST** use the object type of "extension" (see section 9.8 for more information on identifiers). The |

| | | | |
|---|---|---|---|
| | | | value for each key **MUST** be a CACAO extension object (see section 7). |
| **data_marking_definitions** | | `dictionary` | A dictionary of data marking definitions that can be referenced from the playbook found in the **`markings`** property.<br><br>The key for each entry in the dictionary **MUST** be an `identifier` that uniquely identifies the data marking. The id **MUST** use an object type of one of the data marking objects defined in section 8, for example "marking-statement" (see section 9.8 for more information on identifiers). The value for each key **MUST** be a CACAO data marking object (see section 8). |
| **signatures** | | `list` of type `signature` | An optional list of digital signatures for this playbook. Adding a signature to a playbook does not represent a version change of the playbook. See sections 2.6, 9.10, and A.2 in the appendix for a detailed example. |

**Example**

```
{
  "type": "playbook",
  "spec_version": "1.0",
  "id": "playbook--uuid1",
  "name": "Find Malware FuzzyPanda",
  "description": "This playbook will look for FuzzyPanda on the network and in a SIEM",
  "playbook_types": ["investigation"],
  "created_by": "identity--uuid2",
  "created": "2020-03-04T15:56:00.123456Z",
  "modified": "2020-03-04T15:56:00.123456Z",
  "revoked": false,
  "valid_from": "2020-03-04T15:56:00.123456Z",
  "valid_until": "2020-07-31T23:59:59.999999Z",
  "derived_from": ["playbook--uuid99"],
  "priority": 3,
  "severity": 70,
  "impact": 5,
  "labels": [ "malware", "fuzzypanda", "apt"],
  "external_references": [
    {
      "name": "ACME Security FuzzyPanda Report",
      "description": "ACME security review of FuzzyPanda 2021",
      "source": "ACME Security Company, Solutions for FuzzyPanda 2021, January 2021.
        Available online: hxxp://www[.]example[.]com/info/fuzzypanda2021.html",
      "url": "hxxp://www[.]example[.]com/info/fuzzypanda2020.html",
      "hash": "f92d8b0291653d8790907fe55c024e155e460eabb165038ace33bb7f2c1b9019",
      "external_id": "fuzzypanda 2021.01"
    }
  ],
"features": {
  "if_logic": true,
  "data_markings": true
},
  "markings": [
    "marking-statement--uuid0"
  ],
```

```
  "playbook_variables": {
    "$$data_exfil_site$$": {
      "type": "ipv4-addr",
      "description": "The IP address for the data exfiltration site",
      "value": "1.2.3.4",
      "constant": false
    }
  },
  "workflow_start": "step--uuid0",
  "workflow_exception": "step--uuid123",
  "workflow": { },
  "targets": { },
  "extension_definitions": { },
  "data_marking_definitions": { },
  "signatures": [ ]
}
```

## 3.2 Playbook Type Vocabulary

A playbook may be categorized as having multiple types defined from this vocabulary. These definitions are taken from section 2.2.

**Vocabulary Name:** `playbook-type`

| Vocabulary Value | Description |
|---|---|
| `notification` | See section 2.2.1 for an explanation |
| `detection` | See section 2.2.2 for an explanation. |
| `investigation` | See section 2.2.3 for an explanation. |
| `prevention` | See section 2.2.4 for an explanation. |
| `mitigation` | See section 2.2.5 for an explanation. |
| `remediation` | See section 2.2.6 for an explanation. |
| `attack` | See section 2.2.7 for an explanation. |

## 3.3 Playbook Constants & Variables

Each playbook has a set of constants and variables that **MAY** be used throughout the execution of a playbook and its associated workflow.

| Name | Description | Mutable | Type | Default Value |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| `$$LOCAL_TARGET$$` | A constant that defines a target is local to the machine instance executing the current playbook. | No | `string` | "local_target" |
| `$$ACTION_TIMEOUT$$` | A timeout variable in milliseconds that may be used to assign to a specific step timeout. Each specific step timeout may be assigned this value or a distinct value. The step's timeout is evaluated when it is executed and the timeout is used to determine when a step is no longer responsive. When a step is determined to no longer respond, the calling context should call the timeout-assigned step. | Yes | `integer` | 60000 milliseconds |
| `$$RETURN_CALLER$$` | This constant tells the executing program to return to the step that started the current branch.<br><br>NOTE: this is similar to rolling back the stack in a computer program. | No | `string` | "return_caller" |
| `$$RETURN_CALLER_ID$$` | This constant defines a step to call upon completion or failure of a sub-step. This is typically used with parallel steps that define a tree of sub-steps to execute. This constant tells the executing program exactly which step ID it **MUST** return to. | yes | `identifier` | |

# 4 Workflows

Workflows contain a series of steps that are stored in a dictionary, where the key is the step ID and the value is a workflow step. These workflow steps along with the associated commands form the building blocks for playbooks and are used to control the commands that need to be executed. Workflows process steps either sequentially, in parallel, or both depending on the type of steps required by the playbook. In addition to simple processing, workflow steps **MAY** also contain conditional and/or temporal operations to control the execution of the playbook.

Conditional processing means executing steps or commands after some sort of condition is met. Temporal processing means executing steps or commands either during a certain time window or after some period of time has passed.

This section defines the various workflow steps and how they may be used to define a playbook.

## 4.1 Workflow Step Common Properties

Each workflow step contains some base properties that are common across all steps. These common properties are defined in the following table.

| Property Name | Rq | Data Type | Details |
|---|---|---|---|
| **type** | Y | `string` | The type of workflow step being used.<br><br>The value for this property **MUST** come from the `workflow-step-type` vocabulary. |
| **name** | | `string` | A name for this step that is meant to be displayed in a user interface or captured in a log message. |
| **description** | | `string` | More details, context, and possibly an explanation about what this step does and tries to accomplish. |
| **external_references** | | `list` of type `external-reference` | An optional list of external references for this step. |
| **delay** | | `integer` | The amount of time in milliseconds that this step **SHOULD** wait before it starts processing.<br><br>The integer **MUST** be a positive value greater than 0.<br><br>If this field is omitted, then the workflow step executes immediately without delay. |
| **timeout** | | `integer` | The amount of time in milliseconds that this step **MUST** wait before considering the step has failed. Upon timeout occurring for a step, the |

| | | | |
|---|---|---|---|
| | | | **on_failure** step pointer is invoked and the information included in that call states that an ACTION_TIMEOUT occurred including the id of the step that timed out.<br><br>If this field is omitted, the system executing this workflow step **SHOULD** consider implementing a maximum allowed timeout to ensure that no individual workflow step can block a playbook execution indefinitely. |
| **step_variables** | | `dictionary` of type `variable` | This property contains the variables that can be used within this workflow step or within commands and targets referenced by this workflow step. See section 9.13.2 for information about referencing variables.<br><br>The key for each entry in the dictionary **MUST** be a string that uniquely identifies the variable. The value for each key **MUST** be a CACAO `variable` data type (see section 9.13.3). |
| **owner** | | `identifier` | An ID that represents the entity that is assigned as the owner or responsible party for this step.<br><br>The ID **MUST** represent a STIX 2.1+ Identity object. |
| **on_completion** | | `identifier` | The ID of the next step to be processed upon completion of the defined commands.<br><br>The ID **MUST** represent either a CACAO workflow step object or a CACAO playbook object.<br><br>If this property is defined, then **on_success** and **on_failure** **MUST NOT** be defined. |
| **on_success** | | `identifier` | The ID of the next step to be processed if this step completes successfully.<br><br>The ID **MUST** represent either a CACAO workflow step object or a CACAO playbook object. |
| **on_failure** | | `identifier` | The ID of the next step to be processed if this step fails to complete successfully.<br><br>The ID **MUST** represent either a CACAO workflow step object or a CACAO playbook object. |

| | | | |
|---|---|---|---|
| | | | If omitted and a failure occurs, then the playbook's exception handler action step will be invoked. |
| **step_extensions** | | `dictionary` | This property defines the extensions that are in use on this step.<br><br>The key for each entry in the dictionary **MUST** be an `identifier` that uniquely identifies the extension. The id **MUST** use the object type of "extension" (see section 9.7 for more information on identifiers). The value for each key is a JSON object that can contain the structure as defined in the extension's schema location. |

## 4.2 Workflow Step Type Vocabulary

**Vocabulary Name:** `workflow-step-type`

This section defines the following types of workflow steps.

| Workflow Step Type | Description |
|---|---|
| `start` | This workflow step is the start of a playbook. See section 4.3. |
| `end` | This workflow step is the end of a playbook or branch of workflow steps. See section 4.4. |
| `single` | This workflow step contains the actual commands to be executed. See section 4.5. |
| `playbook` | This workflow step executes a named playbook from within the current playbook. See section 4.6. |
| `parallel` | This workflow step contains a list of one or more steps that execute in parallel. See section 4.7. |
| `if-condition` | This workflow step contains an if-then-else statement. See section 4.8. |
| `while-condition` | This workflow step contains a while loop. See section 4.9. |
| `switch-condition` | This workflow step contains a switch statement. See section 4.10. |

## 4.3 Start Step

This workflow step is the starting point of a playbook or branch of steps. While this type inherits all of the common properties of a workflow step it does not define any additional properties.

| Property Name | Rq | Data Type | Details |
|---|---|---|---|
| **type** | Y | `string` | The value of this property **MUST** be `start`. |

**Example**

```
"step--a76dbc32-b739-427b-ae13-4ec703d5797e": {
  "type": "start",
  "name": "Start Playbook Example 1",
  "on_completion": "<some step id>"
}
```

## 4.4 End Step

This workflow step is the ending point of a playbook or branch of steps. While this type inherits all of the common properties of a workflow step it does not define any additional properties. When a playbook or branch of a playbook terminates it **MUST** call an End Step.

| Property Name | Rq | Data Type | Details |
|---|---|---|---|
| **type** | Y | `string` | The value of this property **MUST** be `end`. |

**Example**

```
"step--227b649f-cc38-4b75-b926-de631b4c42b1": {
  "type": "end",
  "name": "End Playbook Example 1",
}
```

## 4.5 Single Action Step

This workflow step contains the actual commands to be executed on one or more targets. These commands are intended to be processed sequentially one at a time. In addition to the inherited properties, this section defines five more specific properties that are valid for this type.

| Property Name | Rq | Data Type | Details |
|---|---|---|---|
| **type** | Y | `string` | The value of this property **MUST** be `single`. |
| **commands** | Y | `list` of type `command-data` | A list of commands that are to be executed as part of this step. If more than one command is listed, the commands **MUST** be processed in the order in which they are listed. |
| **target** | | `target` | A target that **SHOULD** execute the commands defined in this step.<br><br>The value of this property **MUST** contain a CACAO target object (see section 6). If this property is defined the **target_ids** property **MUST NOT** be defined. |

| | | | |
|---|---|---|---|
| **target_ids** | | list of type identifier | A list of target ID references that **SHOULD** execute the commands defined in this step.<br><br>Each ID **MUST** reference a CACAO target object. If this property is defined the **target** property **MUST NOT** be defined. |
| **in_args** | | list of type string | The optional list of arguments passed to the target(s) as input to the step |
| **out_args** | | list of type string | The optional list of arguments that are returned from this step after execution of the commands by the targets |

**Example**

```
"step--ba23c1b3-fdd2-4264-bc5b-c056c6862ba2": {
  "type": "single",
  "delay": 5000,
  "timeout": 60000,
  "on_success": "step--uuid2",
  "on_failure": "step--uuid99"
}
```

## 4.6 Playbook Step

This workflow step executes a referenced playbook on one or more targets. In addition to the inherited properties, this section defines five more specific properties that are valid for this type.

| Property Name | Rq | Data Type | Details |
|---|---|---|---|
| **type** | Y | string | The value of this property **MUST** be playbook. |
| **playbook_id** | Y | identifier | The referenced playbook to execute at the target or targets.<br><br>The playbook ID **SHOULD** be defined such that it is locally relevant to each target that will execute the playbook. |
| **target** | | target | A target that **SHOULD** execute the referenced playbook.<br><br>The value of this property **MUST** contain a CACAO target object (see section 6). If this property is defined the **target_ids** property **MUST NOT** be defined. |
| **target_ids** | | list of type identifier | A list of target ID references that **SHOULD** execute the named playbook<br><br>Each ID **MUST** reference a CACAO target object. If this property is defined the **target** property **MUST NOT** be defined. |

| | | | |
|---|---|---|---|
| **in_args** | | `list` of type `string` | The optional list of arguments passed to the target(s) as input to the referenced playbook. |
| **out_args** | | `list` of type `string` | The optional list of arguments that are returned from this playbook after execution of the commands by the targets. |

**Example**

```
"step--ba23c1b3-fdd2-4264-bc5b-c056c6862ba2": {
  "type": "playbook",
  "playbook_id": "playbook-uuid1",
  "delay": 5000,
  "timeout": 60000,
  "on_completion": "step--uuid2",
  "target_ids": ["$$LOCAL_TARGET$$"],
  "in_args": [ $$vuln_sys_id_1$$, $$vuln_sys_id_2$$ ],
  "out_args": [ $$result_1$$, $$result_2$$ ]
}
```

## 4.7 Parallel Step

This section defines how to create steps that can be processed in parallel. In addition to the inherited properties, this section defines one additional specific property that is valid for this type. A parallel step **MUST** execute all workflow steps that are part of the **`next_steps`** property before this step can be considered complete and the workflow logic moves on.

The Parallel Step is a playbook step that allows playbook authors to define two or more steps that can be executed at the same time. For example, a playbook that responds to an incident may require both the network team and the desktop team to investigate and respond to a threat at the same time. Another example is in response to a cyber attack on an operational technology (OT) environment that would require releasing air / steam / water pressure simultaneously.

The steps referenced from this object are intended to be processed in parallel, however, if an implementation can not support executing steps in parallel, then the steps **MAY** be executed in sequential order if the desired outcome stays the same.

| Property Name | Rq | Data Type | Details |
|---|---|---|---|
| **type** | Y | `string` | The value of this property **MUST** be `parallel`. |
| **next_steps** | Y | `list` of type `identifier` | A list of one or more workflow steps to be processed in parallel. The **`next_steps`** **MUST** contain at least one value.<br><br>The definition of *parallel execution* and how many parallel steps that are possible to execute is implementation dependent and is not part of this specification. |

| | | | |
|---|---|---|---|
| | | | If any of the steps referenced in **`next_steps`** generate an error of any kind (exception or timeout) then implementers **SHOULD** consider defining rollback error handling for the playbook and include those steps in the playbook itself.<br><br>The ID **MUST** represent either a CACAO workflow step object or a CACAO playbook object. |

**Example**

```
"step--46c1d6e1-874e-4588-b2a4-16d31634372c": {
  "type": "parallel",
  "next_steps": [
    "step--9afbcb12-8f82-4d35-ba70-f755b83725e1",
    "step--b4161d26-1c8d-4f19-b82f-aad144de4828"
  ],
  "on_completion": "step--44924d92-58c9-4fcc-9435-6fb651dbbddd"
}
```

## 4.8 If Condition Step

This section defines the 'if-then-else' conditional logic that can be used within the workflow of the playbook. In addition to the inherited properties, this section defines three additional specific properties that are valid for this type.

| Property Name | Rq | Data Type | Details |
|---|---|---|---|
| **`type`** | Y | `string` | The value of this property **MUST** be `if-condition`. |
| **`condition`** | Y | `string` | A boolean expression as defined in the STIX Patterning Grammar that when it evaluates as true executes the workflow step identified by the **`on_true`** property, otherwise it executes the **`on_false`** workflow step |
| **`on_true`** | Y | `list` of type `identifier` | The sequential list of step IDs to be processed if the condition evaluates as true.<br><br>Each ID **MUST** represent either a CACAO workflow step object or a CACAO playbook object. |
| **`on_false`** | Y | `list` of type `identifier` | The sequential list of step IDs to be processed if the condition evaluates as false.<br><br>Each ID **MUST** represent either a CACAO workflow Step object or a CACAO playbook object. |

**Example**

```
"step--uuid1": {
```

```
  "type": "if-condition",
  "delay": "5000",
  "timeout": "60000",
  "condition": "$$variable$$ == '10.0.0.0/8'",
  "on_true": [ "step--uuid2" ],
  "on_false": [ "step--uuid99" ]
}
```

## 4.9 While Condition Step

This section defines the 'while' conditional logic that can be used within the workflow of the playbook. In addition to the inherited properties, this section defines three additional specific properties that are valid for this type.

| Property Name | Rq | Data Type | Details |
|---|---|---|---|
| **type** | Y | `string` | The value of this property **MUST** be `while-condition`. |
| **condition** | Y | `string` | A boolean expression as defined in the STIX Patterning Grammar that ***while it is true*** executes the workflow step identified by on_do otherwise it exits the while conditional workflow step and executes the on_end workflow step |
| **on_true** | Y | `list` of type `identifier` | The list of sequential step IDs to be processed every time the loop condition evaluates as true.<br><br>Each ID **MUST** represent either a CACAO workflow step object or a CACAO playbook object. |
| **on_false** | Y | `identifier` | The ID of the next step to be processed every time the loop condition evaluates as false.<br><br>The ID **MUST** represent either a CACAO workflow step object or a CACAO playbook object. |

**Example**

```
"step--uuid1": {
  "type": "while-condition",
  "delay": "5000",
  "timeout": "60000",
  "condition": "$$variable$$ == '10.0.0.0/8'",
  "on_true": [ "step--uuid2" ],
  "on_false": "step--uuid99"
}
```

## 4.10 Switch Condition Step

This section defines the 'switch' condition logic that can be used within the workflow of the playbook. In addition to the inherited properties, this section defines two additional specific properties that are valid for this type.

| Property Name | Rq | Data Type | Details |
|---|---|---|---|
| **type** | Y | `string` | The value of this property **MUST** be `switch-condition`. |
| **switch** | Y | `string` | A variable that is evaluated to determine which key in the match_props dictionary is matched against to execute the associated step. |
| **cases** | Y | `dictionary` | This property is a dictionary that defines one or more case values (as dictionary keys) and a list of sequential step IDs (as key values) to be processed when the case value is matched against the switch value.<br><br>The value for each entry in the dictionary **MUST** be a `list` of type `identifier` that uniquely identifies a set of sequential steps to be processed when that key/value is chosen. Each id in the list **MUST** use the object type of "step" (see section 9.8 for more information on identifiers).<br><br>This dictionary **MAY** have a "default" case value. |

**Example**

```
"step--uuid1": {
  "type": "switch-condition",
  "delay": "5000",
  "timeout": "60000",
  "switch": "$$variable$$",
  "cases": {
    "1": [ "step--uuid2" ],
    "2": [ "step--uuid3" ],
    "default": [ "step--uuid4" ]
  }
}
```

# 5 Commands

The CACAO command object (`command-data`) contains detailed information about the commands that are to be executed or processed automatically or manually as part of a workflow step (see section 4). Each command listed in a step may be of a different command type, however, all commands listed in a single step **MUST** be processed or executed by all of the targets defined in that step.

Commands can use and refer to variables just like other parts of the playbook. For each command either the **`command`** property or the **`command_b64`** property **MUST** be present.

The individual commands **MAY** be defined in other specifications, and when possible will be mapped to the JSON structure of this specification. When that is not possible, they will be base64 encoded.

## 5.1 Command Data Type

| Property Name | Rq | Data Type | Details |
|---|---|---|---|
| **`type`** | Y | `string` | The type of command being used. The value of this property **MUST** come from the `command-type-ov` vocabulary. |
| **`command`** | | `string` | A string based command as defined by the type. Commands can be simple strings or stringified JSON based on the defined type.<br><br>The command **MUST** be valid for the defined type and version. |
| **`command_b64`** | | `string` | A base64 encoded command as defined by the type. This property is used for structured commands that are not simple strings or native JSON.<br><br>The command **MUST** be valid for the defined type and version. |
| **`version`** | | `string` | An optional version of the command language being used. If no version is specified then the most current version of the command language **SHOULD** be used. |

**Examples**

```
{
  "type": "http-api",
  "command": "hxxps://www[.]example[.]com/v1/getData?id=1234",
}
```

```
{
  "type": "manual",
  "command": "Disconnect the machine from the network and call the SOC on-call person",
}
```

```
{
  "type": "ssh",
  "command": "last; netstat -n; ls -l -a /root",
}
```

## 5.2 Command Type Vocabulary

**Open Vocabulary Name:** `command-type-ov`

This section defines the following types of commands that can be used within a CACAO workflow step.

| Command Type | Description |
|---|---|
| `manual` | This type represents a command that is intended to be processed by a human or a system that acts on behalf of a human. |
| `http-api` | An HTTP API command. |
| `ssh` | An SSH command. |
| `bash` | A Bash command. |
| `openc2-json` | A command expressed in OpenC2 JSON. |
| `attack-cmd` | A command that is used by an attack orchestration system to attack or simulate an attack against a target. These can include attacks from vulnerability assessment or penetration systems.<br><br>An example would be a Caldera ability included in the command when the target specifies a Caldera agent or group. (See [CalderaAbility]).<br><br>If the **`command`** property is used, then that property **SHOULD** only contain the id of the Caldera ability that is to be executed. However, if the entire Caldera ability is to be shared, implementers **SHOULD** use the **`command_b64`** property to contain the base64 encoded version of the ability itself. |

**Caldera Ability Command Example**
Note: the content of the command is shown as text for illustration purposes only. The content of the **`command_b64`** would be a base64 encoded version of the ability text shown below.

```
{
  "type": "attack-cmd",
  "command_b64": "id: 9a30740d-3aa8-4c23-8efa-d51215e8a5b9
    name: Scan WIFI networks
    description: View all potential WIFI networks on host
    tactic: discovery
    technique:
      attack_id: T1016
      name: System Network Configuration Discovery
    platforms:
      darwin:
        sh:
          command: |
```

```
            ./wifi.sh scan
            payload: wifi.sh
      linux:
        sh:
          command: |
            ./wifi.sh scan
            payload: wifi.sh
      windows:
        psh,pwsh:
          command: |
            .\wifi.ps1 -Scan
            payload: wifi.ps1"
}
```

An Attack Command Example

This is used when combined with a caldera target. Highlights that the playbook only needs to pass the ID of the ability to execute in the caldera environment and does not require to send the entire ability.

```
{
  "type": "attack-cmd",
  "command": "id: 9a30740d-3aa8-4c23-8efa-d51215e8a5b9"
}
```

# 6 Targets

The CACAO target object contains detailed information about the entities or devices that accept, receive, process, or execute one or more commands as defined in a workflow step. Targets contain the information needed to send commands as defined in steps to devices or humans.

In a CACAO playbook, targets can be stored in a dictionary where the ID is the key and the target object is the value. Workflow steps can either embed the target or reference it by its ID.

Targets can use and refer to variables just like other parts of the playbook. While the target's name and description are optional, they are encouraged and producers **SHOULD** populate them.

Targets are classified in one of two categories, manual and automated. Targets can include, but are not limited to the following:

- Manual Processing
  - Individual/person
  - Group/team
  - Organization
  - Physical and Logical Locations
  - Sector/industry
- Automated Processing
  - Technology Categories such as firewalls, IPS, Switch, Router, Threat Intelligence Platform, etc.
  - Specific technology and associated version(s) (e.g., Windows 10, Cisco ASA firewall version 13.4)
  - Specific network addressable security functions (Windows 10 at IPv4/IPv6/MAC address, Function Call at specific URL, WebHook, API, Shell Script, SSH, etc.)

** GENERAL NOTE: For any target property values, the producer may define a variable substitution such that the actual property value is determined at runtime based on the variable assigned to the target.

Example: A target is referenced within a workflow step, but the target's actual values are based on variables (e.g., name, email, phone, location) instead of being hard-coded by the target itself.

```
{
  "type": "individual",
  "name": "$$INDIVIDUALS_NAME$$",
  "email": "$$INDIVIDUALS_EMAIL$$",
  "phone": "$$INDIVIDUALS_PHONE$$",
  "location": "$$INDIVIDUALS_LOCATION$$"
}
```

## 6.1 Common Target Properties

Each target contains some base properties that are common across all targets. These properties are defined in the following table. The ID for each target is stored as the key in the **`targets`** dictionary.

| Property Name | Rq | Data Type | Details |
|---|---|---|---|

| | | | |
|---|---|---|---|
| **type** | Y | `string` | The type of target object being used. The value of this property **MUST** come from the `target-type-ov`. |
| **name** | Y | `string` | The name that represents this target that is meant to be displayed in a user interface or captured in a log message. |
| **description** | | `string` | More details, context, and possibly an explanation about this target. |
| **target_extensions** | | `dictionary` | This property defines the extensions that are in use on this target.<br><br>The key for each entry in the dictionary **MUST** be an `identifier` that uniquely identifies the extension. The id **MUST** use the object type of "extension" (see section 9.7 for more information on identifiers). The value for each key is a JSON object that can contain the structure as defined in the extension's schema location. |

## 6.2 Target Type Vocabulary

**Open Vocabulary Name:** `target-type-ov`

This section defines the following types of targets.

| Target Type | Description |
|---|---|
| `individual` | The target is a human-being. |
| `group` | The target is a group typically associated with a team, or organizational group. |
| `organization` | The target is a named organization or business entity. |
| `location` | The target is an identified location (either physical or logical). |
| `sector` | The target is a business or government sector. Includes industrial categories. |
| `http-api` | The target is an HTTP API interface. |
| `ssh` | The target is a device running the SSH service. |
| `security-infrastructure-category` | The target is a named security infrastructure category such as Firewall, IPS, TIP, etc. |
| `net-address` | The target is an identified network addressable entity that supports execution of a workflow step or playbook. |

| | |
|---|---|
| `kali` | The target is a kali linux orchestration system, where the executor accepts SSH commands. |
| `attacker` | The target is an attacker typically running on an attacker orchestration system that will execute the commands instead of the host of the orchestration system. |
| `attack-agent` | The target represents the system that will be targeted by an attack orchestration system. |
| `attack-group` | A target represents a group of systems that will be targeted by an orchestration system at the same time. |

## 6.3 Individual Target

This target type is used for commands that need to be processed or executed by an individual. This object inherits the common target properties. In addition to the inherited properties, this section defines two additional specific properties that are valid for this type.

| Property Name | Rq | Data Type | Details |
|---|---|---|---|
| **`type`** | Y | `string` | The value of this property **MUST** be `individual`. |
| **`contact`** | | `contact` | Contact information for this target. |
| **`location`** | | `civic-location` | Physical address information for this target. |

## 6.4 Group Target

This target type is used for commands that need to be processed or executed by a group. This object inherits the common target properties. In addition to the inherited properties, this section defines two additional specific properties that are valid for this type.

| Property Name | Rq | Data Type | Details |
|---|---|---|---|
| **`type`** | Y | `string` | The value of this property **MUST** be `group`. |
| **`contact`** | | `contact` | Contact information for this target. |
| **`location`** | | `civic-location` | Physical address information for this target. |

## 6.5 Organization Target

This target type is used for commands that need to be processed or executed by an organization. This object inherits the common target properties. In addition to the inherited properties, this section defines two additional specific properties that are valid for this type.

| Property Name | Rq | Data Type | Details |
|---|---|---|---|
| **type** | Y | `string` | The value of this property **MUST** be `organization`. |
| **contact** | | `contact` | Contact information for this target. |
| **location** | | `civic-location` | Physical address information for this target. |

## 6.6 Location Target

This target type is used for commands that need to be processed or executed by a location. This object inherits the common target properties. In addition to the inherited properties, this section defines three additional specific properties that are valid for this type.

| Property Name | Rq | Data Type | Details |
|---|---|---|---|
| **type** | Y | `string` | The value of this property **MUST** be `location`. |
| **location** | | `civic-location` | Physical address information for this target. |
| **gps** | | `gps-location` | GPS information for this target. |
| **logical** | | `list` of type `string` | An optional list of logical location names as defined by the playbook creator. |

## 6.7 Sector Target

This target type is used for commands that need to be processed or executed by a sector. This object inherits the common target properties. In addition to the inherited properties, this section defines one additional specific property that is valid for this type. The values for the inherited **name** property **SHOULD** come from the `industry-sector-ov` vocabulary, see section 6.7.1.

| Property Name | Rq | Data Type | Details |
|---|---|---|---|
| **type** | Y | `string` | The value of this property **MUST** be `sector`. |
| **location** | | `list` of type `civic-location` | An optional list of physical address information for this target. |

### 6.7.1 Industry Sector Vocabulary

**Vocabulary Name:** `industry-sector-ov`

Industry sector is an open vocabulary that describes industrial and commercial sectors.

| Sector Type | Description |
|---|---|
| aerospace | The aerospace sector/industry comprises entities that research, design, manufacture, operate, and maintain aircraft and spacecraft technology. Aerospace activity is very diverse, with a plethora of commercial, industrial, and military applications.<br><br>Subclasses (sector/industry): aviation |
| aviation | The aviation sector/industry is similar to the aerospace sector/industry, but its applications are focused within the earth's atmosphere. |
| agriculture | The agriculture sector/industry comprises entities primarily engaged in farming animals (animal husbandry) and plants (agronomy, horticulture, and forestry in part). |
| automotive | The automotive sector/industry comprises entities involved in the manufacture of motor vehicles, including most components, such as engines and bodies, but excluding tires, batteries, and fuel [VocabAuto]. |
| biotechnology | The biotechnology sector/industry comprises entities involved in utilizing biotechnology to develop products. Biotechnology is known to highly overlap with the pharmaceutical sector/industry, but generally, it has a plethora of use cases like in agriculture, food, and chemical sectors/industries. |
| chemical | The chemical sector/industry comprises entities producing petrochemicals, polymers, basic inorganics, specialties, and consumer chemicals [VocabChem]. The products (chemicals) produced by the chemical sector/industry have a broad range of uses, such as in the food industry, pharmaceutical, agriculture, manufacturing, and industries involved in consumer goods. |
| commercial | The commercial sector/industry comprises entities involved in wholesale and retail trade, generally without making any changes to the goods. The commercial sector, in this case, does not include professional services. |
| consulting | The consulting sector/industry comprises entities that provide expert advice and possibly implementation services in exchange for a fee. |
| construction | The construction sector/industry comprises entities involved in building construction (residential and non-residential) , |

| | |
|---|---|
| | infrastructure construction (e.g., large public works, dams, bridges, roads, airports, railways, and tramlines), and industrial construction (e.g., energy installations, manufacturing plants). |
| `cosmetics` | The cosmetics sector/industry comprises entities that manufacture and distribute cosmetic products (e.g., hygiene products such as soap, shampoo, deodorant, and toothpaste to luxury beauty items including perfumes and makeup). |
| `critical-infrastructure` | Critical infrastructure comprises sectors with assets or systems that are essential for maintaining vital societal functions. It is the case that different nations may define more or fewer assets as critical infrastructures. For example, the U.S.A defines sixteen sectors as critical infrastructures, whereas Norway defines six, namely, communication networks, energy, water and wastewater, transportation, oil and gas, and satellite communications. |
| `dams` | The dams sector/industry comprises entities involved in operating and maintaining dams. Based on its purpose, a dam can be considered critical infrastructure for a nation. Dams provide a wide range of economic, environmental, and social benefits, including hydroelectric power, river navigation, water supply, wildlife habitat, waste management, flood control, and recreation [VocabDams]. |
| `defense` | The defense sector comprises government and commercial entities involved in research and development, design, production, delivery, and maintenance of weapons, weapon systems, subsystems, and components or parts, to meet military requirements. The defense sector covers everything from land, sea, and air defense capabilities and cybersecurity. |
| `education` | The education sector/industry comprises entities that facilitate learning, such as schools, colleges, and universities. |
| `emergency-services` | The emergency services sector/industry provides a wide range of prevention, preparedness, response, and recovery services during both day-to-day operations and incident response. The emergency services sector includes geographically distributed facilities and equipment in both paid and volunteer capacities organized primarily at the federal, state, local, tribal, and territorial levels of government, such as city police departments and fire stations, county sheriff's offices, Department of Defense police and fire departments, and town public works departments. The emergency services sector also includes private sector |

| | |
|---|---|
| | resources, such as industrial fire departments, private security organizations, and private emergency medical services providers [VocabEmSrv]. |
| `energy` | The energy sector/industry comprises entities involved in electricity generation and distribution, such as energy generation infrastructure (e.g., power plants), operators (e.g., grid operators), transmission and distribution lines, and utility providers. Electricity can be generated from renewable resources (e.g., hydro, wind, solar, biomass, geothermal), fossil fuels (such as coal and gas), or nuclear power. The energy sector/industry overlaps with the utilities sector/industry.<br><br>Subclasses (sector/industry): `non-renewable-energy`, `renewable-energy`. |
| `non-renewable-energy` | The non-renewable energy sector/industry comprises entities involved in electricity generation and distribution from non-renewable resources such as oil and petroleum products, gasoline, natural gas, diesel fuel, and nuclear. |
| `renewable-energy` | The renewable energy sector/industry comprises entities involved in electricity generation and distribution from renewable resources such as hydro, wind, solar, biomass, and geothermal. |
| `media` | The media sector/industry consists of film, print, radio, and television, also when provided in electronic form via the Internet. |
| `financial` | The financial sector/industry is a broad term used to describe a range of activities that manage money, encompassing everything from insurance companies and stock brokerages to investment funds and banking. |
| `food` | The food sector/industry comprises entities involved in the processing, preparation, preservation, and packaging of food and beverages. The raw materials used are generally of vegetable or animal origin and produced by agriculture like farming, breeding, and fishing. Foodservice is also included and comprises entities that serve food to people, such as restaurants. |
| `gambling` | The gambling or betting sector/industry includes online or offline gambling entities like casinos and other online betting games. |

| | |
|---|---|
| government | The government sector includes a wide variety of entities involved in governmental nature activities. It includes facilities owned or leased by federal, state, local, and tribal governments. Many government facilities are open to the public for business activities, commercial transactions, or recreational activities. In contrast, others that are not open to the public contain highly sensitive information, materials, processes, and equipment. These facilities include general-use office buildings and special-use military installations, embassies, courthouses, national laboratories, and structures that may house critical equipment, systems, networks, and functions. In addition to physical structures, the sector includes cyber elements that contribute to the protection of sector assets (e.g., access control systems and closed-circuit television systems) as well as individuals who perform essential functions or possess tactical, operational, or strategic knowledge [VocabGov]. The government sector overlaps with multiple other sectors/industries like the defense sector/industry that comprises government entities that support a nation's national security capability, and the emergency services sector that provides a wide range of prevention, preparedness, response, and recovery services during both day-to-day operations and incident response.<br><br>Subclasses (sector/industry): local-government, national-government, regional-government, public-services. |
| local-government | A city or municipality level of government. See government above for more details. |
| national-government | A national level of government. See government above for more details. |
| regional-government | A regional, state, or area level of government. See government above for more details. |
| public-services | The public services sector/industry includes services provided by a government to people living within its jurisdiction, either directly through public sector agencies or by financing provision of services by private businesses or voluntary organizations [VocabPServ]. The public services sector may overlap with other sectors like healthcare, education, transportation, and utilities. See government above for more details. |
| healthcare | The healthcare sector/industry comprises entities that provide healthcare, meaning services to assess, maintain or restore a |

| | |
|---|---|
| | patient's state of health, including the prescription, dispensation, and provision of medicinal products and medical devices [VocabHealth]. |
| `information-communications-technology` | The information and communications technology (ICT) sector/industry, also known as the information technology sector or just technology sector, comprises entities that produce and provide information technology services and products such as software, hardware, electronics, and telecommunications.<br><br>Subclasses (sector/industry): `electronics-hardware`, `software`, `telecommunications` |
| `electronics-hardware` | The electronics and hardware sector/industry comprises entities that produce electronic equipment and components and computer hardware. |
| `software` | The software sector/industry comprises entities dedicated to producing software. |
| `telecommunications` | The telecommunications industry within the ICT sector comprises entities that produce and provide telecommunications equipment and services. Examples are Internet Service Providers (ISPs), wired and mobile telephony providers, and satellite communications operators. A satellite research and production facility would fall optimally under the aerospace sector/industry. |
| `legal-services` | The legal services sector/industry, otherwise known as the legal industry, comprises entities that provide services of lawyers and other legal practitioners to individuals, businesses, government agencies, and nonprofits. |
| `lodging` | The lodging sector/industry is a segment of the hospitality sector/industry specializing in providing customers with accommodation services (e.g., hotels, motels, resorts, and bed and breakfasts). |
| `manufacturing` | The manufacturing sector/industry comprises entities involved in the creation of products from raw materials and commodities. It includes all foods, chemicals, textiles, machines, and equipment, it includes all refined metals and minerals derived from extracted ores, and it includes all lumber, wood, and pulp products. As a best practice, the manufacturing sector/industry element **SHOULD** be used when none of the taxonomy elements is adequate for tagging or classifying a case. In addition, like the rest of the elements, manufacturing can be used in combination with another |

| | |
|---|---|
| | element to provide extra precision in classification/tagging. For example, an incident that affects a company in the automotive sector/industry can be tagged as both automotive and manufacturing to indicate that the manufacturing process or a manufacturing plant was targeted. |
| `maritime` | The maritime sector/industry involves a plethora of organizations and activities related to the ocean and ships and other floating entities. Examples are maritime transportation, shipyards, maritime equipment manufacturers, and commercial fishing. The maritime sector/industry highly overlaps with the transportation sector/industry. |
| `metals` | The metals sector/industry comprises entities involved in the processing of non-ferrous metals such as aluminum, copper, zinc, and ferrous materials such as steel [VocabMetals]. |
| `mining` | The mining sector/industry comprises entities dedicated to locating and extracting metal and mineral reserves. Oil and natural gas extraction are not included in this industry, but they can be referenced using the Petroleum sector/industry. |
| `non-profit` | The nonprofit sector/industry comprises entities organized and operated for a collective, public, or social benefit compared to for-profit organizations that aim to generate a profit. The nonprofit sector/industry may overlap with other sectors/industries that also accommodate nonprofit entities, like public universities that are part of the education sector/industry, but they have a nonprofit cause. Further, non-governmental organizations (NGOs) are not distinguished and thus are included in the nonprofit sector/industry.<br><br>Subclasses (sector/industry): `humanitarian-aid`, `human-rights`. |
| `humanitarian-aid` | The humanitarian aid industry or humanitarian aid organizations provide material and logistic assistance to people who need help (e.g., homeless, refugees, and victims of natural disasters, wars, and famines). |
| `human-rights` | The human rights industry comprises establishments primarily engaged in promoting causes associated with human rights either for a broad or specific constituency. Establishments in this industry address issues, such as protecting and promoting the broad constitutional rights and civil liberties of individuals and those suffering from neglect, abuse, or exploitation; promoting the interests of specific groups, such as children, women, senior citizens, or persons with disabilities; improving relations between racial, ethnic, |

| | |
|---|---|
| | and cultural groups; and promoting voter education and registration [VocabHumanRights]. |
| `nuclear` | The nuclear sector/industry includes nuclear infrastructure, and in this taxonomy, it is unrelated to how the nuclear power is used, such as for energy generation, radioactive materials for healthcare, or for developing nuclear weapons. |
| `petroleum` | The petroleum sector/industry, also known as the oil sector/industry or the oil and gas sector/industry, comprises entities involved in the global processes of exploration, extraction, refining, and transporting (often by oil tankers and pipelines) petroleum. The petroleum industry overlaps with the chemical sector/industry in the sense that many finished products derived from petrochemical processing. |
| `pharmaceuticals` | The pharmaceutical sector/industry comprises entities that research, develop, produce, and distribute pharmaceutical products like medications. |
| `research` | The research industry comprises entities that solely specialize in research like research institutions, think tanks, and research groups/divisions within organizations. |
| `transportation` | The transportation sector/industry comprises entities that provide services to move people or goods, including transportation infrastructure. The transportation sector consists of several industries that focus on transportation, including air freight and logistics, airlines, marine, road and rail, and transportation infrastructures like railroads and marine ports.<br><br>Subclasses (sector/industry): `logistics-shipping` |
| `logistics-shipping` | The logistics and shipping sector/industry comprise entities responsible for planning, implementing, and controlling procedures for the efficient and effective transportation and storage of goods. |
| `utilities` | The utilities sector comprises entities that provide basic amenities, such as water, sewage services, electricity, dams, and natural gas [VocabUtils]. |
| `video-game` | The video game industry comprises entities involved in the development, marketing, and monetization of video games. The video game industry overlaps with other sectors/industries like the ICT and, in particular, the software sector/industry. |

| | |
|---|---|
| `water` | The water sector/industry comprises entities that provide water and wastewater services, including sewage treatment. The water sector/industry does not include manufacturers and suppliers of bottled water, which is part of the beverage production and belongs to the food sector/industry [VocabWater]. |

**Example**

```
"targets": {
  "target--5aa10ecd-c367-4157-82b1-2b4891d4ae3e": {
    "type": "sector",
    "name": "healthcare"
  }
}
```

## 6.8 HTTP API Target

This target type contains an HTTP API target. In addition to the inherited properties, this section defines six additional specific properties that are valid for this type. In addition to the inherited properties, this section defines two additional specific properties that are valid for this type.

| Property Name | Rq | Data Type | Details |
|---|---|---|---|
| **type** | Y | `string` | The value of this property **MUST** be `http-api`. |
| **http_url** | Y | `string` | A full URL of the HTTP API service that should be called. |
| **http_auth_type** | | `string` | The authentication type required to access this HTTP target (e.g., "basic", "oauth2", etc.) |
| **user_id** | | `string` | The user-id property used in HTTP Basic authentication as defined by [RFC7617]. |
| **password** | | `string` | The password property used in HTTP Basic authentication as defined by [RFC7617]. |
| **token** | | `string` | The bearer token used in HTTP Bearer Token authentication as defined by [RFC6750]. |
| **oauth_header** | | `string` | The OAuth header used in OAuth authentication as defined in section 3.5.1 of [RFC5849]. |

## 6.9 SSH CLI Target

This target type contains an SSH CLI target. In addition to the inherited properties, this section defines five additional specific properties that are valid for this type.

| Property Name | Rq | Data Type | Details |
|---|---|---|---|
| **type** | Y | `string` | The value of this property **MUST** be `ssh`. |
| **address** | Y | `string` | The IP address or domain name of the host that should be contacted. |
| **port** | | `string` | The TCP port number for the SSH service. The default value is 22 based on standard port number services [PortNumbers]. |
| **username** | | `string` | The username to access this target. |
| **password** | | `string` | The password associated with the username to access this target. This value will most often be passed in via a variable. |
| **private_key** | | `string` | The private key associated with the username to access this target. This value will most often be passed in via a variable. |

## 6.10 Security Infrastructure Category Target

This target type contains a Security Infrastructure Category Target. In addition to the inherited properties, this section defines one additional specific property that is valid for this type.

| Property Name | Rq | Data Type | Details |
|---|---|---|---|
| **type** | Y | `string` | The value of this property **MUST** be `security-infrastructure-category`. |
| **category** | Y | `list` of type `string` | One or more identified categories of security infrastructure types that this target represents. A product instantiation may include one or more security infrastructure types as hints to assist in describing the target features most likely required by a playbook step or playbook.<br><br>The values for this property **MUST** come from the `security-infrastructure-type-ov` vocabulary. |

### 6.10.1 Security Infrastructure Type Vocabulary

**Open Vocabulary Name:** `security-infrastructure-type-ov`

This section defines the infrastructure types where a type captures the key characteristics a playbook or playbook step may relate to. It includes values from the very general to the more specific and is not intended to be exhaustive nor binary. This information is intended as a hint.

| Infrastructure Type | Description |
|---|---|
| `endpoint` | The infrastructure supports general computer device features with no specific constraints or requirements. |
| `handset` | The infrastructure supports handset device features. |
| `router` | The infrastructure supports routing at L2, L3, L4. |
| `firewall` | The infrastructure supports L3, L4 or above firewalling. |
| `ids` | The infrastructure supports intrusion detection. |
| `ips` | The infrastructure supports intrusion prevention. |
| `aaa` | The infrastructure supports authentication, authorization and accounting services. |
| `os-windows` | The infrastructure supports Windows operating system specific constraints. |
| `os-linux` | The infrastructure supports Linux operating system specific constraints. |
| `os-mac` | The infrastructure supports Mac-OS operating system specific constraints. |
| `switch` | The infrastructure supports L2, L3 or above switching constraints. |
| `wireless` | The infrastructure supports wireless communications typically associated with 802.11 radio communication. |
| `desktop` | The infrastructure is a desktop. |
| `server` | The infrastructure supports server functionality common in deployments such as the cloud or services supporting multiple client devices and applications. |
| `content-gateway` | The infrastructure supports content gateway inspection and mitigation. |
| `analytics` | The infrastructure supports some form of analytical processing such as flow processing, anomaly detection, machine-learning, behavioral detection, etc. |
| `siem` | The infrastructure supports SIEM functionality. |
| `tip` | The infrastructure supports threat intelligence platform features. |
| `ticketing` | The infrastructure supports trouble-ticketing, workload processing, etc. |

**Example**

```
"targets": {
  "target--f81aa730-2c59-4190-b8d5-3f2b4beecd95": {
    "type": "security-infrastructure-category",
```

```
    "category": ["firewall"]
  }
}
```

## 6.11 General Network Address Target

This target type contains a Network Address Target. In addition to the inherited properties, this section defines six additional specific properties that are valid for this type.

| Property Name | Rq | Data Type | Details |
|---|---|---|---|
| **type** | Y | `string` | The value of this property **MUST** be `net-address`. |
| **address** | Y | `dictionary` | The key for each entry in the dictionary **MUST** be a `string` that uniquely identifies the address type. The value for each key **MUST** be a `string` and **MUST** have a value of ipv4, ipv6, l2mac, vlan, or url. |
| **username** | | `string` | The username to access this target. |
| **password** | | `string` | The password associated with the username to access this target. This value will most often be passed in via a variable. |
| **private_key** | | `string` | The private key associated with the username to access this target. This value will most often be passed in via a variable. |
| **category** | | `string` | The optional categories of security infrastructure this network addressable entity represents. See section 6.10.1.<br><br>The values for this property, if defined, **MUST** come from the `security-infrastructure-type-ov` vocabulary. |
| **location** | | `civic-location` | Physical address information for this target. |

**Example**

```
"targets": {
  "target--6f6f9814-5982-4322-9a9c-0ef25d33ef2a": {
    "type": "net-address",
    "address": {
      "url": "hxxps://someorg[.]com/tellmetoorchestratewhat/amethod"
    },
    "username": "someusername",
    "password": "apassword",
    "category": "firewall",
    "location": {
```

```
    }
  }
}
```

## 6.12 Attacker Target

This target type is used for commands that need to be executed on an attack orchestration system. This object inherits the common target properties. In addition to the inherited properties, this section defines three additional specific properties that are valid for this type.

| Property Name | Rq | Data Type | Details |
|---|---|---|---|
| **type** | Y | `string` | The value of this property **MUST** be `attacker`. |
| **executor** | | `target` | The executor of the attack |
| **executor-type** | | `string` | The type of orchestration system that runs the attacks.<br><br>The values for this property **MUST** come from the `orchestration-system-type` vocabulary. |
| **subject** | | `target` | The subject of the attack |

### 6.12.1 Orchestration System Type Vocabulary

**Vocabulary Name:** `orchestration-system-type`

| Vocabulary Value | Description |
|---|---|
| `kali` | Kali Linux is an open-source, Debian-based Linux distribution geared towards various information security tasks, such as Penetration Testing, Security Research, Computer Forensics and Reverse Engineering. |
| `caldera` | Caldera allows organizations to test endpoint security solutions and assess a network's security posture against the common post-compromise adversarial techniques contained in the MITRE ATT&CK model. |
| `redcanary-atomicred` | Atomic Red Team is a collection of small, highly portable detection tests mapped to MITRE ATT&CK®. This gives defenders a highly actionable way to immediately start testing their defenses against a broad spectrum of attacks. |

**Examples**

```
"target" : {
  "type": "attacker",
  "executor" : {
    "type": "kali",
    "address" : "12.1.1.1",
```

```
    "port": "22"
  },
  "executor-type": "kali",
  "subject": {
    "type": "net-address",
    "address" : "10.1.1.1"
  }
},

"target" : {
  "type": "attacker",
  "executor" : {
    "type": "net-address",
    "address" : "12.1.12.12",
    "port": "8888"
  },
  "executor-type": "caldera",
    "subject": {
    "type": "attack-agent",
    "address" : "10.1.1.1",
    "agent-type": "Sandcat"
  }
},
```

## 6.13 Attack Agent Target

This target type contains an Attack Agent target. In addition to the inherited properties, this section defines three additional specific properties that are valid for this type. For a Caldera agent example see [CalderaAgent] for more details.

| Property Name | Rq | Data Type | Details |
|---|---|---|---|
| **type** | Y | `string` | The value of this property **MUST** be `attack-agent`. |
| **address** | Y | `string` | The IP address or domain name of the Attack agent that should execute the attack command sent to it. |
| **agent-type** | | `string` | The type of agent being used.<br><br>The values for this property **MUST** come from the `attack-agent-type` vocabulary. |

### 6.13.1 Attack Agent Type Vocabulary

**Vocabulary Name:** `attack-agent-type`

| Vocabulary Value | Description |
|---|---|
| `sandcat` | The Sandcat plugin is the default agent used in a Caldera operation. |

| | |
|---|---|
| `manx` | The Manx plugin supplies shell access into Caldera, along with reverse-shell payloads for entering/exiting agents manually. |
| `ragdoll` | The Ragdoll plugin gets instructions by scraping the decoy web page, it then sends results through GET URL parameters (encoded). |

## 6.14 Attack Group Target

This target type contains an Attack Group target. In addition to the inherited properties, this section defines one additional specific property that is valid for this type. For a Caldera Group example, see [CalderaGroup] for more details.

| Property Name | Rq | Data Type | Details |
|---|---|---|---|
| **type** | Y | `string` | The value of this property **MUST** be `attack-group`. |
| **name** | Y | `string` | The group name of the Attack group that should execute the attack command sent to it and all agents that are part of the group. |

## 6.15 Kali Linux Target

This target type contains a Kali CLI target. In addition to the inherited properties, this section defines five additional specific properties that are valid for this type.

| Property Name | Rq | Data Type | Details |
|---|---|---|---|
| **type** | Y | `string` | The value of this property **MUST** be `kali`. |
| **address** | Y | `string` | The IP address or domain name of the host that should be contacted. |
| **port** | | `string` | The TCP port number for the Kali linux service. The default value is 22 based on standard port number services [PortNumbers]. |
| **username** | | `string` | The username to access this target. |
| **password** | | `string` | The password associated with the username to access this target. This value will most often be passed in via a variable. |
| **private_key** | | `string` | The private key associated with the username to access this target. This value will most often be passed in via a variable. |

# 7 Extension Definitions

The CACAO extension object allows a playbook producer to define detailed information about the extensions that are in use in a playbook that they created. In a playbook, extensions are stored in a dictionary where the ID is the key and the extension object is the value. Workflow steps, targets, data markings and playbooks themselves can use extensions by referencing their IDs.

Extensions can use and refer to all objects that may be used in other parts of a playbook including variables and constants just like other parts of the playbook. While the extension's name and description are optional, they are encouraged and producers **SHOULD** populate them.

## 7.1 Extension Properties

| Property Name | Rq | Data Type | Details |
|---|---|---|---|
| **type** | Y | `string` | The value of this property **MUST** be `extension-definition`. |
| **name** | Y | `string` | A name used to identify this extension for display purposes during execution, development or troubleshooting. |
| **description** | | `string` | More details, context, and possibly an explanation about what this extension does and accomplishes.<br><br>While the extension's description is optional, it is encouraged that producers **SHOULD** populate the field.<br><br>Note that the schema property is the normative definition of the extension, and this property, if present, is for documentation purposes only. |
| **created_by** | Y | `identifier` | An ID that represents the entity that created this extension. The ID **MUST** represent a STIX 2.1+ identity object. |
| **schema** | Y | `string` | The normative definition of the extension, either as a URL or as text explaining the definition.<br><br>A URL **SHOULD** point to a JSON schema or a location that contains information about the schema. |
| **version** | Y | `string` | The version of this extension. Producers of playbook extensions are encouraged to follow standard semantic versioning procedures where the version number follows the pattern, MAJOR.MINOR.PATCH **[SemVer]**. This will allow consumers to distinguish between the three different levels of compatibility typically identified by such versioning strings. |

### Step Extension Example 1

```
"extension-definition--uuid1": {
  "type": "extension-definition",
  "name": "Extension Foo",
  "description": "This schema adds foo to bar for steps",
  "created_by": "identity--uuid1",
  "schema": "hxxps://www[.]example[.]com/schema-foo/v1/",
  "version": "1.2.1"
}
```

### Step Extension Example 2

```
{
  "type": "playbook",
  ...
  "workflow": {
    "step--uuid1": {
      "type": "single",
      "delay": 5000,
      "timeout": 60000,
      "on_success": "step--uuid2",
      "on_failure": "step--uuid99",
      "step_extensions": {
        "extension-definition--45c72acc-d124-481e-8b12-57ab1fd4c136": {
          "dosome-custom-command": {
            "command_uuid" : "command-uuud1",
            "command_value" : "1.0.1.1"
          },
          "dosome-custom-command2": "command-uuid2"
        }
      }
    }
  },
  "extension-definitions": {
    "extension-definition--45c72acc-d124-481e-8b12-57ab1fd4c136": {
      "type": "extension-definition",
      "name": "Some cool schema",
      "description": "This schema adds foo to bar",
      "created_by": "identity--uuid1",
      "schema": "hxxps://www[.]example[.]com/schema-foo/v1/",
      "version": "1.2.1"
    }
  }
}
```

### Target Extension Example

```
{
  "type": "playbook",
  ...
  "target": {
    "type": "net-address",
    "address": {
      "url": "hxxps://someorg[.]com/tellmetoorchestratewhat/amethod",
      "vlan": "vlan1"
    },
    "username": "someusername",
    "password": "apassword",
    "target_extensions": {
      "extension-definition--45c72acc-d124-481e-8b12-57ab1fd4c144": {
        "l2_address": "010203040506"
      }
```

```
    }
  },
  "extension-definitions": {
    "extension-definition--45c72acc-d124-481e-8b12-57ab1fd4c144": {
      "type": "extension-definition",
      "name": "Network Target with Mac",
      "description": "This schema adds L2 mac address to network targets",
      "created_by": "identity--uuid1",
      "schema": "hxxps://www[.]example[.]com/schema-foo/v1/",
      "version": "1.2.1"
    }
  }
}
```

# 8 Data Marking Definitions

CACAO data marking definition objects contain detailed information about a specific data marking. Data markings typically represent handling or sharing requirements and are applied via the **markings** property in a playbook.

Data marking objects **MUST NOT** be versioned because it would allow for indirect changes to the markings on a playbook. For example, if a statement marking definition is changed from "Reuse Allowed" to "Reuse Prohibited", all playbooks marked with that statement marking definition would effectively have an updated marking without being updated themselves. Instead, in this example a new statement marking definition with the new text should be created and the marked objects updated to point to the new data marking object.

Playbooks may be marked with multiple marking statements. In other words, the same playbook can be marked with both a statement saying "Copyright 2020" and a statement saying, "Terms of use are ..." and both statements apply.

## 8.1 Data Marking Common Properties

Each data marking object contains some base properties that are common across all data markings. These common properties are defined in the following table.

| Property Name | Rq | Data Type | Details |
|---|---|---|---|
| **type** | Y | `string` | The type of data marking being used.<br><br>The value for this property **MUST** come from the `data-marking-type` vocabulary. |
| **name** | | `string` | A name used to identify this data marking. |
| **description** | | `string` | More details, context, and possibly an explanation about what this data marking does and tries to accomplish. |
| **created_by** | Y | `identifier` | An ID that represents the entity that created this data marking. The ID **MUST** represent a STIX 2.1+ identity object. |
| **created** | Y | `timestamp` | The time at which this data marking was originally created. The creator can use any time it deems most appropriate as the time the data marking was created, but it **MUST** be precise to the nearest millisecond (exactly three digits after the decimal place in seconds). The created property **MUST NOT** be changed. |
| **modified** | Y | `timestamp` | Data markings **MUST NOT** be versioned. This property **MUST** always equal the timestamp of the **created** property. |

| | | | |
|---|---|---|---|
| **revoked** | | `boolean` | A boolean that identifies if the creator deems that this data marking is no longer valid. The default value is `false`. Processing of data that has been previously shared with an associated data marking that is subsequently revoked is unspecified and dependent on the implementation of the consuming software. |
| **valid_from** | | `timestamp` | The time from which this data marking is considered valid.<br><br>If omitted, the data marking is valid at all times or until the timestamp defined by **valid_until**. |
| **valid_until** | | `timestamp` | The time at which this data marking **SHOULD** no longer be considered a valid marking definition.<br><br>If the **valid_until** property is omitted, then there is no constraint on the latest time for which the data marking is valid.<br><br>This property **MUST** be greater than the timestamp in the **valid_from** property if the **valid_from** property is defined. |
| **labels** | | `list` of type `string` | An optional set of terms, labels, or tags associated with this data marking. The values may be user, organization, or trust-group defined and their meaning is outside the scope of this specification. |
| **external_references** | | `list` of type `external-reference` | An optional list of external references for this data marking. |
| **marking_extensions** | | `dictionary` | This property defines the extensions that are in use on this data marking.<br><br>The key for each entry in the dictionary **MUST** be an `identifier` that uniquely identifies the extension. The id **MUST** use the object type of "extension" (see section 8.7 for more information on identifiers). The value for each key is a JSON object that can contain the structure as defined in the extension's schema location. |

## 8.2 Data Marking Type Vocabulary

**Vocabulary Name:** `data-marking-type`

This section defines the following types of data markings.

| Data Marking Type | Description |
|---|---|
| `marking-statement` | The statement marking definition defines the representation of a textual marking statement (e.g., copyright, terms of use). See section 8.3. |
| `marking-tlp` | The TLP marking definition. See section 8.4. |
| `marking-iep` | The IEP marking definition. See section 8.5. |

## 8.3 Statement Marking

The statement marking object defines the representation of a textual marking statement (e.g., copyright, terms of use, etc.). Statement markings are generally not machine-readable, and this specification does not define any behavior or actions based on their values.

| Property Name | Rq | Data Type | Details |
|---|---|---|---|
| **type** | Y | `string` | The value of this property **MUST** be `marking-statement`. |
| **statement** | Y | `string` | A statement (e.g., copyright, terms of use) applied to the content marked by this marking definition. |

**Example**

```
{
  "type": "marking-statement",
  "created_by": "identity--uuid2",
  "created": "2020-04-01TT00:00:00.000Z",
  "modified": "2020-04-01TT00:00:00.000Z",
  "statement": "Copyright 2020, Example Corp"
}
```

## 8.4 TLP Marking

The TLP marking object defines the representation of a FIRST TLP marking statement. If the TLP marking is externally defined, producers **SHOULD** use the external_refernces property of this object.

| Property Name | Rq | Data Type | Details |
|---|---|---|---|
| **type** | Y | `string` | The value of this property **MUST** be `marking-tlp`. |
| **tlp_level** | Y | `string` | The name of the TLP level taken from the following:<br><br>`TLP:RED`, `TLP:AMBER`, `TLP:GREEN`, `TLP:WHITE` |

**Example**

```
{
  "type": "marking-tlp",
```

```
  "created_by": "identity--uuid2",
  "created": "2020-04-01TT00:00:00.000Z",
  "modified": "2020-04-01TT00:00:00.000Z",
  "tlp_level": "TLP:WHITE",
}
```

## 8.5 IEP Marking

The IEP marking object defines the representation of a FIRST IEP marking statement. For more information about the properties from the IEP specification, please refer to that document [IEP].

| Property Name | Rq | Data Type | Details |
|---|---|---|---|
| **type** | Y | `string` | The value of this property **MUST** be `marking-iep`. |
| **name** | Y | `string` | The name of the IEP policy. |
| **tlp_level** | | `string` | See IEP Specification [IEP]. |
| **description** | | `string` | See IEP Specification [IEP]. |
| **iep_version** | | `string` | See IEP Specification [IEP]. |
| **start_date** | | `timestamp` | See IEP Specification [IEP]. |
| **end_date** | | `timestamp` | See IEP Specification [IEP]. |
| **encrypt_in_transit** | | `string` | See IEP Specification [IEP]. |
| **permitted_actions** | | `string` | See IEP Specification [IEP]. |
| **attribution** | | `string` | See IEP Specification [IEP]. |
| **unmodified_resale** | | `string` | See IEP Specification [IEP]. |

**Example**

```
{
  "type": "marking-iep",
  "created_by": "identity--uuid2",
  "created": "2020-04-01TT00:00:00.000Z",
  "modified": "2020-04-01TT00:00:00.000Z",
  "name": "FIRST IEP TLP-AMBER",
  "tlp_level": "TLP:AMBER"
}
```

# 9 Data Types

This section defines the common data types and objects used throughout this specification, their permitted values including vocabularies, and how they map to the MTI JSON serialization. It does not, however, define the meaning of any properties using these types. These types **MAY** be further restricted elsewhere in the specification.

## 9.1 Boolean

The `boolean` data type is a literal unquoted value of either `true` or `false` and uses the JSON true and false values [RFC8259] for serialization.

## 9.2 Civic Location

The `civic-location` data type captures civic location information and uses the JSON object type [RFC8259] for serialization.

| Property Name | Rq | Data Type | Details |
|---|---|---|---|
| **`description`** | | `string` | A detailed description about this location. |
| **`building_details`** | | `string` | Additional details about the location within a building including things like floor, room, etc. |
| **`network_details`** | | `string` | Additional details about this network location including things like wiring closet, rack number, rack location, and VLANs. |
| **`region`** | | `string` | The geographical region for this location.<br><br>The value for this property **MUST** come from the `region` vocabulary (see section 9.2.1). |
| **`country`** | | `string` | The country for this location. This property **MUST** contain a valid ISO 3166-1 ALPHA-2 Code [ISO3166-1]. |
| **`administrative_area`** | | `string` | The state, province, or other sub-national administrative area for this location.<br><br>This property **SHOULD** contain a valid ISO 3166-2 Code [ISO3166-2]. |
| **`city`** | | `string` | The city for this location. |
| **`street_address`** | | `string` | The street address for this location. This property includes all aspects or parts of the street address. For example, some addresses may have multiple lines including a mailstop or apartment number. |

| | | | |
|---|---|---|---|
| **postal_code** | | `string` | The postal code for this location. |

### 9.2.1 Region Vocabulary

A list of world regions based on the United Nations geoscheme [UNSD M49].

**Vocabulary Name:** `region`

| Vocabulary Value |
|---|
| `africa` |
| `eastern-africa` |
| `middle-africa` |
| `northern-africa` |
| `southern-africa` |
| `western-africa` |
| `americas` |
| `caribbean` |
| `central-america` |
| `latin-america-caribbean` |
| `northern-america` |
| `south-america` |
| `asia` |
| `central-asia` |
| `eastern-asia` |
| `southern-asia` |
| `south-eastern-asia` |
| `western-asia` |
| `europe` |
| `eastern-europe` |

| |
|---|
| `northern-europe` |
| `southern-europe` |
| `western-europe` |
| `oceania` |
| `antarctica` |
| `australia-new-zealand` |
| `melanesia` |
| `micronesia` |
| `polynesia` |

## 9.3 Contact Information

The `contact` information data type captures general contact information and uses the JSON object type [RFC8259] for serialization.

| Property Name | Rq | Data Type | Details |
|---|---|---|---|
| **`email`** | | `dictionary` of type `string` | An email address for this contact.<br><br>The key for each entry in the dictionary **MUST** be a `string` that uniquely identifies the contact type (e.g., the keys could be things like "work", "home", "personal", etc). The value for each key **MUST** be a `string`. |
| **`phone`** | | `dictionary` of type `string` | A phone number for this contact.<br><br>The key for each entry in the dictionary **MUST** be a `string` that uniquely identifies the contact type (e.g., the keys could be things like "work", "home", "personal", etc). The value for each key **MUST** be a `string`. |
| **`contact_details`** | | `string` | Additional contact information. |

## 9.4 Dictionary

The `dictionary` data type captures an arbitrary set of key/value pairs and uses the JSON object type [RFC8259] for serialization.

Dictionary keys:

- **MUST** be unique in each dictionary
- **MUST** be in ASCII
- **MUST** only contain the characters: a-z (lowercase ASCII), A-Z (uppercase ASCII), 0-9, and underscore (_)
- **MUST** be no longer than 250 ASCII characters in length and **SHOULD** be lowercase
- **MUST** start with a letter or the underscore character
- **MUST NOT** start with a number

The values for all keys in a dictionary **MUST** be valid property types as defined where the dictionary is used.

## 9.5 External Reference

The `external-reference` data type captures the location of information represented outside of a CACAO playbook and uses the JSON object type [RFC8259] for serialization. For example, a playbook could reference external documentation about a specific piece of malware that the playbook is trying to address. In addition to the **`name`** properties at least one of the following properties **MUST** be present: **`description`**, **`source`**, **`url`**, or **`external_id`**.

| Property Name | Rq | Data Type | Description |
|---|---|---|---|
| **`name`** | Y | `string` | The name of the author or title of the source of this external reference. |
| **`description`** | | `string` | A detailed description of this external reference. |
| **`source`** | | `string` | A textual citation of this source. The citation source **MAY** use a standard citation format like Chicago, MLA, APA, or similar style. |
| **`url`** | | `url` | A URL [RFC3986] for this external reference. |
| **`external_id`** | | `string` | An identifier used by the source to reference this content. Some organizations give names or numbers to content that they publish. This property would capture that information to help ensure that a consumer is being referred to the correct content. |
| **`reference_id`** | | `identifier` | A UUID based identifier that this content is referenced to. This property is especially useful when referencing content that already exists in a graph dataset or can be referenced via a UUID based ID. |

**Example**

```
{
  "name": "ACME Security FuzzyPanda Report",
  "description": "ACME security review of FuzzyPanda 2021",
  "source": "ACME Security Company, Solutions for FuzzyPanda 2021, January 2021.
    Available online: hxxp://www[.]example[.]com/info/fuzzypanda2021.html",
  "url": "hxxp://www[.]example[.]com/info/fuzzypanda2021.html",
```

```
  "external_id": "fuzzypanda 2021.01",
  "reference_id": "malware--2008c526-508f-4ad4-a565-b84a4949b2af"
}
```

## 9.6 GPS Location

The `gps-location` data type captures GPS location information and uses the JSON object type [RFC8259] for serialization.

| Property Name | Rq | Data Type | Details |
|---|---|---|---|
| **`latitude`** | | `string` | The GPS latitude of the target in decimal degrees. Positive numbers describe latitudes north of the equator, and negative numbers describe latitudes south of the equator. The value of this property **MUST** be less than or equal to 90.0 and greater than -90.0 (i.e., 90.0 >= x > -90.0).<br><br>If the longitude property is present, this property **MUST** be present. |
| **`longitude`** | | `string` | The GPS longitude of the target in decimal degrees. Positive numbers describe longitudes east of the prime meridian and negative numbers describe longitudes west of the prime meridian. The value of this property **MUST** be less than or equal to 180.0 and a value that is greater than -180.0 (i.e., 180.0 >= x > -180.0).<br><br>If the latitude property is present, this property **MUST** be present. |
| **`precision`** | | `string` | Defines the precision of the coordinates specified by the **`latitude`** and **`longitude`** properties. This is measured in meters. The actual target may be anywhere up to precision meters from the defined point.<br><br>If this property is not present, then the precision is unspecified.<br><br>If this property is present, the **`latitude`** and **`longitude`** properties **MUST** be present. |

## 9.7 Features

The `playbook-features` data type represents the major features and functionality of a playbook.

| Property Name | Rq | Data Type | Details |
|---|---|---|---|

| | | | |
|---|---|---|---|
| **parallel_processing** | | `boolean` | See section 4.7. |
| **if_logic** | | `boolean` | See section 4.8. |
| **while_logic** | | `boolean` | See section 4.9. |
| **switch_logic** | | `boolean` | See section 4.10. |
| **temporal_logic** | | `boolean` | See section 4.1 **delay** and **timeout** properties. |
| **data_markings** | | `boolean` | See section 2.4 and section 8. |
| **extensions** | | `boolean` | See section 7. |

## 9.8 Identifier

The `identifier` data type represents an RFC 4122-compliant UUID [RFC4122] and uses the JSON string type [RFC8259] for serialization.

An identifier uniquely identifies a CACAO object and **MAY** allow producers and consumers using the same namespace and contributing properties to generate the same identifier for the exact same content defined in the CACAO object. All identifiers **MUST** follow the form object-type--UUID, where object-type is the exact value (all type names are lowercase strings by definition) from the type property of the object being identified and where the UUID **MUST** be an RFC 4122-compliant UUID [RFC4122].

The UUID part of the identifier **MUST** be unique across all objects produced by a given producer regardless of the type identified by the object-type prefix. Meaning, a producer **MUST NOT** reuse the UUID portion of the identifier for objects of different types.

CACAO objects **SHOULD** use UUIDv5 for the UUID portion of the identifier and the UUID portion of the UUIDv5-based identifier **SHOULD** be generated according to the following rules:

- The namespace **SHOULD** be aa7caf3a-d55a-4e9a-b34e-056215fba56a
- The value of the name portion **SHOULD** be the list of contributing properties defined on each object and those properties **SHOULD** be stringified according to the JSON Canonicalization Scheme [RFC8785] to ensure a canonical representation of the JSON data
- Producers not following these rules **MUST NOT** use a namespace of aa7caf3a-d55a-4e9a-b34e-056215fba56a

## 9.9 Integer

The `integer` data type represents a whole number and uses the JSON number type [RFC7493] for serialization. Unless otherwise specified, all integers **MUST** be capable of being represented as a signed 54-bit value ([-(2**53)+1, (2**53)-1]), not a 64-bit value, as defined in [RFC7493]. When a 64-bit integer is needed in this specification, it will be encoded using the `string` data type.

## 9.10 Signature

The `signature` data type captures the actual digital signature and meta-data about the signature and uses the JSON object type [RFC8259] for serialization. See section A.2 in the appendix for a detailed example.

NOTE: All of the properties in this data type, except the **`value`** property, can be considered "claims" per JWS [RFC7515]. It is important to note that CACAO signatures do not use the JWS [RFC7515] header construct.

* One of the following properties **MUST** be populated, **`public_keys`** (preferred), **`cert_url`**, or **`thumbprint`**.

| Property Name | Rq | Data Type | Description |
|---|---|---|---|
| **`type`** | Y | `string` | The value of this property **MUST** be `signature`. |
| **`spec_version`** | Y | `string` | The version of the specification used to represent this signature. The value of this property **MUST** be "1.0" to represent the version of this specification. |
| **`id`** | Y | `identifier` | A value that uniquely identifies the signature. All signatures with the same id are considered different versions of the same signature and the version of the signature is identified by its modified property. |
| **`created_by`** | | `identifier` | An ID that represents the entity that created this signature. The ID **MUST** represent a STIX 2.1+ identity object. |
| **`created`** | Y | `timestamp` | The time at which this signature was originally created. The creator can use any time it deems most appropriate as the time the signature was created, but it **MUST** be precise to the nearest millisecond (exactly three digits after the decimal place in seconds). The created property **MUST NOT** be changed when creating a new version of the object. |
| **`modified`** | Y | `timestamp` | The time that this particular version of the signature was last modified. The creator can use any time it deems most appropriate as the time that this version of the signature was modified, but it **MUST** be precise to the nearest millisecond (exactly three digits after the decimal place in seconds). The modified property **MUST** be later than or equal to the value of the created property. |
| **`revoked`** | | `boolean` | A boolean that identifies if the signature creator deems that this signature is no longer valid. The default value is "false". |
| **`signee`** | Y | `string` | The name of the entity or organization that produced this signature. |
| **`valid_from`** | | `timestamp` | The time from which this signature is considered valid. |

| | | | |
|---|---|---|---|
| | | | If omitted, the signature is valid at all times or until the timestamp defined by **valid_until**. |
| **valid_until** | | timestamp | The time at which this signature should no longer be considered valid.<br><br>If the **valid_until** property is omitted, then there is no constraint on the latest time for which the signature is valid.<br><br>This property **MUST** be greater than the timestamp in the **valid_from** property if the **valid_from** property is defined. |
| **related_to** | Y | identifier | The CACAO Playbook that this signature is for.<br><br>The value of this property **MUST** be a CACAO Playbook. |
| **related_version** | Y | timestamp | The version of the CACAO Playbook that this signature is for.<br><br>The value of this property **MUST** be the **modified** timestamp from the CACAO Playbook that this signature is for. |
| **sha256** | Y | string | A Base64URL.Encoded SHA256 hash of the canonicalized CACAO Playbook data to be signed. See section 2.6.2 step 6. |
| **algorithm** | Y | string | This property is called "alg" in section 4.4 of [RFC7517].<br><br>This property identifies the algorithm intended for use with the key and is a case-sensitive ASCII string.<br><br>The values for this property **MUST** come from the signature-algorithm-type vocabulary. |
| **public_keys** | * | list of type string | This property is called "x5c" in section 4.7 of [RFC7517] and has the following requirements as defined in that section.<br><br>This property "contains a chain (X.509 certificate chain) of one or more PKIX certificates [RFC5280]. The certificate chain is represented as a JSON array of certificate value strings. Each string in the array is a base64-encoded (Section 4 of [RFC4648] -- not base64url-encoded) DER [ITU.X690.1994] PKIX certificate value. The PKIX certificate containing the key value **MUST** be the first certificate. This **MAY** be followed by additional certificates, with each subsequent certificate being the one used to certify the previous one. The key in the first certificate **MUST** match the public key." |
| **cert_url** | * | string | This property is called "x5u" in section 4.6 of [RFC7517] and has the following requirements as defined in that section. |

| | | | |
|---|---|---|---|
| | | | This property "is a URI [RFC3986] that refers to a resource for an X.509 public key certificate or certificate chain [RFC5280]. The identified resource **MUST** provide a representation of the certificate or certificate chain that conforms to RFC 5280 [RFC5280] in PEM-encoded form, with each certificate delimited as specified in Section 6.1 of RFC 4945 [RFC4945]. The key in the first certificate **MUST** match the public key."<br><br>"The protocol used to acquire the resource **MUST** provide integrity protection; an HTTP GET request to retrieve the certificate MUST use TLS [RFC2818] [RFC5246]; the identity of the server MUST be validated, as per Section 6 of RFC 6125 [RFC6125]." |
| **thumbprint** | * | `string` | This property is called "x5t#S256" in section 4.9 of [RFC7517] and has the following requirements as defined in that section.<br><br>This property "is a base64url-encoded SHA-256 thumbprint (a.k.a. digest, X.509 certificate SHA-256 thumbprint) of the DER encoding of an X.509 certificate [RFC5280]. Note that certificate thumbprints are also sometimes known as certificate fingerprints. The key in the certificate **MUST** match the public key."<br><br>If this property is provided then the **`created_by`** property **MUST** also be provided and the **`public_key`** or **`cert_url`** **MUST** be conveyed via that identity object. |
| **value** | Y | `string` | A Base64URL.Encoded signature as defined in JWS [RFC7515] and JCS/CT but without the JWS header or payload. This is because the algorithm and cert/key information is kept in plaintext JSON in the signature object.<br><br>NOTE: In traditional JWS / JWT this property would have three parameters separated by periods ".". The first is the JWS [RFC7515] header, the second is the payload which is omitted per JCS/CT, and the third is the actual signature. So instead of AAAA.BBBB.CCCC or AAAA..CCCC this property will only have the signature CCCC. |

### 9.10.1 Signature Algorithm Type Vocabulary

**Vocabulary Name:** `signature-algorithm-type`

The algorithm used for creating CACAO digital signatures **MUST** come from one of the following options that are defined in JWA [RFC7518] section 3.1 and [RFC8037] section 3.1 and **SHOULD** be either `ES256` or `RS256` as defined in JWA.

| Vocabulary Value | Description |
|---|---|
| `RS256` | Recommend per JWA [RFC7518]. See section 2.6 for more information. |
| `RS384` | See section 2.6 for more information. |
| `RS512` | See section 2.6 for more information. |
| `ES256` | Recommend per JWA [RFC7518]. See section 2.6 for more information. |
| `ES384` | See section 2.6 for more information. |
| `ES512` | See section 2.6 for more information. |
| `PS256` | See section 2.6 for more information. |
| `PS384` | See section 2.6 for more information. |
| `PS512` | See section 2.6 for more information. |
| `Ed25519` | See section 2.6 for more information. |
| `Ed448` | See section 2.6 for more information. |

## 9.11 String

The `string` data type represents a finite-length string of valid characters from the Unicode coded character set [ISO10646] and uses the JSON string type [RFC8259] for serialization.

## 9.12 Timestamp

The `timestamp` data type represents dates and times and uses the JSON string type [RFC8259] for serialization. The timestamp data **MUST** be a valid RFC 3339-formatted timestamp [RFC3339] using the format yyyy-mm-ddThh:mm:ss[.s+]Z where the "s+" represents 1 or more sub-second values. The brackets denote that sub-second precision is optional, and that if no digits are provided, the decimal place **MUST NOT** be present. The timestamp **MUST** be represented in the UTC+0 timezone and **MUST** use the "Z" designation to indicate this.

## 9.13 Variables

Variables can be defined and used as the playbook is executed and are stored in a `dictionary` where the key is the name of the variable and the value is a `variable` data type. Variables can represent stateful elements that may need to be captured to allow for the successful execution of the playbook. All playbook variables are mutable unless identified as a constant.

In addition to the rules for all dictionary keys, variable names:

- **MUST** be unique within the contextual scope they are declared
- **MUST** be prefixed and suffixed with **$$** for both declaration and use
- **MUST** be no longer than 250 ASCII characters in length, excluding the variable prefix and suffix $$
- **MUST** start with a letter or the underscore character after the variable prefix $$
- are case-sensitive (age, Age and AGE are three different variables) but **SHOULD** be lowercase

### 9.13.1 Variable Scope

The scope of a variable is determined by where the variable is declared. A variable may be defined globally for the entire playbook or locally within a workflow step. Variables are scoped to the object they are defined in, and any object that is used or referenced by that object. A specific variable can only be defined once, however, a variable can be assigned and used in the object where it is defined or in any object used or referenced by that object (e.g., a playbook variable can be assigned at the playbook level but also reassigned a different value within a workflow step).

### 9.13.2 Using Variables

Variables are referenced by using the key name from the dictionary prepended with two dollar signs. For example, if you had a variable in the dictionary called "ip_addresses", one could reference this in that object or a referenced object by using "$$ip_addresses$$". Variables may be passed to and from external systems provided that system supports passing of arguments when the system function is invoked or returns its results.

### 9.13.3 Variable

The `variable` data type captures variable information and uses the JSON object type [RFC8259] for serialization.

| Property Name | Rq | Data Type | Details |
|---|---|---|---|
| **`type`** | Y | `string` | The type of variable being used. The values for this property **MUST** come from the `variable-type` vocabulary. |
| **`description`** | | `string` | An optional detailed description of this variable. |
| **`value`** | | `string` | The value of the variable represented by a JSON string. The value **MAY** be populated with a string value (or number encoded as a string), an empty string "", or with the special JSON NULL value.<br><br>NOTE: An empty string is **NOT** equivalent to a JSON NULL value. An empty string means the value is known to be empty. A value of NULL means the value is unknown or undefined. |
| **`constant`** | | `boolean` | Is this variable immutable or mutable. If true, the variable is immutable and **MUST NOT** be changed. If false, the |

| | | | |
|---|---|---|---|
| | | | variable can be updated later on in the playbook. The default value is `false`. If this property is not present then then value is `false`. |
| **external** | | boolean | This property only applies to playbook scoped variables.<br><br>When set to `true` the variable declaration defines that the variable's initial value is passed into the playbook from a calling context.<br><br>When set to `false` or omitted, the variable is defined within the playbook. |

**Examples**

General Variable Example:

```
{
  "type": "playbook",
  …,
  "playbook_variables": {
    "<$$variable name$$>": {
      "type": "<variable_type>",
      "description": "<details about variable>",
      "value": "<variable_value>",
      "constant": false
      "
    }
  }
}
```

Data exfil address variable example

```
{
  "type": "playbook",
  …,
  "playbook_variables": {
    "$$data_exfil_site$$": {
      "type": "ipv4-addr",
      "description": "The IP address for the data exfiltration site",
      "value": "1.2.3.4",
      "constant": false
    }
```

## 9.13.4 Variable Type Vocabulary

**Vocabulary Name:** `variable-type`

| Vocabulary Value |
|---|
| `string` |
| `uuid` |
| `integer` |

| |
|---|
| `long` |
| `mac-addr` |
| `ipv4-addr` |
| `ipv6-addr` |
| `uri` |
| `sha256-hash` |
| `hexstring` |
| `dictionary` |

# 10 Conformance

## 10.1 CACAO Playbook Producers and Consumers

A "CACAO 1.0 Producer" is any software that can create CACAO 1.0 content and conforms to the following normative requirements:

- It **MUST** be able to create content encoded as JSON.
- All properties marked required in the property table for the CACAO object or type **MUST** be present in the created content.
- All properties **MUST** conform to the data type and normative requirements for that property.
- It **MUST** support all features listed in section 10.2, Mandatory Features.
- It **MAY** support any features listed in section 10.3, Optional Features. Software supporting an optional feature **MUST** comply with the normative requirements of that feature.
- It **MUST** support JSON as a serialization format and **MAY** support serializations other than JSON.

A "CACAO 1.0 Consumer" is any software that can consume CACAO 1.0 content and conforms to the following normative requirements:

- It **MUST** support parsing all required properties for the content that it consumes.
- It **MUST** support all features listed in section 10.2, Mandatory Features.
- It **MAY** support any features listed in section 10.3, Optional Features. Software supporting an optional feature **MUST** comply with the normative requirements of that feature.
- It **MUST** support JSON as a serialization format and **MAY** support serializations other than JSON.

## 10.2 CACAO Mandatory Features

### 10.2.1 Versioning

A CACAO 1.0 Producer or CACAO 1.0 Consumer **MUST** support versioning by following the normative requirements listed in section 2.4.

### 10.2.2 Playbooks

A CACAO 1.0 Producer or CACAO 1.0 Consumer **MUST** support the playbook object defined in this specification by following the normative requirements listed in section 3

### 10.2.3 Workflow Steps

A CACAO 1.0 Producer or CACAO 1.0 Consumer **MUST** support the workflow steps defined in this specification by following the normative requirements listed in sections 3.1 and 4.

### 10.2.4 Commands

A CACAO 1.0 Producer or CACAO 1.0 Consumer **MUST** support the command object as defined in this specification by following the normative requirements listed in sections 3.1 and 5. However, a CACAO 1.0 Producer or CACAO 1.0 Consumer **MAY** support only a subset of command object types.

### 10.2.5 Targets

A CACAO 1.0 Producer or CACAO 1.0 Consumer **MUST** support the targets defined in this specification by following the normative requirements listed in sections 3.1 and 6.

## 10.3 CACAO Optional Features

### 10.3.1 Data Markings

A CACAO 1.0 Producer or CACAO 1.0 Consumer **MAY** support Data Markings. Software that supports Data Markings **MUST** follow the normative requirements listed in sections 2.5, 3.1, and 8.

### 10.3.2 Extensions

A CACAO 1.0 Producer or CACAO 1.0 Consumer **MAY** support Extensions. Software that supports Extensions **MUST** follow the normative requirements listed in sections 3.1 and 7.

#### 10.3.2.1 Requirements for Extension Properties

- A CACAO Playbook **MAY** have any number of Extensions containing one or more properties.
- Extension property names **MUST** be in ASCII and **MUST** only contain the characters a–z (lowercase ASCII), 0–9, and underscore (_).
- Extension property names **MUST** have a minimum length of 3 ASCII characters.
- Extension property names **MUST** be no longer than 250 ASCII characters in length.
- Extension properties **SHOULD** only be used when there are no existing properties defined by the CACAO Playbook specification that fulfils that need.

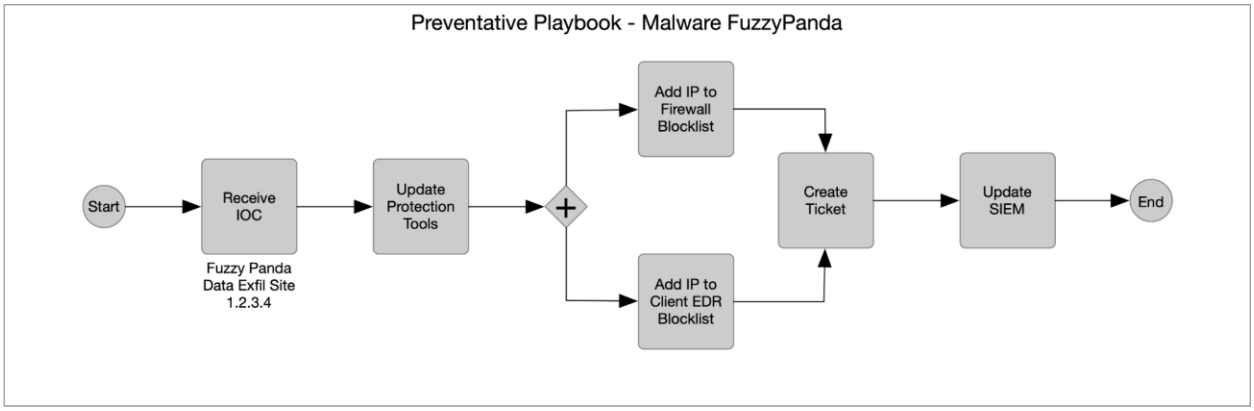
### 10.3.3 Digital Signatures

A CACAO 1.0 Producer or CACAO 1.0 Consumer **MAY** support Digital Signatures. Software that supports Digital Signatures **MUST** follow the normative requirements listed in sections 2.6, 3.1 and 9.10.

# Appendix A. Examples

The examples in this section are based on various scenarios and are included to help readers understand how CACAO playbooks can be developed. In these examples it is common to not actually use UUIDs, but rather simple IDs to make it easier for visual human inspection. These simple IDs will have a form of "uuid--1". In some of these examples we have elected to show all optional properties and all properties that have defaults. This is done to help implementers fully understand the schema.

## A.1 Playbook Example 1




```
{
  "type": "playbook",
  "spec_version": "1.0",
  "id": "playbook--6b74199d-42a6-43a1-99cb-75d52207a778",
  "name": "Prevent FuzzyPanda Malware",
  "description": "This playbook will block traffic to the FuzzyPanda data exfil site",
  "playbook_types": [
    "prevention"
  ],
  "created_by": "identity--uuid2",
  "created": "2021-04-19T23:32:24.399Z",
  "modified": "2021-04-19T23:32:24.399Z",
  "valid_from": "2021-04-19T23:32:24.39964Z",
  "valid_until": "2021-12-31T23:59:59.999999Z",
  "derived_from": [
    "playbook--uuid50"
  ],
  "priority": 3,
  "severity": 70,
  "impact": 5,
  "labels": [
    "malware",
    "fuzzypanda",
    "apt"
  ],
  "external_references": [
    {
      "name": "ACME Security FuzzyPanda Report",
```

```
      "description": "ACME security review of FuzzyPanda 2021",
      "source": "ACME Security Company, Solutions for FuzzyPanda 2021, January 2021. Available
online: http://www.example.com/info/fuzzypanda2021.html",
      "url": "http://www.example.com/info/fuzzypanda2020.html",
      "external_id": "fuzzypanda 2021.01",
      "reference_id": "malware--2008c526-508f-4ad4-a565-b84a4949b2af"
    }
  ],
  "features": {
    "parallel_processing": true,
    "data_markings": true
  },
  "markings": [
    "marking-statement--16a48f6b-ab42-4f99-ba9b-8b21e1225836",
    "marking-tlp--a099a2eb-1113-4368-9b17-d7ef75841239"
  ],
  "workflow_start": "start--7269bda2-e651-44d3-9fe5-aa7e88484b93",
  "workflow": {
    "start--7269bda2-e651-44d3-9fe5-aa7e88484b93": {
      "type": "start",
      "on_completion": "single--a13c8450-2bd1-4a2b-9241-cf4f7e9f48cb"
    },
    "single--a13c8450-2bd1-4a2b-9241-cf4f7e9f48cb": {
      "type": "single",
      "name": "Receive IOC",
      "description": "Get FuzzyPanda Data Exfil Site IP Address of 1.2.3.4",
      "on_completion": "parallel--054c7e3a-20e7-4fdf-a95f-6c6e401c65c3",
      "commands": [
        {
          "type": "manual",
          "command": "Get IOC from threat feed"
        }
      ]
    },
    "parallel--054c7e3a-20e7-4fdf-a95f-6c6e401c65c3": {
      "type": "parallel",
      "name": "Update Protection Tools",
      "description": "This step will update the firewall and client EDR in parallel",
      "next_steps": [
        "single--8c46cab0-46a3-48f4-b4bb-9643dcfaf642",
        "single--3d930f08-e22c-4dd4-996f-61f2d022121c"
      ]
    },
    "single--8c46cab0-46a3-48f4-b4bb-9643dcfaf642": {
      "type": "single",
      "name": "Add IP to Firewall Blocklist",
      "description": "This step will add the IP address of the FuzzyPanda data exfil site to
the firewall",
      "on_completion": "single--d5780323-5107-4cd0-bac4-6553c9d90c8e",
      "commands": [
        {
          "type": "manual",
          "command": "Open firewall console and add 1.2.3.4 to the firewall blocking policy"
```

```
        }
      ]
    },
    "single--3d930f08-e22c-4dd4-996f-61f2d022121c": {
      "type": "single",
      "name": "Add IP to Client EDR Blocklist",
      "description": "This step will add the IP address of the FuzzyPanda data exfil site to
the client EDR solution",
      "on_completion": "single--d5780323-5107-4cd0-bac4-6553c9d90c8e",
      "commands": [
        {
          "type": "manual",
          "command": "Open EDR console and add 1.2.3.4 to the blocking policy"
        }
      ]
    },
    "single--d5780323-5107-4cd0-bac4-6553c9d90c8e": {
      "type": "single",
      "name": "Create Ticket",
      "description": "This step will create a ticket for this issue",
      "on_completion": "single--33dc512c-263d-4f8a-a07d-cfe9f6d6ed26",
      "commands": [
        {
          "type": "manual",
          "command": "Open case management tool and create a ticket with the details of what
was done"
        }
      ]
    },
    "single--33dc512c-263d-4f8a-a07d-cfe9f6d6ed26": {
      "type": "single",
      "name": "Update SIEM",
      "description": "This step will update the SIEM to look for traffic attempts to the
FuzzyPanda data exfil site",
      "on_completion": "end--6d43fbf3-54b3-432a-978b-e2b96647b786",
      "commands": [
        {
          "type": "manual",
          "command": "Open SIEM solution and add rule to look for 1.2.3.4"
        }
      ]
    },
    "end--6d43fbf3-54b3-432a-978b-e2b96647b786": {
      "type": "end"
    }
  },
  "data_marking_definitions": {
    "marking-statement--16a48f6b-ab42-4f99-ba9b-8b21e1225836": {
      "type": "marking-statement",
      "spec_version": "1.0",
      "id": "marking-statement--16a48f6b-ab42-4f99-ba9b-8b21e1225836",
      "created": "2021-04-19T23:32:24.399Z",
      "modified": "2021-04-19T23:32:24.399Z",
```

```
      "statement": "Copyright 2021 ACME Security Company"
    },
    "marking-tlp--a099a2eb-1113-4368-9b17-d7ef75841239": {
      "type": "marking-tlp",
      "spec_version": "1.0",
      "id": "marking-tlp--a099a2eb-1113-4368-9b17-d7ef75841239",
      "created": "2021-04-19T23:32:24.399Z",
      "modified": "2021-04-19T23:32:24.399Z",
      "tlp_level": "TLP-GREEN"
    }
  }
}
```

## A.2 Playbook Signature

---

Step 0: Given the following public and private keys

---

```
-----BEGIN PRIVATE KEY-----
MIIEvAIBADANBgkqhkiG9w0BAQEFAASCBKYwggSiAgEAAoIBAQCm0pnIU9K2+Y6VvRaKE4GGUdSv
rAUMcL61buEkC519NDmYdlCkHw+gzPTu51kD50bx2FQg+SZeWnVOBER5hMd2HGG5/TL8aFulm/kk
9gfHfBq074dY7apiSNEwRytaE8x1pWRL9d7+WJoxyjDiNihZoWbxWht5izJUPZtZZ3KXiOhMQROb
VnjtGed9HXmRWFW51WsPMQWYziddX/p2YiDXzhEkTiG23AEXFHypkJALBOImayjInF9RHQazh48p
zmwHQ1OSYVlzmSVBKK13rtEmfaV2FuoTsSkOXheUi35TIsmbWC4IGW2JrwCCR7t1e6GkHuFDosnB
jgSPO2GQnwg/AgMBAAECggEAKT6KTNAEmb5rdTPxvaOC832J0wD5opDBZcQLH8lLX6go0Tv3Rgxz
5bKmn+ZMyL1GegadDiXrSYqd0/MUJuMgGWB8/OnP0D3Q4soEOBIn7DcPt0o9MUxZQsF0DraZzkR0
2WVRvcIFJucrAEJYAaWYJkjUVbmMb2ltwQwWO21rFHGbpE73nsfr/oAWsZEvKsQZoYm4fh5jVI5+
wKyRnKaN1uqAcNgj75cdywCHBVwgEefEgOPM77CDMH0+JumSirQiBfR35+HWRwHwpm09wI6Aqtvg
y5bzxvLDDRgrhX4LCPtUHGrUXNJHRKYiHQX6P6bIVuBrHV6VFpyS+5weu0w6kQKBgQDQo4QeLtO7
S3KH8UL3lX4lhH1K7/Q99uBHmvLXdiDkHjLbBbh0JfrHgHtnK9bvJ2GvVcwhI9fTiO1p1o5RM5jb
iVUSCS91sLcTPFv8X83sExBZnrvlSlb/va+4yW+Lzvr6ZiDlZYsVRNvNAHUTojHRCOH2P4eX1+ql
5P4FMdfvSQKBgQDMsQ4LBpxjD9KdDzJzw9a0xbL47QdCeZBqNUy6MvwLE0+KsF+prvoigNZCaTcJ
2FfoPxpE3/o0A/byCTuDkfddrd/hcAO0gd1R9CYJDXJfnIbZfheUmHW7ShbXyqhpqQKVjzH+jnLq
VjbGD6tz3dN+AwNgULD/vvwXM2TWpu9TRwKBgGkPPdMZD2NLzaNouKkFbR0lRxY6GEovi6Zi/w/C
GzPjhQZHLifGjC5zozBDohqRQR5SXNT/QInzdGGMOePn0HwT/nNzjqN71eRoy4UdFQtgWiZWyRTf
x0lGUjsBrBrBoh3+2WfKJywRnYDwTwQQ83boOyiNuxCaGD1rPwKMo8iJAoGAPIePE4uc615edbts
u/cJouNjjWDqaKnyHrYsPlOdXNkVCHonj9ICffmDYpgignLLbA5dAkkJgCA8Ak7gnoOnlrg4ID4z
mklc3UNJjBvB2qw65E35QyPijMPYBXAUZUppTTjPG+ub59ge0msH1Hegdv8FHJJABSDBA0tbYm5z
DzkCgYA9/0KtWKFMhF3v01L54AXF5b15RroBhZAfzI1U0wPO4J6Tz+1KqmtrwHTBPI36nzITIhlM
hcoTsMRMgnv0NHzxlcQQmAy3foFBFOyHXql3hPtWbEViB5jQs4cP5ts1oivVhrEtrrE51TG4V/Ef
fD1JKiHl7MECYEMyBz31PsRCuw==
-----END PRIVATE KEY-----

-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAptKZyFPStvmOlb0WihOBhlHUr6wFDHC+
tW7hJAudfTQ5mHZQpB8PoMz07udZA+dG8dhUIPkmXlp1TgREeYTHdhxhuf0y/GhbpZv5JPYHx3wa
tO+HWO2qYkjRMEcrWhPMdaVkS/Xe/liaMcow4jYoWaFm8VobeYsyVD2bWWdyl4joTEETm1Z47Rnn
fR15kVhVudVrDzEFmM4nXV/6dmIg184RJE4httwBFxR8qZCQCwTiJmsoyJxfUR0Gs4ePKc5sB0NT
kmFZc5klQSitd67RJn2ldhbqE7EpDl4XlIt+UyLJm1guCBltia8Agke7dXuhpB7hQ6LJwY4Ejzth
kJ8IPwIDAQAB
-----END PUBLIC KEY-----
```

### A.2.1 Signing a Playbook

---

Step 1.0: Create or receive a JSON playbook object to sign

---

```
{
  "type": "playbook",
  "spec_version": "1.0",
  "id": "playbook--a0777575-5c4c-4710-9f01-15776103837f",
  "name": "Playbook 1",
  "created": "2021-01-25T20:31:31.319Z",
  "modified": "2021-01-25T20:31:31.319Z",
  "signatures": [
    {
      "type": "signature",
      "spec_version": "1.0",
      "id": "signature--uuid1",
      "created_by": "identity--uuid2",
      "created": "2021-01-25T20:31:31.319516Z",
      "modified": "2021-01-25T20:31:31.319516Z",
      "signee": "Existing Example Company",
      "valid_from": "2021-01-25T20:31:31.319516Z",
      "valid_until": "2022-01-01T12:12:12.123456Z",
      "related_to": "playbook--a0777575-5c4c-4710-9f01-15776103837f",
      "related_version": "2021-01-25T20:31:31.319Z",
      "sha256": "hHuhBwKscfqvLC3y2FfZtHi3DNkzE0o8kE8eE6x50pM",
      "algorithm": "RS256",
      "public_keys": [
        "some public key"
      ],
      "value": "some signature"
    }
  ]
}
```

---

Step 1.1: Remove existing signature objects contained in the playbook's signatures property before computing the hash

---

```
{
  "type": "playbook",
  "spec_version": "1.0",
  "id": "playbook--a0777575-5c4c-4710-9f01-15776103837f",
  "name": "Playbook 1",
  "created": "2021-01-25T20:31:31.319Z",
  "modified": "2021-01-25T20:31:31.319Z"
}
```

---

Step 1.2: Create JCS [RFC8785] canonical version of the playbook from step 1.1

---

{"created":"2021-01-25T20:31:31.319Z","id":"playbook--a0777575-5c4c-4710-9f01-15776103837f","modified":"2021-01-25T20:31:31.319Z","name":"Playbook 1","spec_version":"1.0","type":"playbook"}

---

Step 1.3: Create SHA256 (in hex) of canonical version of playbook from step 1.2

---

847ba10702ac71faaf2c2df2d857d9b478b70cd933134a3c904f1e13ac79d293

---

Step 1.4: Create base64URL.encoded version of the SHA256 hash from step 1.3 and remove any padding

---

hHuhBwKscfqvLC3y2FfZtHi3DNkzE0o8kE8eE6x50pM

---

Step 2.0: Create a signature object and set the SHA256 string property to the string value of the b64 hash of the playbook from step 1.4

---


```
{
  "type": "signature",
  "spec_version": "1.0",
  "id": "signature--af892292-c4b4-47eb-9be6-4897ff4b9388",
  "created_by": "identity--uuid2",
  "created": "2021-01-25T20:31:31.319516Z",
  "modified": "2021-01-25T20:31:31.319516Z",
  "signee": "ACME Cyber Company",
  "valid_from": "2021-01-25T20:31:31.319516Z",
  "valid_until": "2022-01-01T12:12:12.123456Z",
  "related_to": "playbook--a0777575-5c4c-4710-9f01-15776103837f",
  "related_version": "2021-01-25T20:31:31.319Z",
  "sha256": "hHuhBwKscfqvLC3y2FfZtHi3DNkzE0o8kE8eE6x50pM",
  "algorithm": "RS256",
  "public_keys": [

"MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAptKZyFPStvmOlb0WihOBhlHUr6wFDHC+tW7hJAudfTQ5mHZQpB8PoMz07udZA+dG8dhUIPkmXlp1TgREeYTHdhxhuf0y/GhbpZv5JPYHx3watO+HWO2qYkjRMEcrWhPMdaVkS/Xe/liaMcow4jYoWaFm8VobeYsyVD2bWWdyl4joTEETm1Z47RnnfR15kVhVudVrDzEFmM4nXV/6dmIg184RJE4httwBFxR8qZCQCwTiJmsoyJxfUR0Gs4ePKc5sB0NTkmFZc5klQSitd67RJn2ldhbqE7EpDl4XlIt+UyLJm1guCBltia8Agke7dXuhpB7hQ6LJwY4EjzthkJ8IPwIDAQAB"
  ]
}
```


---

Step 2.1: Create JCS canonical version of signature from step 2.0

---

{"algorithm":"RS256","created":"2021-01-25T20:31:31.319516Z","created_by":"identity--uuid2","id":"signature--af892292-c4b4-47eb-9be6-4897ff4b9388","modified":"2021-01-25T20:31:31.319516Z","public_keys":["MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAptKZyFPStvmOlb0WihOBhlHUr6wFDHC+tW7hJAudfTQ5mHZQpB8PoMz07udZA+dG8dhUIPkmXlp1TgREeYTHdhxhuf0y/GhbpZv5JPYHx3w

atO+HWO2qYkjRMEcrWhPMdaVkS/Xe/liaMcow4jYoWaFm8VobeYsyVD2bWWdyl4joTEETm1Z47RnnfR15kVhVudVrDzEFm
M4nXV/6dmIg184RJE4httwBFxR8qZCQCwTiJmsoyJxfUR0Gs4ePKc5sB0NTkmFZc5klQSitd67RJn2ldhbqE7EpDl4XlIt
+UyLJm1guCBltia8Agke7dXuhpB7hQ6LJwY4EjzthkJ8IPwIDAQAB"],"related_to":"playbook--a0777575-5c4c-
4710-9f01-15776103837f","related_version":"2021-01-
25T20:31:31.319Z","sha256":"hHuhBwKscfqvLC3y2FfZtHi3DNkzE0o8kE8eE6x50pM","signee":"ACME Cyber
Company","spec_version":"1.0","type":"signature","valid_from":"2021-01-
25T20:31:31.319516Z","valid_until":"2022-01-01T12:12:12.123456Z"}

---

Step 2.2: Create base64URL.encoded version of the JCS signature from step 2.1

---

eyJhbGdvcml0aG0iOiJSUzI1NiIsImNyZWF0ZWQiOiIyMDIxLTAxLTI1VDIwOjMxOjMxLjMxOTUxNloiLCJjcmVhdGVkX2
J5IjoiaWRlbnRpdHktLXV1aWQyIiwiaWQiOiJzaWduYXR1cmUtLWFmODkyMjkyLWM0YjQtNDdlYi05YmU2LTQ4OTdmZjRi
OTM4OCIsIm1vZGlmaWVkIjoiMjAyMS0wMS0yNVQyMDozMTozMS4zMTk1MTZaIiwicHVibGljX2tleXMiOlsiTUlJQklqQU
5CZ2txaGtpRzl3MEJBUUVGQUFPQ0FROEFNSUlCQ2dLQ0FRRUFwdEtaeUZQU3R2bU9sYjBXaWhPQmhsSFVyNndGREhDK3RX
N2hKQXVkZlRRNW1IWlFwQjhQb016MDd1ZFpBK2RHOGRoVUlQa21YbHAxVGdSRWVZVEhkaHhodWYweS9HaGJwWnY1SlBZSH
gzd2F0TytIV08ycVlralJNRWNyV2hQTWRhVmtTL1hlL2xpYU1jb3c0allvV2FGbThWb2JlWXN5VkQyYldXZHlsNGpvVEVF
VG0xWjQ3Um5uZlIxNWtWaFZ1ZFZyRHpFRm1NNG5YVi82ZG1JZzE4NFJKRTRodHR3QkZ4UjhxWkNRQ3dUaUptc295SnhmVV
IwR3M0ZVBLYzVzQjBOVGttRlpjNWtsUVNpdGQ2N1JKbjJsZGhicUU3RXBEbDRYbEl0K1V5TEptMWd1Q0JsdGlhOEFna2U3
ZFh1aHBCN2hRNkxKd1k0RWp6dGhrSjhJUHdJREFRQUIiXSwicmVsYXRlZF90byI6InBsYXlib29rLS1hMDc3NzU3NS01Yz
RjLTQ3MTAtOWYwMS0xNTc3NjEwMzgzN2YiLCJyZWxhdGVkX3ZlcnNpb24iOiIyMDIxLTAxLTI1VDIwOjMxOjMxLjMxOVoi
LCJzaGEyNTYiOiJoSHVoQndLc2NmcXZMQzN5MkZmWnRIaTNETmt6RTBvOGtFOGVFNng1MHBNIiwic2lnbmVlIjoiQUNNRS
BDeWJlciBDb21wYW55Iiwic3BlY192ZXJzaW9uIjoiMS4wIiwidHlwZSI6InNpZ25hdHVyZSIsInZhbGlkX2Zyb20iOiIy
MDIxLTAxLTI1VDIwOjMxOjMxLjMxOTUxNloiLCJ2YWxpZF91bnRpbCI6IjIwMjItMDEtMDFUMTI6MTI6MTIuMTIzNDU2Wi
J9

---

Step 3.0: Sign the data from step 2.2 using the algorithm defined in the signature object and base64URL.encode it (RS256)

---

Signature: lfmqOpMlNcUb4coQ9n6RhFqKCLCocqTEdyb9S4t5F4INN9Q4pXPAUpd28hnVS-D3BgmPACq6dQgNY1nXnU-
QqcChlVDGeliRTu5OLULrBCkQTZ8OcAhyUprXYP4vhzN81w-
eSmQz9urEGe98o2RbhLbZCrEuBUqgvmPdsu5cUnJr9wdkMHwoToS-
rbc_xuWHQAFzqi0YarCAfbPop0jDQxO8KNDFIoy98mjbL2FXv0Y4GQOSZaJNgZpxdSmgqpQfF5vxOEzQpwirvoUkjGydro
Jsim7XhAsQwiQwEuegl0GzawhIODVMVz2ZIW0jByUnCH2G21oa1mlA2sX5nciGKw

---

Step 4.0: Append the new b64 digital signature from step 3.0 to the signatures property (with existing signatures, if any) of the playbook itself. As stated in section 3.1 the modified property on the playbook does not change when adding a signature. Meaning, adding a signature does not constitute a revision of the object.

---


```
{
  "type": "playbook",
  "spec_version": "1.0",
  "id": "playbook--a0777575-5c4c-4710-9f01-15776103837f",
  "name": "Playbook 1",
  "created": "2021-01-25T20:31:31.319Z",
  "modified": "2021-01-25T20:31:31.319Z",
  "signatures": [
    {
      "type": "signature",
```

```
      "spec_version": "1.0",
      "id": "signature--uuid1",
      "created_by": "identity--uuid2",
      "created": "2021-01-25T20:31:31.319516Z",
      "modified": "2021-01-25T20:31:31.319516Z",
      "signee": "Existing Example Company",
      "valid_from": "2021-01-25T20:31:31.319516Z",
      "valid_until": "2022-01-01T12:12:12.123456Z",
      "related_to": "playbook--a0777575-5c4c-4710-9f01-15776103837f",
      "related_version": "2021-01-25T20:31:31.319Z",
      "sha256": "hHuhBwKscfqvLC3y2FfZtHi3DNkzE0o8kE8eE6x50pM",
      "algorithm": "RS256",
      "public_keys": [
        "some public key"
      ],
      "value": "some signature"
    },
    {
      "type": "signature",
      "spec_version": "1.0",
      "id": "signature--af892292-c4b4-47eb-9be6-4897ff4b9388",
      "created_by": "identity--uuid2",
      "created": "2021-01-25T20:31:31.319516Z",
      "modified": "2021-01-25T20:31:31.319516Z",
      "signee": "ACME Cyber Company",
      "valid_from": "2021-01-25T20:31:31.319516Z",
      "valid_until": "2022-01-01T12:12:12.123456Z",
      "related_to": "playbook--a0777575-5c4c-4710-9f01-15776103837f",
      "related_version": "2021-01-25T20:31:31.319Z",
      "sha256": "hHuhBwKscfqvLC3y2FfZtHi3DNkzE0o8kE8eE6x50pM",
      "algorithm": "RS256",
      "public_keys": [

"MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAptKZyFPStvmOlb0WihOBhlHUr6wFDHC+tW7hJAudfTQ5mHZQp
B8PoMz07udZA+dG8dhUIPkmXlp1TgREeYTHdhxhuf0y/GhbpZv5JPYHx3watO+HWO2qYkjRMEcrWhPMdaVkS/Xe/liaMco
w4jYoWaFm8VobeYsyVD2bWWdyl4joTEETm1Z47RnnfR15kVhVudVrDzEFmM4nXV/6dmIg184RJE4httwBFxR8qZCQCwTiJ
msoyJxfUR0Gs4ePKc5sB0NTkmFZc5klQSitd67RJn2ldhbqE7EpDl4XlIt+UyLJm1guCBltia8Agke7dXuhpB7hQ6LJwY4
EjzthkJ8IPwIDAQAB"
      ],
      "value": "lfmqOpMlNcUb4coQ9n6RhFqKCLCocqTEdyb9S4t5F4INN9Q4pXPAUpd28hnVS-
D3BgmPACq6dQgNY1nXnU-QqcChlVDGeliRTu5OLULrBCkQTZ8OcAhyUprXYP4vhzN81w-
eSmQz9urEGe98o2RbhLbZCrEuBUqgvmPdsu5cUnJr9wdkMHwoToS-
rbc_xuWHQAFzqi0YarCAfbPop0jDQxO8KNDFIoy98mjbL2FXv0Y4GQOSZaJNgZpxdSmgqpQfF5vxOEzQpwirvoUkjGydro
Jsim7XhAsQwiQwEuegl0GzawhIODVMVz2ZIW0jByUnCH2G21oa1mlA2sX5nciGKw"
    }
  ]
}
```


## A.2.2 Verifying a Playbook

---

Step 1.0: Receive a JSON playbook object to verify

```
{
  "type": "playbook",
  "spec_version": "1.0",
  "id": "playbook--a0777575-5c4c-4710-9f01-15776103837f",
  "name": "Playbook 1",
  "created": "2021-01-25T20:31:31.319Z",
  "modified": "2021-01-25T20:31:31.319Z",
  "signatures": [
    {
      "type": "signature",
      "spec_version": "1.0",
      "id": "signature--af892292-c4b4-47eb-9be6-4897ff4b9388",
      "created_by": "identity--uuid2",
      "created": "2021-01-25T20:31:31.319516Z",
      "modified": "2021-01-25T20:31:31.319516Z",
      "signee": "ACME Cyber Company",
      "valid_from": "2021-01-25T20:31:31.319516Z",
      "valid_until": "2022-01-01T12:12:12.123456Z",
      "related_to": "playbook--a0777575-5c4c-4710-9f01-15776103837f",
      "related_version": "2021-01-25T20:31:31.319Z",
      "sha256": "hHuhBwKscfqvLC3y2FfZtHi3DNkzE0o8kE8eE6x50pM",
      "algorithm": "RS256",
      "public_keys": [

"MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAptKZyFPStvmOlb0WihOBhlHUr6wFDHC+tW7hJAudfTQ5mHZQp
B8PoMz07udZA+dG8dhUIPkmXlp1TgREeYTHdhxhuf0y/GhbpZv5JPYHx3watO+HWO2qYkjRMEcrWhPMdaVkS/Xe/liaMco
w4jYoWaFm8VobeYsyVD2bWWdyl4joTEETm1Z47RnnfR15kVhVudVrDzEFmM4nXV/6dmIg184RJE4httwBFxR8qZCQCwTiJ
msoyJxfUR0Gs4ePKc5sB0NTkmFZc5klQSitd67RJn2ldhbqE7EpDl4XlIt+UyLJm1guCBltia8Agke7dXuhpB7hQ6LJwY4
EjzthkJ8IPwIDAQAB"
      ],
      "value": "lfmqOpMlNcUb4coQ9n6RhFqKCLCocqTEdyb9S4t5F4INN9Q4pXPAUpd28hnVS-
D3BgmPACq6dQgNY1nXnU-QqcChlVDGeliRTu5OLULrBCkQTZ8OcAhyUprXYP4vhzN81w-
eSmQz9urEGe98o2RbhLbZCrEuBUqgvmPdsu5cUnJr9wdkMHwoToS-
rbc_xuWHQAFzqi0YarCAfbPop0jDQxO8KNDFIoy98mjbL2FXv0Y4GQOSZaJNgZpxdSmgqpQfF5vxOEzQpwirvoUkjGydro
Jsim7XhAsQwiQwEuegl0GzawhIODVMVz2ZIW0jByUnCH2G21oa1mlA2sX5nciGKw"
    }
  ]
}
```

---

Step 1.1: Capture the signature object from step 1.0

---

```
{
  "type": "signature",
  "spec_version": "1.0",
  "id": "signature--af892292-c4b4-47eb-9be6-4897ff4b9388",
  "created_by": "identity--uuid2",
  "created": "2021-01-25T20:31:31.319516Z",
  "modified": "2021-01-25T20:31:31.319516Z",
  "signee": "ACME Cyber Company",
  "valid_from": "2021-01-25T20:31:31.319516Z",
```

```
  "valid_until": "2022-01-01T12:12:12.123456Z",
  "related_to": "playbook--a0777575-5c4c-4710-9f01-15776103837f",
  "related_version": "2021-01-25T20:31:31.319Z",
  "sha256": "hHuhBwKscfqvLC3y2FfZtHi3DNkzE0o8kE8eE6x50pM",
  "algorithm": "RS256",
  "public_keys": [

"MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAptKZyFPStvmOlb0WihOBhlHUr6wFDHC+tW7hJAudfTQ5mHZQp
B8PoMz07udZA+dG8dhUIPkmXlp1TgREeYTHdhxhuf0y/GhbpZv5JPYHx3watO+HWO2qYkjRMEcrWhPMdaVkS/Xe/liaMco
w4jYoWaFm8VobeYsyVD2bWWdyl4joTEETm1Z47RnnfR15kVhVudVrDzEFmM4nXV/6dmIg184RJE4httwBFxR8qZCQCwTiJ
msoyJxfUR0Gs4ePKc5sB0NTkmFZc5klQSitd67RJn2ldhbqE7EpDl4XlIt+UyLJm1guCBltia8Agke7dXuhpB7hQ6LJwY4
EjzthkJ8IPwIDAQAB"
  ],
  "value": "lfmqOpMlNcUb4coQ9n6RhFqKCLCocqTEdyb9S4t5F4INN9Q4pXPAUpd28hnVS-
D3BgmPACq6dQgNY1nXnU-QqcChlVDGeliRTu5OLULrBCkQTZ8OcAhyUprXYP4vhzN81w-
eSmQz9urEGe98o2RbhLbZCrEuBUqgvmPdsu5cUnJr9wdkMHwoToS-
rbc_xuWHQAFzqi0YarCAfbPop0jDQxO8KNDFIoy98mjbL2FXv0Y4GQOSZaJNgZpxdSmgqpQfF5vxOEzQpwirvoUkjGydro
Jsim7XhAsQwiQwEuegl0GzawhIODVMVz2ZIW0jByUnCH2G21oa1mlA2sX5nciGKw"
}
```


---

Step 1.2: Parse the public key from step 1.1

---

public key is of type RSA:


```
&{210594097065308710271599231525752260161004913040350793512639218334427419314517401467120719131287943231450667055098807729668988029511795426624160485140091001692043195821864310038554400317125046820270078735768758789009684616094818765170689587820073644804049723577150603020849027713765014693711708754991948648319810636105460103915672966682985955471902434903607422718833473024840908621915396045359684024600617312172654636786668094609933114747322870580558784915995973610952379783769865223654952652255806506293732116022445160383228653225313980871857493085091972257765692391785178663028942231846833310236726307000089071473276553 65537}
```


Public RSA Key E: `65537`

Public RSA Key N:


```
210594097065308710271599231525752260161004913040350793512639218334427419314517401467120719131287943231450667055098807729668988029511795426624160485140091001692043195821864310038554400317125046820270078735768758789009684616094818765170689587820073644804049723577150603020849027713765014693711708754991948648319810636105460103915672966682985955471902434903607422718833473024840908621915396045359684024600617312172654636786668094609933114747322870580558784915995973610952379783769865223654952652255806506293732116022445160383228653225313980871857493085091972257765692391785178663028942231846833310236726307000089071473276553
```


---

Step 1.3: Remove the digital signature from the signature object from step 1.1

---


```
{
  "type": "signature",
  "spec_version": "1.0",
  "id": "signature--af892292-c4b4-47eb-9be6-4897ff4b9388",
  "created_by": "identity--uuid2",
  "created": "2021-01-25T20:31:31.319516Z",
  "modified": "2021-01-25T20:31:31.319516Z",
  "signee": "ACME Cyber Company",
```

```
  "valid_from": "2021-01-25T20:31:31.319516Z",
  "valid_until": "2022-01-01T12:12:12.123456Z",
  "related_to": "playbook--a0777575-5c4c-4710-9f01-15776103837f",
  "related_version": "2021-01-25T20:31:31.319Z",
  "sha256": "hHuhBwKscfqvLC3y2FfZtHi3DNkzE0o8kE8eE6x50pM",
  "algorithm": "RS256",
  "public_keys": [

"MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAptKZyFPStvmOlb0WihOBhlHUr6wFDHC+tW7hJAudfTQ5mHZQp
B8PoMz07udZA+dG8dhUIPkmXlp1TgREeYTHdhxhuf0y/GhbpZv5JPYHx3watO+HWO2qYkjRMEcrWhPMdaVkS/Xe/liaMco
w4jYoWaFm8VobeYsyVD2bWWdyl4joTEETm1Z47RnnfR15kVhVudVrDzEFmM4nXV/6dmIg184RJE4httwBFxR8qZCQCwTiJ
msoyJxfUR0Gs4ePKc5sB0NTkmFZc5klQSitd67RJn2ldhbqE7EpDl4XlIt+UyLJm1guCBltia8Agke7dXuhpB7hQ6LJwY4
EjzthkJ8IPwIDAQAB"
  ]
}
```

Full Signature: `lfmqOpMlNcUb4coQ9n6RhFqKCLCocqTEdyb9S4t5F4INN9Q4pXPAUpd28hnVS-D3BgmPACq6dQgNY1nXnU-QqcChlVDGeliRTu5OLULrBCkQTZ8OcAhyUprXYP4vhzN81w-eSmQz9urEGe98o2RbhLbZCrEuBUqgvmPdsu5cUnJr9wdkMHwoToS-rbc_xuWHQAFzqi0YarCAfbPop0jDQxO8KNDFIoy98mjbL2FXv0Y4GQOSZaJNgZpxdSmgqpQfF5vxOEzQpwirvoUkjGydroJsim7XhAsQwiQwEuegl0GzawhIODVMVz2ZIW0jByUnCH2G21oa1mlA2sX5nciGKw`

---

Step 1.4: Create canonical version of signature object from step 1.3

---

```
{"algorithm":"RS256","created":"2021-01-25T20:31:31.319516Z","created_by":"identity--uuid2","id":"signature--af892292-c4b4-47eb-9be6-4897ff4b9388","modified":"2021-01-25T20:31:31.319516Z","public_keys":["MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAptKZyFPStvmOlb0WihOBhlHUr6wFDHC+tW7hJAudfTQ5mHZQpB8PoMz07udZA+dG8dhUIPkmXlp1TgREeYTHdhxhuf0y/GhbpZv5JPYHx3watO+HWO2qYkjRMEcrWhPMdaVkS/Xe/liaMcow4jYoWaFm8VobeYsyVD2bWWdyl4joTEETm1Z47RnnfR15kVhVudVrDzEFmM4nXV/6dmIg184RJE4httwBFxR8qZCQCwTiJmsoyJxfUR0Gs4ePKc5sB0NTkmFZc5klQSitd67RJn2ldhbqE7EpDl4XlIt+UyLJm1guCBltia8Agke7dXuhpB7hQ6LJwY4EjzthkJ8IPwIDAQAB"],"related_to":"playbook--a0777575-5c4c-4710-9f01-15776103837f","related_version":"2021-01-25T20:31:31.319Z","sha256":"hHuhBwKscfqvLC3y2FfZtHi3DNkzE0o8kE8eE6x50pM","signee":"ACME Cyber Company","spec_version":"1.0","type":"signature","valid_from":"2021-01-25T20:31:31.319516Z","valid_until":"2022-01-01T12:12:12.123456Z"}
```

---

Step 1.5: Create base64URL.encoded version of the JCS signature from step 1.4

---

```
eyJhbGdvcml0aG0iOiJSUzI1NiIsImNyZWF0ZWQiOiIyMDIxLTAxLTI1VDIwOjMxOjMxLjMxOTUxNloiLCJjcmVhdGVkX2
J5IjoiaWRlbnRpdHktLXV1aWQyIiwiaWQiOiJzaWduYXR1cmUtLWFmODkyMjkyLWM0YjQtNDdlYi05YmU2LTQ4OTdmZjRi
OTM4OCIsIm1vZGlmaWVkIjoiMjAyMS0wMS0yNVQyMDozMTozMS4zMTk1MTZaIiwicHVibGljX2tleXMiOlsiTUlJQklqQU
5CZ2txaGtpRzl3MEJBUUVGQUFPQ0FROEFNSUlCQ2dLQ0FRRUFwdEtaeUZQU3R2bU9sYjBXaWhPQmhsSFVyNndGREhDK3RX
N2hKQXVkZlRRNW1IWlFwQjhQb016MDd1ZFpBK2RHOGRoVUlQa21YbHAxVGdSRWVZVEhkaHhodWYweS9HaGJwWnY1SlBZSH
gzd2F0TytIV08ycVlralJNRWNyV2hQTWRhVmtTL1hlL2xpYU1jb3c0allvV2FGbThWb2JlWXN5VkQyYldXZHlsNGpvVEVF
VG0xWjQ3Um5uZlIxNWtWaFZ1ZFZyRHpFRm1NNG5YVi82ZG1JZzE4NFJKRTRodHR3QkZ4UjhxWkNRQ3dUaUptc295SnhmVV
IwR3M0ZVBLYzVzQjBOVGttRlpjNWtsUVNpdGQ2N1JKbjJsZGhicUU3RXBEbDRYbEl0K1V5TEptMWd1Q0JsdGlhOEFna2U3
ZFh1aHBCN2hRNkxKd1k0RWp6dGhrSjhJUHdJREFRQUIiXSwicmVsYXRlZF90byI6InBsYXlib29rLS1hMDc3NzU3NS01Yz
RjLTQ3MTAtOWYwMS0xNTc3NjEwMzgzN2YiLCJyZWxhdGVkX3ZlcnNpb24iOiIyMDIxLTAxLTI1VDIwOjMxOjMxLjMxOVoi
LCJzaGEyNTYiOiJoSHVoQndLc2NmcXZMQzN5MkZmWnRIaTNETmt6RTBvOGtFOGVFNng1MHBNIiwic2lnbmVlIjoiQUNNRS
BDeWJlciBDb21wYW55Iiwic3BlY192ZXJzaW9uIjoiMS4wIiwidHlwZSI6InNpZ25hdHVyZSIsInZhbGlkX2Zyb20iOiIy
```

MDIxLTAxLTI1VDIwOjMxOjMxLjMxOTUxNloiLCJ2YWxpZF91bnRpbCI6IjIwMjItMDEtMDFUMTI6MTI6MTIuMTIzNDU2WiJ9

---

Step 2.0: Verify the signature received in step 1.3 of the B64JCS data from step 1.5 using the public key form 1.2 and using the algorithm from the signature object RS256

---

Signature is valid

---

Step 3.0: Compute the hash of the playbook and make sure it matches the hash in the signed signature

---

---

Step 3.1: Remove existing signatures before computing hash

---


```
{
  "type": "playbook",
  "spec_version": "1.0",
  "id": "playbook--a0777575-5c4c-4710-9f01-15776103837f",
  "name": "Playbook 1",
  "created": "2021-01-25T20:31:31.319Z",
  "modified": "2021-01-25T20:31:31.319Z"
}
```


---

Step 3.2: Create JCS canonical version of the playbook from step 3.1

---


```
{"created":"2021-01-25T20:31:31.319Z","id":"playbook--a0777575-5c4c-4710-9f01-15776103837f","modified":"2021-01-25T20:31:31.319Z","name":"Playbook 1","spec_version":"1.0","type":"playbook"}
```


---

Step 3.3: Create SHA256 (in hex) of canonical version of playbook from step 3.2

---

847ba10702ac71faaf2c2df2d857d9b478b70cd933134a3c904f1e13ac79d293

---

Step 3.4: Create base64URL.encoded version of the SHA256 hash from step 1.3 and remove any padding

---

hHuhBwKscfqvLC3y2FfZtHi3DNkzE0o8kE8eE6x50pM

---

Step 4.0: Compare computed hash with hash found in signed signature

---

Computed Hash: hHuhBwKscfqvLC3y2FfZtHi3DNkzE0o8kE8eE6x50pM
Signed Hash: hHuhBwKscfqvLC3y2FfZtHi3DNkzE0o8kE8eE6x50pM
Hashes match and content is valid

# Appendix B. Security and Privacy Considerations

The following two sections are copied verbatim into the IANA Considerations Appendix.

## B.1 Security Considerations

Security considerations relating to the generation and consumption of CACAO messages are similar to application/json and are discussed in section 12 of [RFC8259].

Unicode is used to represent text such as descriptions in the format. The considerations documented by Unicode Technical Report #36: Unicode Security Considerations [UnicodeTR#36] should be taken into account.

The CACAO standard does not itself specify a transport mechanism for CACAO documents. As there is no transport mechanism specified, it is up to the users of this specification to use an appropriately secured transport method, for example TLS.

Documents of "application/cacao+json" are CACAO based Cybersecurity Playbook documents. The documents may contain active or executable content as well as URLs, IP addresses, and domain names that are known or suspected to be malicious. Systems should thus take appropriate precautions before decoding any of this content, either for persistent storage or execution purposes. Such precautions may include measures such as de-fanging, sandboxing, or other measures. The samples included in CACAO documents are reference samples only, and there is no provision or expectation in the specification that they will be loaded and/or executed. There are provisions in the specification to encrypt these samples so that even if a tool decodes the data, a further active step must be done before the payload will be "live". It is highly recommended that all active code be armored in this manner.

CACAO specifies the use of hashing and encryption mechanisms for some data types. A cryptography expert should be consulted when choosing which hashing or encryption algorithms to use to ensure that they do not have any security issues.

CACAO specifies the use of digital signature technology that is based on concepts from JWS [RFC7515], JWK [RFC7517], and relies on JCS [RFC8785]. In addition to the security considerations defined in section 10 of JWS, section 9 of JWK, and section 5 of JCS, implementers should carefully consider and verify any digital certificate that is delivered via the CACAO Playbook itself to ensure that it is coming from the identity that it claims to come from.

CACAO provides a graph-based data model. As such, CACAO implementations should implement protections against graph queries that can potentially consume a significant amount of resources and prevent the implementation from functioning in a normal way.

## B.2 Privacy Considerations

These considerations are, in part, derived from section 10 of the Resource-Oriented Lightweight Information Exchange [RFC8322].

Documents may include highly confidential, personally identifiable (PII), and classified information. There are methods in the standard for marking elements of the document such that the consumer knows of these limitations. These markings may not always be used. For example, an out-of-band agreement may cover and restrict sharing. Just because a document is not marked as containing information that should not be shared does not mean that a document is free for sharing. It may be the case that a legal agreement has been entered into between the parties sharing documents, and that each party understands and follows their obligations under that agreement as well as any applicable laws or regulations.

Further, a client may succeed in assembling a data set that would not have been permitted within the context of the authorization policies of either provider when considered individually. Thus, providers may face a risk of an attacker obtaining an access that constitutes an undetected separation of duties (SOD) violation. It is important to note that this risk is not unique to this specification, and a similar potential for abuse exists with any other cybersecurity information-sharing protocol.

# Appendix C. IANA Considerations

This appendix contains the required information to register the CACAO media type with IANA. While some of the information here is only for IANA, implementers of CACAO should pay close attention to the security considerations and privacy considerations outlined in this appendix.

This document defines the "application/cacao+json" media type

Media type name:  application

Media subtype name:  cacao+json

Required parameters:  None

Optional parameters:  version
This parameter is used to designate the specification version of CACAO that is being used during HTTP content negotiation. Example: "application/cacao+json;version=1.0". The parameter value is of the form 'n.m', where n is the major version and m the minor version, both unsigned integer values.

Encoding considerations:  binary
Encoding considerations are identical to those specified for the "application/json" media type. See [RFC8259].

Security considerations:
Security considerations relating to the generation and consumption of CACAO messages are similar to application/json and are discussed in section 12 of [RFC8259].

Unicode is used to represent text such as descriptions in the format. The considerations documented by Unicode Technical Report #36: Unicode Security Considerations [UnicodeTR#36] should be taken into account.

The CACAO standard does not itself specify a transport mechanism for CACAO documents. As there is no transport mechanism specified, it is up to the users of this specification to use an appropriately secured transport method, for example TLS.

Documents of "application/cacao+json" are CACAO based Cybersecurity Playbook documents. The documents may contain active or executable content as well as URLs, IP addresses, and domain names that are known or suspected to be malicious. Systems should thus take appropriate precautions before decoding any of this content, either for persistent storage or execution purposes. Such precautions may include measures such as de-fanging, sandboxing, or other measures. The samples included in CACAO documents are reference samples only, and there is no provision or expectation in the specification that they will be loaded and/or executed. There are provisions in the specification to encrypt these samples so that even if a tool decodes the data, a further active step must be done before the payload will be "live". It is highly recommended that all active code be armored in this manner.

CACAO specifies the use of hashing and encryption mechanisms for some data types. A cryptography expert should be consulted when choosing which hashing or encryption algorithms to use to ensure that they do not have any security issues.

CACAO specifies the use of digital signature technology that is based on concepts from JWS [RFC7515], JWK [RFC7517], and relies on JCS [RFC8785]. In addition to the security considerations defined in section 10 of JWS, section 9 of JWK, and section 5 of JCS, implementers should carefully consider and verify any digital certificate that is delivered via the CACAO Playbook itself to ensure that it is coming from the identity that it claims to come from.

CACAO provides a graph-based data model. As such, CACAO implementations should implement protections against graph queries that can potentially consume a significant amount of resources and prevent the implementation from functioning in a normal way.

Privacy considerations:
These considerations are, in part, derived from section 10 of the Resource-Oriented Lightweight Information Exchange [RFC8322].

Documents may include highly confidential, personally identifiable (PII), and classified information. There are methods in the standard for marking elements of the document such that the consumer knows of these limitations. These markings may not always be used. For example, an out-of-band agreement may cover and restrict sharing. Just because a document is not marked as containing information that should not be shared does not mean that a document is free for sharing. It may be the case that a legal agreement has been entered into between the parties sharing documents, and that each party understands and follows their obligations under that agreement as well as any applicable laws or regulations.

Further, a client may succeed in assembling a data set that would not have been permitted within the context of the authorization policies of either provider when considered individually. Thus, providers may face a risk of an attacker obtaining an access that constitutes an undetected separation of duties (SOD) violation. It is important to note that this risk is not unique to this specification, and a similar potential for abuse exists with any other cybersecurity information-sharing protocol.

Interoperability considerations:
The CACAO specification specifies the format of conforming messages and the interpretation thereof. In addition, the OASIS Collaborative Automated Course of Action Operations (CACAO) Technical Committee has defined interoperability tests to ensure conforming products and solutions can exchange CACAO documents.

Published specification:
CACAO Version 1.0 OASIS Committee Specification 01

https://docs.oasis-open.org/cacao/security-playbooks/v1.0/cs01/security-playbooks-v1.0-cs02.html

Cited in the "OASIS Standards" document:

https://www.oasis-open.org/standards#oasiscommiteespecs, from

https://www.oasis-open.org/standards#security-playbooks1.0

Applications which use this media:

Collaborative Automated Course of Action Operations (CACAO) defines a language and serialization format used to exchange cybersecurity playbooks. CACAO enables organizations to share playbooks with one another in a consistent and machine-readable manner, allowing security communities to better understand how to respond to computer-based attacks and to anticipate and/or respond to those attacks faster and more effectively. CACAO is designed to improve many different capabilities, such as collaborative threat analysis, automated threat exchange, automated detection and response, and more.

Fragment identifier considerations: None

Restrictions on usage: None

Additional information:
1. Deprecated alias names for this type: None

2. Magic number(s): n/a [RFC8259]

3. File extension(s): cacao

4. Macintosh file type code: TEXT [RFC8259]

5. Object Identifiers: None

Person and email to contact for further information: Chet Ensign (chet.ensign@oasis-open.org)

Intended usage: COMMON

Author:
OASIS Collaborative Automated Course of Action Operations (CACAO) Technical Committee;

URI reference: https://www.oasis-open.org/committees/cacao/.

Change controller: OASIS

Provisional registration: No

# Appendix D. References

This appendix contains the normative and informative references that are used in this document. Normative references are specific (identified by date of publication and/or edition number or version number) and Informative references are either specific or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the reference document (including any amendments) applies. While any hyperlinks included in this appendix were valid at the time of publication, OASIS cannot guarantee their long term validity.

## D.1 Normative References

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


**[IEP]**
*FIRST Information Exchange Policy 2.0*, FIRST IEP 2.0, 2019. [Online]. Available: https://www.first.org/iep/FIRST_IEP_Framework_v2.0.pdf

**[ISO3166-1]**
*ISO 3166-1:2013 Codes for the Representation of Names of Countries and their Subdivisions — Part 1: Country Codes,* ISO 3166-1:2013, 2013. [Online]. Available: https://www.iso.org/standard/63545.html

**[ISO3166-2]**
*ISO 3166-2:2020 Codes for the Representation of Names of Countries and their Subdivisions — Part 2: Country Subdivision Code*, ISO 3166-2:2020, 2020. [Online]. Available: https://www.iso.org/standard/72483.html

**[ISO10646]**
*ISO/IEC 10646:2014 Information Technology -- Universal Coded Character Set (UCS)*, ISO/IEC 10646:2014, 2014. [Online]. Available: https://standards.iso.org/ittf/PubliclyAvailableStandards/c063182_ISO_IEC_10646_2014.zip

**[RFC2119]**
*Key Words for Use in RFCs to Indicate Requirement Levels*, BCP 14, RFC 2119, March 1997. [Online]. Available: https://www.rfc-editor.org/info/rfc2119

**[RFC3339]**
*Date and Time on the Internet: Timestamps*, RFC 3339, July 2002. [Online]. Available: https://www.rfc-editor.org/info/rfc3339

**[RFC3986]**
*Uniform Resource Identifier (URI): Generic Syntax*, STD 66, RFC 3986, January 2005. [Online]. Available: https://www.rfc-editor.org/info/rfc3986

**[RFC4122]**

*A Universally Unique IDentifier (UUID) URN Namespace*, RFC 4122, July 2005. [Online]. Available: https://www.rfc-editor.org/info/rfc4122

**[RFC5849]**
*The OAuth 1.0 Protocol*, RFC 5849, April 2010. [Online]. Available: https://www.rfc-editor.org/info/rfc5849

**[RFC6750]**
*The OAuth 2.0 Authorization Framework: Bearer Token Usage*, RFC 6750, October 2012. [Online]. Available: https://www.rfc-editor.org/info/rfc6750

**[RFC7493]**
*The I-JSON Message Format*, RFC 7493, March 2015. [Online]. Available: https://www.rfc-editor.org/info/rfc7493

**[RFC7515]**
*JSON Web Signature (JWS)*, RFC 7515, May 2015. [Online]. Available: https://www.rfc-editor.org/info/rfc7515

**[RFC7517]**
*JSON Web Key (JWK)*, RFC 7517, May 2015. [Online]. Available: https://www.rfc-editor.org/info/rfc7517

**[RFC7518]**
*JSON Web Algorithms (JWA)*, RFC 7518, May 2015. [Online]. Available: https://www.rfc-editor.org/info/rfc7518

**[RFC7617]**
*The 'Basic' HTTP Authentication Scheme*, RFC 7617, September 2015. [Online]. Available: https://www.rfc-editor.org/info/rfc7617

**[RFC8037]**
*CFRG Elliptic Curve Diffie-Hellman (ECDH) and Signatures in JSON Object Signing and Encryption (JOSE)*, RFC 8037, January 2017. [Online]. Available: https://www.rfc-editor.org/info/rfc8037

**[RFC8174]**
*Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words*, BCP 14, RFC 8174, May 2017. [Online]. Available: https://www.rfc-editor.org/info/rfc8174

**[RFC8259]**
*The JavaScript Object Notation (JSON) Data Interchange Format*, RFC 8259, December 2017. [Online]. Available: https://www.rfc-editor.org/info/rfc8259.txt

**[RFC8785]**
*JSON Canonicalization Scheme (JCS)*, RFC 8785, June 2020. [Online]. Available: https://www.rfc-editor.org/info/rfc8785

**[STIX-v2.1]**
STIX™ Version 2.1. Edited by Bret Jordan, Rich Piazza, and Trey Darley. 25 January 2021. OASIS Committee Specification 02. https://docs.oasis-open.org/cti/stix/v2.1/cs02/stix-v2.1-cs02.html. Latest stage: https://docs.oasis-open.org/cti/stix/v2.1/stix-v2.1.html.

**[UNSD M49]**
*Standard Country or Area Codes for Statistical Use (M49)*, M49 Standard. [Online]. Available: https://unstats.un.org/unsd/methodology/m49/

## D.2 Informative References

The following referenced documents are not required for the application of this document but may assist the reader with regard to a particular subject area.

**[CalderaAbility]**
Caldera™. "What is an Ability?." Accessed: April 2021. [Online]. Available: https://caldera.readthedocs.io/en/latest/Learning-the-terminology.html#what-is-an-ability

**[CalderaAgent]**
Caldera™. "What is an Agent?." Accessed: April 2021. [Online]. Available: https://caldera.readthedocs.io/en/latest/Learning-the-terminology.html#what-is-an-agent

**[CalderaGroup]**
Caldera™. "What is a Group?." Accessed: April 2021. [Online]. Available: https://caldera.readthedocs.io/en/latest/Learning-the-terminology.html#what-is-a-group

**[PortNumbers]**
IANA. "Service Name and Transport Protocol Port Number Registry." Accessed: April 2021. [Online]. Available: https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml

**[RFC8322]**
*Resource-Oriented Lightweight Information Exchange (ROLIE)*, RFC 8322, February 2018. [Online]. Available: https://www.rfc-editor.org/info/rfc8322

**[SemVer]**
T. Preston-Werner. "Semantic Versioning." Accessed: April 2021. [Online]. Available: https://semver.org/

**[VocabAuto]**
J. B. Rae. "Automotive Industry." Britannica.com. Accessed: December 2020. [Online]. Available: https://www.britannica.com/technology/automotive-industry

**[VocabChem]**
European Commission. "Sectors: Chemicals." Accessed: December 2020.[Online]. Available: https://ec.europa.eu/growth/sectors/chemicals_en

**[VocabDams]**

United States Department of Homeland Security. "National Infrastructure Protection Plan: Dams Sector." Accessed: December 2020. [Online]. Available: https://www.dhs.gov/xlibrary/assets/nppd/nppd-dams-sector-snapshot-508.pdf

**[VocabEmSrv]**
Cybersecurity and Infrastructure Security Agency (CISA). "Critical Infrastructure Sectors: Emergency Services Sector." Accessed: December 2020. [Online]. Available: https://www.cisa.gov/emergency-services-sector

**[VocabGov]**
Cybersecurity and Infrastructure Security Agency (CISA). "Critical Infrastructure Sectors: Government Facilities Sector." Accessed: December 2020. [Online]. Available: https://www.cisa.gov/government-facilities-sector

**[VocabHealth]**
European Commission. "DIRECTIVE (EU) 2016/1148 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 6 July 2016 concerning measures for a high common level of security of network and information systems across the Union." in *Official Journal of the European Union*, 2016. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016L1148&from=EN#d1e836-1-1

**[VocabHumanRights]**
*North American Industry Classification System: 813311 Human Rights Organizations, NAICS, 2017.* [Online]. Available: https://www.census.gov/naics/?input=813311&year=2017&details=813311

**[VocabMetals]**
European Commission. "Sectors: Raw Materials: Industries: Metals." Accessed: December 2020. [Online]. Available: https://ec.europa.eu/growth/sectors/raw-materials/industries/metals_en

**[VocabPServ]**
Wikipedia. "Public Service." Wikipedia.org. Accessed: April 2021. [Online]. Available: https://en.wikipedia.org/wiki/Public_service

**[VocabUtils]**
C. Murphy. "Utilities Sector." Investopedia. Accessed: March 2021. [Online]. Available: https://www.investopedia.com/terms/u/utilities_sector.asp

**[VocabWater]**
Wikipedia. "Water industry." Wikipedia.org. Accessed: December 2020. [Online]. Available: https://en.wikipedia.org/wiki/Water_industry

# Appendix E. Acknowledgments


### Special Thanks:

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

Bret Jordan, Individual
Allan Thomson, Individual
Stephanie Hazlewood, IBM
Emily Ratliff, IBM
Andrew Storms, New Context Services, Inc.
Lior Kolnik, Palo Alto Networks
Marco Caselli, Siemens AG
Vasileios Mavroeidis, University of Oslo

### Participants:

The following individuals were members of this Technical Committee during the creation of this specification and their contributions are gratefully acknowledged:

Curtis Kostrosky, Accenture
Anup Ghosh, Accenture
Patrick Maroney, AT&T
Dean Thompson, Australia and New Zealand Banking Group (ANZ Bank)
Arnaud Taddei, Broadcom
Omar Santos, Cisco Systems
Naasief Edross, Cisco Systems
Jyoti Verma, Cisco Systems
Arsalan Iqbal, CTM360
Avkash Kathiriya, Cyware Labs
Ryan Joyce, DarkLight, Inc.
Paul Patrick, DarkLight, Inc.
Ryan Hohimer, DarkLight, Inc.
Michael Rosa, DHS Office of Cybersecurity and Communications (CS&C)
Aukjan van Belkum, EclecticIQ
Gerald Stueve, Fornetix
Stephanie Hazlewood, IBM
Mahbod Tavallaee, IBM
Srinivas Tummalapenta, IBM
Emily Ratliff, IBM
Jason Keirstead, IBM
John Morris, IBM
Joerg Eschweiler, Individual
Bret Jordan, Individual
Terry MacDonald, Individual
Anil Saldanha, Individual
Frans Schippers, Individual
Allan Thomson, Individual

Rodger Frank, Johns Hopkins University Applied Physics Laboratory
Karin Marr, Johns Hopkins University Applied Physics Laboratory
Chris Dahlheimer, LookingGlass
Jason Webb, LookingGlass
David Kemp, National Security Agency
Christian Hunt, New Context Services, Inc.
Andrew Storms, New Context Services, Inc.
Kaleb Wade, New Context Services, Inc.
Stephen Banghart, NIST
David Darnell, North American Energy Standards Board
Lior Kolnik, Palo Alto Networks
David Bizeul, SEKOIA
Duncan Sparrell, sFractal Consulting LLC
Marco Caselli, Siemens AG
Greg Reaume, TELUS
Ryan Trost, ThreatQuotient, Inc.
Franck Quinard, TIBCO Software Inc.
Toby Considine, University of North Carolina at Chapel Hill
Vasileios Mavroeidis, University of Oslo
Mateusz Zych, University of Oslo

# Appendix F. Revision History

| Revision | Date | Editor(s) | Changes Made |
|---|---|---|---|
| 01 | 2020-03-27 | Bret Jordan, Allan Thomson | Initial Version |
| 02 | 2020-04-21 | Bret Jordan, Allan Thomson | Added terminology, actions, targets, and data markings. A lot of editorial cleanup. |
| 03 | 2020-07-29 | Bret Jordan, Allan Thomson | Added extensions, cleaned up the use of commands and the former actions concept. Enabled embedded targets. Added conformance language. Refactored data markings.<br><br>This version became CSD01. |
| 04 | 2020-11-20 | Bret Jordan, Allan Thomson | Addressed feedback from public review, fixed some editorial and readability issues. Added security infrastructure category vocabulary. Populated the marking TLP and IEP objects. Added related_to property to external references. Added sector vocab. |
| 05 | 2020-12-01 | Bret Jordan, Allan Thomson | Changed 7.7 sector target to have a list of physical locations.<br><br>This version became CSD02 and CS01. |
| 06 | 2021-04-20 | Bret Jordan, Allan Thomson | Moved Data Types to section 9 and fixed all internal section references. Changed playbook features from a vocab to a data type. Added support for digital signatures. Added support for attack types. Cleaned up the industry sectors vocab. Basic editorial cleanup. Cleanup the industry sector vocabulary. Changed variables to have suffix as well as prefix.<br><br>Submitted to be approved as CSD03. |
| 07 | 2021-06-08 | Bret Jordan, Allan Thomson | Fixed some broken reference links. Minor non-material editorial fixes. Cleaned up references by putting them all into IEEE reference format. |

| | | | |
|---|---|---|---|
| | | | Submitted to be approved as CS02 with non-material changes. |

# Appendix G. Notices



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: [http://www.oasis-open.org/policies-guidelines/ipr]

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 AND ITS MEMBERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THIS DOCUMENT OR ANY PART THEREOF.

As stated in the OASIS IPR Policy, the following three paragraphs in brackets apply to OASIS Standards Final Deliverable documents (Committee Specifications, OASIS Standards, or Approved Errata).

[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 Standards Final Deliverable, 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 deliverable.]

[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 OASIS Standards Final Deliverable 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 OASIS Standards Final Deliverable. 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 OASIS Standards Final Deliverable 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 Standards Final Deliverable, 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.]