CACAO Security Playbooks Version 1.0

Committee Specification 01

12 January 2021

This version:
https://docs.oasis-open.org/cacao/security-playbooks/v1.0/cs01/security-playbooks-v1.0-cs01.docx
(Authoritative)
https://docs.oasis-open.org/cacao/security-playbooks/v1.0/cs01/security-playbooks-v1.0-cs01.html
https://docs.oasis-open.org/cacao/security-playbooks/v1.0/cs01/security-playbooks-v1.0-cs01.pdf

Previous version:
https://docs.oasis-open.org/cacao/security-playbooks/v1.0/csd02/security-playbooks-v1.0-csd02.docx
(Authoritative)
https://docs.oasis-open.org/cacao/security-playbooks/v1.0/csd02/security-playbooks-v1.0-csd02.html
https://docs.oasis-open.org/cacao/security-playbooks/v1.0/csd02/security-playbooks-v1.0-csd02.pdf

Latest version:
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 (bret.jordan@broadcom.com), Broadcom
Allan Thomson (atcyber1000@gmail.com), Individual

Editors:
Bret Jordan (bret.jordan@broadcom.com), Broadcom
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. 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:

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. 12 January 2021. OASIS Committee Specification 01. https://docs.oasis-open.org/cacao/security-playbooks/v1.0/cs01/security-playbooks-v1.0-cs01.html. Latest version: https://docs.oasis-open.org/cacao/security-playbooks/v1.0/security-playbooks-v1.0.html.

Notices:

Table of Contents

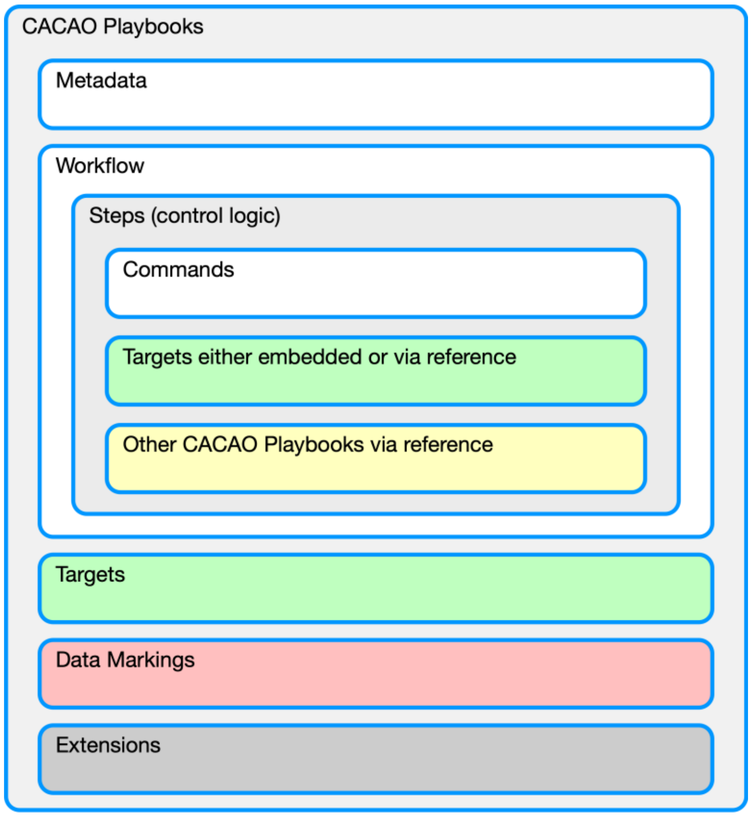
1 Introduction

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 4), workflow steps (section 5), commands (section 6), targets (section 7), extensions (section 8), and data markings (section 9).



1.2 Executable Playbooks

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.

1.3 Playbook Template

A playbook template provides example actions related to a particular security incident, malware, vulnerability or other security response. 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.

1.4 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.

1.5 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.

1.6 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.

1.7 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.

1.8 Changes From Earlier Versions

N/A

1.9 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 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.

2.1 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.

2.2 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. The value for this property **MUST** come from the `region` vocabulary (see section 2.2.1).
`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.
`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.

2.2.1 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`

2.3 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. 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. 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.

2.4 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.

2.5 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 2020",
  "source": "ACME Security Company, Solutions for FuzzyPanda 2020, January 2020. [Online].
    Available: http://www.example.com/info/fuzzypanda2020.html",
  "url": "http://www.example.com/info/fuzzypanda2020.html",
  "external_id": "fuzzypanda 2020.01",
  "reference_id": "malware--2008c526-508f-4ad4-a565-b84a4949b2af"
}
```

2.6 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). 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). 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. If this property is not present, then the precision is unspecified. If this property is present, the **`latitude`** and **`longitude`** properties **MUST** be present.

2.7 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 [JCS] 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

2.8 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.

2.9 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.

2.10 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.

2.11 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 with **$$** for both declaration and use

- **MUST** be no longer than 250 ASCII characters in length, excluding the variable prefix $$
- **MUST** start with a letter or the underscore character after the variable prefix $$
- are case-sensitive (age, Age and AGE are three different variables) but **SHOULD** be lowercase

2.11.1 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).

2.11.2 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.

2.11.3 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. 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.

			When set to `true` the variable declaration defines that the variable's initial value is passed into the playbook from a calling context. 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
    }
```

2.11.4 Variable Type Vocabulary

Vocabulary Name: `variable-type`

Vocabulary Value	Description
`string`	
`uuid`	
`integer`	

`long`	
`mac-addr`	
`ipv4-addr`	
`ipv6-addr`	
`uri`	
`sha256_hash`	
`hexstring`	
`dictionary`	

3 Core Concepts

3.1 Types of Actions

This section defines the types of actions used by CACAO security playbooks.

3.1.1 Investigate

This is an action used to gather information relevant to the construction or modification of cyber security playbooks. This includes gathering of information about a possible incident, problem, attack, or compromise. In some cases, an investigative action could require changes to the systems, networks or application behaviors in order to facilitate a deeper understanding of the investigation and resultant potential response.

3.1.2 Prevent

This is an action used to help ensure that an incident, problem, attack, or compromise does not happen in the first place. In some cases, preventive actions may overlap with other mitigative and remediation actions.

3.1.3 Mitigate

This is an action used to respond to problems that can occur from an incident, problem, attack, or compromise. This is often done when a remediative action is not currently possible. For example, when a system patch is not yet available, one might deploy compensating controls such as moving the system into a sandbox virtual LAN (VLAN) or deploying more stringent firewall rules.

3.1.4 Remediate

This is an action often used for the purpose of eradicating an issue, problem, attack, or compromise on one or more systems that have been determined to be compromised or involved in the particular event.

3.2 Playbook Terminology

This section defines some of the terminology that is used by CACAO security playbooks.

3.2.1 Playbook

This defines a workflow for security orchestration where that workflow contains a set of workflow steps representing a set of commands to take in a logical process. A playbook is a collection of one or more steps that defines a behavior and provides guidance on how to address a certain security event, incident, problem, attack, or compromise. A playbook may be triggered by an automated or manual event or observation. 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 function playbooks similar to how software application development leverages modular libraries of common functions shared across different applications.

3.2.2 Detection Playbook

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

3.2.3 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 done. These playbooks will likely inform other subsequent actions upon completion of the investigation.

3.2.4 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. These 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.

3.2.5 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. These 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.

3.2.6 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. These playbooks can fix affected assets by selectively correcting problems due to malicious activity by reverting the system or network to a known good state.

3.3 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 (TODO REF) 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.

3.4 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 4.1).

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.

3.4.1 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).

3.4.2 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.

3.5 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.

4 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.

4.1 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 and section 1.3, 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. 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`**. 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. If the **`valid_until`** property is omitted, then there is no constraint on the latest time for which the playbook is valid. This property **MUST** be greater than the timestamp in the **`valid_from`** property if the **`valid_from`** property is defined.
`derived-from`		`identifier`	The ID of a playbook that this playbook was derived from. 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. 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. If specified, the value of this property **MUST** be between 0 and 100. 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). Marketplaces and sharing organizations **MAY** define additional rules for how this property should be assigned. If specified, the value of this property **MUST** be between 0 and 100. 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. 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`		`dictionary`	An optional property that contains a list of features that are enabled for this playbook. The keys for this dictionary **MUST** come from the `playbook-features` vocabulary. The values for each key **MUST** be a boolean of either `true` or `false`. If a key is not present in the dictionary, then the value is unknown.
`markings`		`list` of type	An optional list of data marking objects that apply to

		identifier	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). 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 2.11 for information about referencing variables. 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 2.11).
workflow_start		identifier	The first workflow step included in the **workflow** property that **MUST** be executed when starting the workflow. The ID **MUST** represent a CACAO workflow step object.
workflow_exception		identifier	The workflow step invoked whenever a playbook exception condition occurs. The ID **MUST** represent a CACAO workflow step object.
workflow		dictionary	The **workflow** property contains the processing logic for the playbook as workflow steps. 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 2.7 for more information on identifiers). The value for each key **MUST** be a CACAO workflow step object (see section 5).
targets		dictionary	A dictionary of targets that can be referenced from workflow steps found in the **workflow** property. 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 2.7 for more information on identifiers). The value for each key **MUST** be a CACAO target object (see section 7).

extension_definitions		dictionary	A dictionary of extension definitions that are referenced from workflow steps found in the **workflow** property. 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 2.7 for more information on identifiers). The value for each key **MUST** be a CACAO extension object (see section 8).
data_marking_definitions		dictionary	A dictionary of data marking definitions that can be referenced from the playbook found in the **markings** property. The key for each entry in the dictionary **MUST** be an identifier that uniquely identifies the data marking. The id **MUST** use the object type of "data-marking" (see section 2.7 for more information on identifiers). The value for each key **MUST** be a CACAO data marking object (see section 9).

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": "2020-07-31T23:59:59.999999",
  "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 2020",
      "source": "ACME Security Company, Solutions for FuzzyPanda 2020, January 2020. [Online].
        Available: http://www.example.com/info/fuzzypanda2020.html",
      "url": "http://www.example.com/info/fuzzypanda2020.html",
      "hash": "f92d8b0291653d8790907fe55c024e155e460eabb165038ace33bb7f2c1b9019",
      "external_id": "fuzzypanda 2020.01"
    }
  ],
"features": {
  "if-logic": true,
  "data-markings": true
},
  "markings": [
    "data-marking--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": { }
}
```

4.2 Playbook Type Vocabulary

A playbook may be categorized as having multiple types defined from this vocabulary. These definitions are taken from section 3.2.

Vocabulary Name: `playbook-type`

Vocabulary Value	Description
`detection`	See section 3.2.2 for an explanation.
`investigation`	See section 3.2.3 for an explanation.
`prevention`	See section 3.2.4 for an explanation.
`mitigation`	See section 3.2.5 for an explanation.
`remediation`	See section 3.2.6 for an explanation.

4.3 Playbook Features Vocabulary

The major features and functionality of a playbook.

Vocabulary Name: `playbook-features`

Vocabulary Value	Description
`parallel-processing`	See section 5.7.
`if-logic`	See section 5.8.
`while-logic`	See section 5.9.
`switch-logic`	See section 5.10.
`temporal-logic`	See section 5.1 **`delay`** and **`timeout`** properties.
`data-markings`	See section 3.4 and section 9.
`extensions`	See section 8.

4.4 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.	No	`string`	"return_caller"

	NOTE: this is similar to rolling back the stack in a computer program.			
`$$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`	

5 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.

5.1 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. 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. The integer **MUST** be a positive value greater than 0. 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. 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 2.11.2 for information about referencing variables. 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 2.11.3).
`owner`		`identifier`	An ID that represents the entity that is assigned as the owner or responsible party for this step. 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. The ID **MUST** represent either a CACAO workflow step object or a CACAO playbook object. 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. 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.

			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. 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 2.7 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.

5.2 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 5.3.
`end`	This workflow step is the end of a playbook or branch of workflow steps. See section 5.4.
`single`	This workflow step contains the actual commands to be executed. See section 5.5.
`playbook`	This workflow step executes a named playbook from within the current playbook. See section 5.6.
`parallel`	This workflow step contains a list of one or more steps that execute in parallel. See section 5.7.
`if-condition`	This workflow step contains an if-then-else statement. See section 5.8.
`while-condition`	This workflow step contains a while loop. See section 5.9.
`switch-condition`	This workflow step contains a switch statement. See section 5.10.

5.3 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>"
},
```

5.4 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",
},
```

5.5 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. The value of this property **MUST** contain a CACAO target object (see section 7). 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. 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 `variables`	The optional list of arguments passed to the target(s) as input to the step
`out_args`		`list` of type `variables`	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"
}
```

5.6 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. 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.

			The value of this property **MUST** contain a CACAO target object (see section 7). 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 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 `variables`	The optional list of arguments passed to the target(s) as input to the referenced playbook.
`out_args`		`list` of type `variables`	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 ]
}
```

5.7 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. 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. 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"
},
```

5.8 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. 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. 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" ]
}
```

5.9 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. 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.

			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"
}
```

5.10 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. 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 2.7 for more information on identifiers). 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" ]
  }
}
```

6 Commands

The CACAO command object contains detailed information about the commands that are to be executed or processed automatically or manually as part of a workflow step (see section 5). 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.

6.1 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. 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. 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": "https://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",
}
```

6.2 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.

7 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"
}
```

7.1 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. 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 2.7 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.

7.2 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.
`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

7.3 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.

7.4 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.

7.5 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.

7.6 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.

7.7 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 7.7.1.

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.

7.7.1 Industry Sector Vocabulary

Vocabulary Name: `industry-sector-ov`

Industry sector is an open vocabulary that describes industrial and commercial sectors. It is intended to be holistic; it has been derived from several other lists and is not limited to "critical infrastructure" sectors.

Sector Type	Description
`agriculture`	
`aerospace`	

automotive	
chemical	
commercial	
communications	
construction	
defense	
education	
energy	
entertainment	
financial-services	
government	
emergency-services	
government-local	
government-national	
government-public-services	e.g., sanitation
government-regional	
healthcare	
hospitality-leisure	
infrastructure	
dams	
nuclear	
water	
insurance	
manufacturing	
mining	
non-profit	

pharmaceuticals	
retail	
technology	
telecommunications	
transportation	
utilities	

Example

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

7.8 HTTP API Target

This target type contains an HTTP API target. In addition to the inherited properties, this section defines one additional specific property that is 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].

7.9 SSH CLI Target

This target type contains an SSH CLI target. 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 `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.

7.10 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. The values for this property **MUST** come from the `security-infrastructure-type-ov` vocabulary.

7.10.1 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"]
  }
}
```

7.11 General Network Address Target

This target type contains a Network Address Target. In addition to the inherited properties, this section defines four 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 7.10.1. 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": "https://someorg.com/tellmetoorchestratewhat/amethod"
    },
    "username": "someusername",
    "password": "apassword",
    "category": "firewall",
    "location": {
    }
  }
}
```

8 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.

8.1 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. While the extension's description is optional, it is encouraged that producers **SHOULD** populate the field. 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. 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",
  "name": "Extension Foo",
  "description": "This schema adds foo to bar for steps",
  "created_by": "identity--uuid1",
  "schema": "https://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": "https://www.example.com/schema-foo/v1/",
      "version": "1.2.1"
    }
  }
}
```

Target Extension Example
```
{
  "type": "playbook",
  ...
  "target": {
    "type": "net-address",
    "address": {
      "url": "https://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",
      "name": "Network Target with Mac",
      "description": "This schema adds L2 mac address to network targets",
      "created_by": "identity--uuid1",
      "schema": "https://www.example.com/schema-foo/v1/",
      "version": "1.2.1"
    }
  }
}
```

9 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.

9.1 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. 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.
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.
valid_from		timestamp	The time from which this data marking is considered valid. 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. If the **valid_until** property is omitted, then there is no constraint on the latest time for which the data marking is valid. This property **MUST** be greater than the timestamp in the **valid_from** property if the **valid_from** property is defined.
marking_extensions		dictionary	This property defines the extensions that are in use on this data marking. 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 2.7 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.

9.2 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 9.3.
`marking-tlp`	The TLP marking definition. See section 9.4.
`marking-iep`	The IEP marking definition. See section 9.5.

9.3 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"
}
```

9.4 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:
			
			`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",
}
```

9.5 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"
}
```

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 3.4.

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 4.

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 4.1 and 5.

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 4.1 and 6. 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 4.1 and 7.

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 3.4, 4.1, and 9.

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 4.1 and 8.

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.

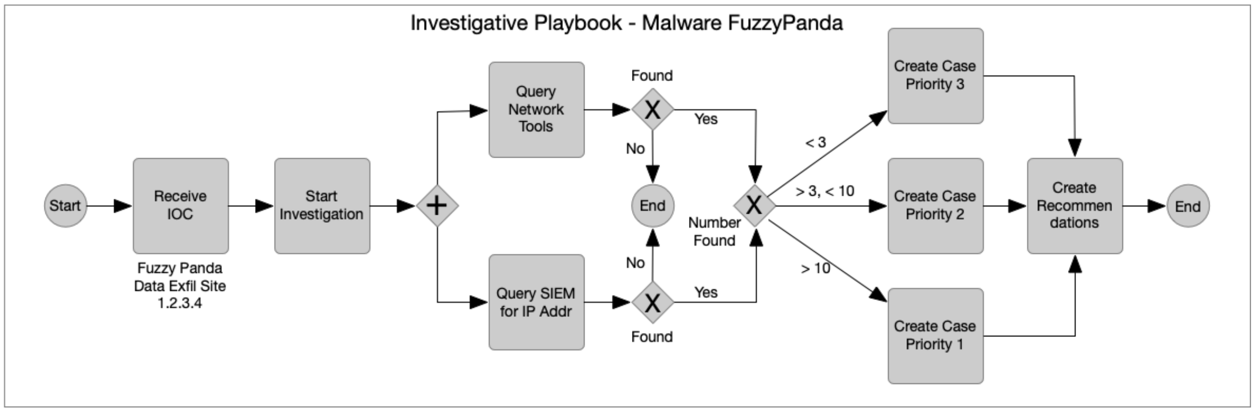
Appendix A. Examples

The examples in this section are based on real and hypothetical 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 Example: Investigative Playbook

This is an example playbook for investigating the presence of a fictitious piece of malware called FuzzyPanda.

A.1.1 Diagram



A.1.2 Playbook in JSON

```
{
  "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": "2020-07-31T23:59:59.999999",
  "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 2020",
      "source": "ACME Security Company, Solutions for FuzzyPanda 2020, January 2020. [Online].
Available: http://www.example.com/info/fuzzypanda2020.html",
      "url": "http://www.example.com/info/fuzzypanda2020.html",
      "hash": "f92d8b0291653d8790907fe55c024e155e460eabb165038ace33bb7f2c1b9019",
      "external_id": "fuzzypanda 2020.01"
    }
  ],
  "markings": [
    "data-marking--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"
}
```

A.1.2.1 Workflow

A.1.2.2 Actions

A.2 Example: Mitigation Playbook

A new domain that is less than 7 days old, that has never been seen before communicating with the internal system has been detected.

A.2.1 Playbook in JSON

```
{
  "type": "playbook",
  "spec_version": "1.0",
  "id": "playbook--uuid1",
  "name": "Traffic Flow Redirect",
  "description": "This playbook redirect, log and copy specific traffic flows",
  "playbook_types": ["mitigation"],
  "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": "2020-07-31T23:59:59.999999",
  "derived-from": "playbook--uuid99",
  "priority": 100,
  "severity": 70,
  "impact": 5,
  "labels": [ "Domain", "Mitigation", "Malicious", "Network Traffic"],
  "variables": {
    "$$redirection_site": {
      "type": "ipv4-addr",
      "description": "The IP address for the redirection of traffic",
      "value": "1.2.3.4",
      "constant": false,
      "external": false
    },
    "$$copy_to_ip": {
      "type": "ipv4-addr",
      "description": "The IP address to send a copy of the traffic",
      "value": "1.2.3.4",
      "constant": false,
      "external": false
    }
    "$$domain": {
      "type": "string",
      "description": "The Domain identified on lookup passed into this playbook",
      "constant": true,
      "external": true
    }
  },
  "workflow_start": "workflow-step-uuid01",
```

A.2.1.1 Workflow

```
  "workflow": {
    "workflow-step-uuid01": {
      "type": "start",
      "name": "Start Traffic Flow Redirect Playbook Example",
      "on_completion": "workflow-step-uuid02"
    },
    "workflow-step-uuid02": {
      "type": "if-condition",
      "name": "Redirect Traffic Flow",
      "description": "In this step the traffic flow will be redirected if it matches a
particular domain",
      "condition": "$$domain == www.test.com",
      "on_true": [ "workflow-step-uuid03" ],
      "on_false": [ "workflow-end" ]
    },
    "workflow-step-uuid03": {
      "type": "parallel",
      "name": "Log and Copy",
      "description": "In this step the traffic flow will be logged and redirected",
      "next_steps": [
        "workflow-step-uuid03.1",
        "workflow-step-uuid03.2"
      ]
      "on_completion": "workflow-uuid4",
      "on_failure": "workflow-end"
    },
    "workflow-step-uuid03.1": {
      "type": "single",
      "name": "Log the event",
      "description": "This is a step the traffic flow will  be logged",
      "timeout": "$$method_timeout",
      "step_name": "logTrafficFlows",
      "commands" : [
              {
                     "type": "http-api",
                     "command" : "logTrafficFlows"
              }
       ],
      "in_args" : [ "$$domain", "5mins" ],
      "out_args" : null,
      "target" : {
              "type": "http-api",
              "http_url" : "https://trafficcontroller/logTrafficFlows"
              "http_auth": // username/password
      },
      "on_completion": $$RETURN_CALLER,
      "on_failure": "workflow-end"
    },
    "workflow-step-uuid03.2": {
      "type": "single",
      "name": "Log the event",
      "description": "This is a step the traffic flow will  be copied to a new destination",
      "timeout": "$$method_timeout",
      "step_name": "copyTrafficFlows",
      "commands" : [
              {
                     "type": "http-api",
                     "command" : "copyTrafficFlows"
              }
       ],
      "in_args" : [ "$$domain", "$$copy_to_ip", "5mins" ],
      "out_args" : null,
      "target" : {
              "type": "http-api",
              "http_url" : "https://trafficcontroller/copyTrafficFlows"
              "http_auth": // username/password
      },
      "on_completion": $$RETURN_CALLER,
      "on_failure": "workflow-end"

  },
  "workflow-end": {
      "type": "single",
      "name": "End",
      "description": "This is a step to end the workflow",
      "action_id": "action--end"
  }
```

A.3 Example: Alert Investigation & Analysis

Background: This PLAYBOOK provides threat analysts the necessary documentation to review and vet alerts from various threat intelligence and telemetry systems to deliver an analysis to a security operations team for further action.

The systems used to provide information on the analysis includes INDICATORS OF COMPROMISE (IOC) like MALWARE HOSTING/DISTRIBUTION and VIRUS/BOTNET infections and INDICATORS OF RISK like open PORTS, expired CERTIFICATES, and inferred VULNERABILITIES.

CACAO Features Deployed

- Multiple commands
- Sequential commands
- Human process commands
- Automated commands
- Conditional logic checks

A.3.1 High Level Flow: AlertInvestigationAnalysis-01

There are 3 high-level aspects of this playbook:

1. **Trigger**...*What causes this playbook to occur*
    a. Receive alert in email from security operations team to threat intelligence analysis team asking for further investigation around alert and provide further analysis back to them

2. **Action**...*What the playbook defines to occur (1 or more action steps) when the initial trigger occurs*
    a. Threat Intelligence analysis team investigates the alert and builds analysis report based on the alert
    b. High-Level Action:
        i. Gather the following list of information and add to the report
            1. All active-threats and associated meta-data that are related to the element(s) defined in the alert. An active threat is one that has occurred before the time of alert, or immediately following the alert timestamp.
                a. Include for all active threats assessment on suggested mitigations
            2. All network CIDR/ASN ownership information
            3. All Passive DNS history information
            4. All current Whois information related to the element(s) defined in the alert.

3. **Outcome**...*What is the expected outcome of the playbook upon execution*
    a. Analysis report is provided back to the security operations team

A.3.2 Playbook: AlertInvestigationAnalysis-01

Playbooks have predefined global variables/macros that allow use throughout
$$flow-error-id
$$flow-error-msg
$$flow-id
$$flow-name
$$alert.element
$$alert.source

Playbook uses the following local variables
$$active_threats
$$network_cidrs
$$network_asns
$$passive_dns_history
$$whois
$$report

```
{
  "type": "playbook",
  "spec_version": "1.0",
  "id": "playbook--uuid1",
  "name": "Alert Analysis Playbook Example 1",
  "description": "This playbook provides alert triage and analysis workflow",
  "playbook_types": ["investigation"],
  "created_by": "identity--uuid2",
  "created": "2020-06-04T15:56:00.123456Z",
  "modified": "2020-06-04T15:56:00.123456Z",
  "revoked": false,
  "labels": [ "Network Support", "Network Traffic" ],
  "external_references": [
    {
      "name": "ACME Security Company",
      "description": "ACME alert and analysis security review",
      "url": "http://www.example.com/info/alertanalysis01.html",
      "hash": "f92d8b0291653d8790907fe55c024e155e460eabb165038ace33bb7f2c1b9019",
      "external_id": "ACME AlertInvestigationAnalysis-01"
    }
  ],
  "markings": [
    "data-markings--uuid1"
  ],
  "valid_from": "2020-06-04T15:56:00.123456Z",
  "playbook_variables": {
    "$$active_threats": {
      "type": "NOTE: Need a hash map type",
      "description": "The list of active threats gathered during the investigation",
      "value": [ map of threats with a key == uniqueid for threat; value == the threat object
],
      "constant": false
    },
    "$$network_cidrs": {
      "type": "NOTE: Need a list type",
      "description": "The list of relevant network CIDRs gathered during the investigation",
      "value": [ list of IPs ],
      "constant": false
    }
    "$$network_asns: {
      "type": "NOTE: Need a list type",
      "description": "The list of relevant ASNs gathered during the investigation",
      "value": [ 2, 8, 12 ],
      "constant": false
    },
    "$$passive_dns_history: {
      "type": "NOTE: Need a hash map type",
      "description": "The map of relevant Passive DNS entries gathered during the
investigation",
      "value": map of maps representing passive DNS entry structures,
      "constant": false
    },
    "$$whois: {
      "type": "NOTE: Need a hash map type",
      "description": "The map of relevant who is structure gathered during the investigation",
      "value": map of maps representing whois data structures,
      "constant": false
    },
    "$$report: {
      "type": "NOTE: Need a blob type",
      "description": "The generated report",
      "value": // we should not require value to be defined in all variable cases in the
declaration
      "constant": false
     },
    "$$method_timeout {
        "type: "integer",
       "description": "Timeout used for playbook HTTP methods",
        "value": "60000",
        "constant": true
        }
  },
  "workflow_start": "step--a76dbc32-b739-427b-ae13-4ec703d5797e",
   "workflow" : {
        "step--a76dbc32-b739-427b-ae13-4ec703d5797e": {
          "type": "start",
          "name": "Start Playbook Example 1",
          "on_completion": "step--a76dbc32-b739-427b-ae13-4ec703d5797f"
        },
        "step--a76dbc32-b739-427b-ae13-4ec703d5797f": {
          "type": "single",
          "timeout": "$$method_timeout",
          "step_name": "gatherActiveThreats",
          "description": "Gathers Active Threats Associated with Alert Trigger",
          "commands" : [
                {
                        "type": "http-api",
                        "command" : "getactivethreats"
                }
          ],
          "in_args" : [ "$$alert.element" ]
          "out_args" : [ "$$active_threats" ]
          "target" : {
                "type": "http-api",
                "http_url" : "https://datastore/getactivethreats"
                "http_auth": // username/password
         },
          "on_success": "step--a76dbc32-b739-427b-ae13-4ec703d57977",
          "on_failure": "step--227b649f-cc38-4b75-b926-de631b4c42b1"
        },
        "step--a76dbc32-b739-427b-ae13-4ec703d57977": {
          "type": "single",
          "timeout": "$$method_timeout",
          "step_name": "gatherCIDRInfo",
          "description": "Gathers CIDR Info for Alert Trigger",
          "commands" : [
                {
                        "type": "http-api",
                        "command" : "getCIDRInfo"
                }
          ],
          "in_args" : [ "$$alert.element" ]
          "out_args" : [ "$$network_cidrs", "$$asn_cidrs" ]
          "target" : {
                "type": "http-api",
                "http_url" : "https://datastore/getcidrinfo"
                "http_auth": // username/password
          },
          "on_success": "step--a76dbc32-b739-427b-ae13-4ec703d57988",
          "on_failure": "step--227b649f-cc38-4b75-b926-de631b4c42b1"
        },
        "step--a76dbc32-b739-427b-ae13-4ec703d57988": {
          "type": "single",
          "timeout": "$$method_timeout",
          "step_name": "getPassiveDNSInfo",
          "description": "Gathers Passive DNS Info for Alert Trigger",
          "commands" : [
                {
                        "type": "http-api",
                        "command" : "getPassiveDNSInfo"
                }
          ],
          "in_args" : [ "$$alert.element" ]
          "out_args" : [ "$$passive_dns_info" ]
          "target" : {
                "type": "http-api",
                "http_url" : "https://datastore/getpassivednsinfo"
                "http_auth": // username/password
          },
          "on_success": "step--a76dbc32-b739-427b-ae13-4ec703d57999",
          "on_failure": "step--227b649f-cc38-4b75-b926-de631b4c42b1"
        },
        "step--a76dbc32-b739-427b-ae13-4ec703d57999": {
          "type": "single",
          "timeout": "$$method_timeout",
          "step_name": "getWhoisInfo",
          "description": "Gathers Who Is Info for Alert Trigger",
          "commands" : [
               {
                      "type": "http-api",
                      "command" : "getWhoIsInfo"
               }
          ],
          "in_args" : [ "$$alert.element" ]
          "out_args" : [ "$$who_is_info" ]
          "target" : {
               "type": "http-api",
               "http_url" : "https://datastore/getwhoisInfo"
               "http_auth": // username/password
          },
          "on_success": "step--a76dbc32-b739-427b-ae13-4ec703d57911",
          "on_failure": "step--227b649f-cc38-4b75-b926-de631b4c42b1"
        },
        "step--a76dbc32-b739-427b-ae13-4ec703d57911": {
          "type": "single",
          "timeout": "$$method_timeout",
          "step_name": "buildAlertReport",
          "description": "Builds Alert Report for Alert Trigger",
          "commands" : [
               {
                      "type": "http-api",
                      "command" : "buildAlertReport"
               }
          ],
          "in_args" : [ "$$alert.element", $$active_threats, $$network_cidrs, $$network_asns,
$$passive_dns_history, $$whois ],
          "out_args" : [ "$$report" ],
         "target" : {
               "type": "http-api",
               "http_url" : "https://datastore/buildAlertReport"
               "http_auth": // username/password
          },
          "on_success": "step--a76dbc32-b739-427b-ae13-4ec703d57900"
          "on_failure": "step--227b649f-cc38-4b75-b926-de631b4c42b1"
        },
        "step--a76dbc32-b739-427b-ae13-4ec703d57900": {
          "type": "single",
          "timeout": "$$method_timeout",
          "step_name": "emailReport",
          "description": "Emails Alert Report for Alert Trigger",
          "commands" : [
               {
                      "type": "http-api",
                      "command" : "emailAlertReport"
               }
          ],
          "on_success": "step--227b649f-cc38-4b75-b926-de631b4c42b1"
          "on_failure": "step--227b649f-cc38-4b75-b926-de631b4c42b1"
        },
        "step--227b649f-cc38-4b75-b926-de631b4c42b1": {
          "type": "end",
          "name": "End Playbook"
        }
},
"targets": {},
}
```

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 to use an appropriately secured transport method. For example, TLS, JSON Web Encryption [RFC7516] and/or JSON Web Signature [RFC7515] can provide such mechanisms.

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 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 to use an appropriately secured transport method. For example, TLS, JSON Web Encryption [RFC7516] and/or JSON Web Signature [RFC7515] can provide such mechanisms.

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 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, personal (PII), and/or 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-cs01.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", 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", 2013. [Online]. Available: https://www.iso.org/standard/63545.html.

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

[JCS]
Rundgren, A., Jordan, B., and S. Erdtman, "JSON Canonicalization Scheme (JCS)", RFC 8785, DOI 10.17487/RFC8785, June 2020, https://www.rfc-editor.org/info/rfc8785 .

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

[RFC3339]
Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, http://www.rfc-editor.org/info/rfc3339.

[RFC3986]
Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, http://www.rfc-editor.org/info/rfc3986.

[RFC4122]
Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, http://www.rfc-editor.org/info/rfc4122.

[RFC5849]
Hammer-Lahav, E., Ed., "The OAuth 1.0 Protocol", RFC 5849, DOI 10.17487/RFC5849, April 2010, https://www.rfc-editor.org/info/rfc5849.

[RFC6750]
Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", RFC 6750, DOI 10.17487/RFC6750, October 2012, https://www.rfc-editor.org/info/rfc6750.

[RFC7493]
Bray, T., Ed., "The I-JSON Message Format", RFC 7493, DOI 10.17487/RFC7493, March 2015, https://www.rfc-editor.org/info/rfc7493.

[RFC7617]
Reschke, J., "The 'Basic' HTTP Authentication Scheme", RFC 7617, DOI 10.17487/RFC7617, September 2015, https://www.rfc-editor.org/info/rfc7617.

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

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

[UNSD M49]
Standard country or area codes for statistical use (M49), UN Statistics Division (UNSD), 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.

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

[RFC7515]
Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, https://www.rfc-editor.org/info/rfc7515.

[RFC7516]
Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)", RFC 7516, DOI 10.17487/RFC7516, May 2015, https://www.rfc-editor.org/info/rfc7516.

[RFC8322]
Field, J., Banghart, S., and D. Waltermire, "Resource-Oriented Lightweight Information Exchange (ROLIE)", RFC 8322, DOI 10.17487/RFC8322, February 2018, https://www.rfc-editor.org/info/rfc8322.

[SemVer]
Tom Preston-Werner, "Semantic Versioning", Available: https://semver.org/

Appendix E. Acknowledgments

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

Bret Jordan, Broadcom
Stephanie Hazlewood, IBM
Emily Ratliff, IBM
Allan Thomson, Individual
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)
Bret Jordan, Broadcom
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
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.
Stephen Banghart, NIST
David Darnell, North American Energy Standards Board
Lior Kolnik, Palo Alto Networks
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

Other Contributions:

We would also like to specifically thank Kamer Vishi, University of Oslo for the CACAO logo.

Appendix F. Revision History

Revision	Date	Editor(s)	Changes Made
00.01	2020-03-27	Bret Jordan, Allan Thomson	Initial Version
00.02	2020-04-21	Bret Jordan, Allan Thomson	Added terminology, actions, targets, and data markings. A lot of editorial cleanup.
00.03	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. Submitted to be approved as CSD01.
01.04	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.
01.05	2020-12-01	Bret Jordan, Allan Thomson	Changed 7.7 sector target to have a list of physical locations. Submitted to be approved as CSD02.

Appendix G. Notices