CACAO Security Playbooks Version 2.0

Committee Specification Draft ~~02~~03

~~26 May~~25 August 2023

This version:
https://docs.oasis-open.org/cacao/security-playbooks/v2.0/csd03/security-playbooks-v2.0-csd03.docx (Authoritative)
https://docs.oasis-open.org/cacao/security-playbooks/v2.0/csd03/security-playbooks-v2.0-csd03.html
https://docs.oasis-open.org/cacao/security-playbooks/v2.0/csd03/security-playbooks-v2.0-csd03.pdf

Previous version:
https://docs.oasis-open.org/cacao/security-playbooks/v2.0/csd02/security-playbooks-v2.0-csd02.docx (Authoritative)
https://docs.oasis-open.org/cacao/security-playbooks/v2.0/csd02/security-playbooks-v2.0-csd02.html
https://docs.oasis-open.org/cacao/security-playbooks/v2.0/csd02/security-playbooks-v2.0-csd02.pdf

~~**Previous version:**~~
~~(Authoritative)~~

Latest version:
https://docs.oasis-open.org/cacao/security-playbooks/v2.0/security-playbooks-v2.0.docx (Authoritative)
https://docs.oasis-open.org/cacao/security-playbooks/v2.0/security-playbooks-v2.0.html
https://docs.oasis-open.org/cacao/security-playbooks/v2.0/security-playbooks-v2.0.pdf

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

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

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

Related Work:
This specification replaces or supersedes:

- *CACAO Security Playbooks Version 1.0*. Edited by Bret Jordan and Allan Thomson. 08 June 2021. OASIS Committee Specification 02. Latest version: https://docs.oasis-open.org/cacao/security-playbooks/v1.0/security-playbooks-v1.0.html.

This document is related to:

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

Abstract:

To defend against threat actors and their tactics, techniques, and procedures organizations need to detect, investigate, prevent, mitigate, and remediate threats in cyber relevant time. To do this, organizations need to identify, create, document, and test the orchestration steps needed to achieve these outcomes. 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) for cyber security playbooks and describes how these playbooks can be created, documented, and shared in a structured and standardized way across organizational boundaries and technological solutions.

Status:

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

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

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

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

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-v2.0]

CACAO Security Playbooks Version 2.0. Edited by Bret Jordan and Allan Thomson. ~~26 May~~25 August 2023. OASIS Committee Specification Draft ~~02~~03. https://docs.oasis-open.org/cacao/security-playbooks/v2.0/csd03/security-playbooks-v2.0-csd03.html. Latest version: https://docs.oasis-open.org/cacao/security-playbooks/v2.0/security-playbooks-v2.0.html.

Notices:

Table of Contents

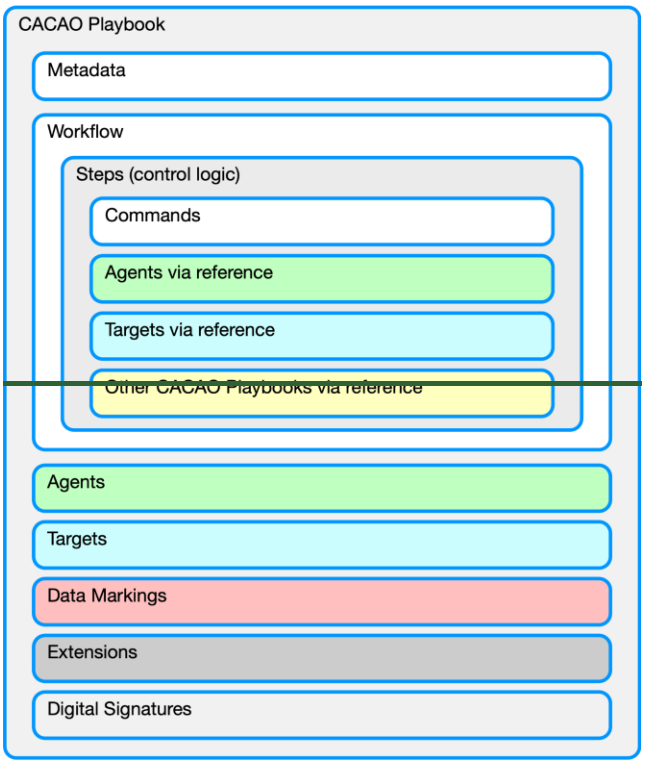
1 Introduction

To defend against threat actors and their tactics, techniques, and procedures organizations need to detect, investigate, prevent, mitigate, and remediate threats in cyber relevant time. To do this, organizations need to identify, create, document, and test the orchestration steps needed to achieve these outcomes. 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) for cyber security playbooks and describes how these playbooks can be created and shared in a structured and standardized way across organizational boundaries and technological solutions.

1.1 Overview of CACAO Playbook Structure and Classes of Objects

This specification defines the following classes of objects: playbooks (section 3), workflow steps (section 4), commands (section 5), authentication types (section 6), agents (section ~~6~~7), targets (section ~~6~~7), extensions (section ~~7~~8), data markings (sections 2.4, ~~8~~9), and digital signatures (sections 2.5, ~~9.13~~10.15). Within a playbook, these objects are grouped and organized as shown in Figure 1.



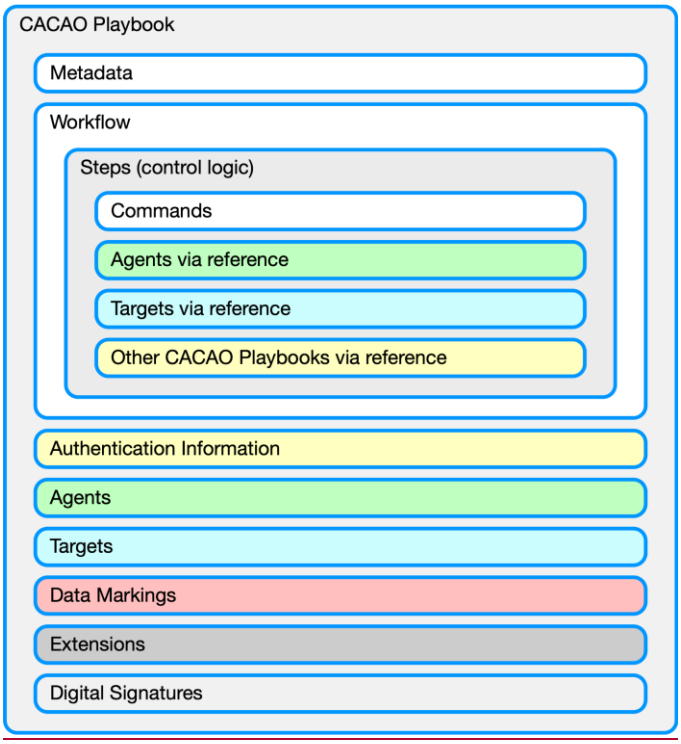


Figure 1: CACAO Playbook Structure

1.2 Playbook

CACAO standardizes the definition and use of two important concepts often used by organizations to protect themselves or the broader ecosystem they connect with: **action** and **playbook**.

An action represents every security activity or operation (referred to as a **security action**, or just **action**) that an organization may take to detect, investigate, prevent, mitigate or remediate a specific security state that has either occurred or may occur.

A playbook is a workflow for security orchestration containing a set of steps (**security actions)** to be performed based on a logical process and may be executed ad-hoc, periodically, or triggered by an automated or manual event or an observation. A playbook provides guidance on how to address a certain security event, incident, problem, attack, or compromise. Playbooks can be written by the organization, an entity external to the organization, a sharing community, or a vendor.

A playbook may be defined in one system by one or more authors, but then executed in a different operational environment where the systems and users may have different authentication and authorization requirements.

Even though rare, it may be the case that shared playbooks may be immediately actionable in an organization’s security infrastructure without requiring modification or updates to the workflow and commands. However, most will not be immediately usable or supported in the recipient's organization due to environmental differences and will require some amount of modification to make them operational. In addition, some playbooks may be purposefully written at a higher level of abstraction in order to describe key concepts and tactics that should be followed in a sequence to address a security event, incident, problem, attack, or compromise.

1.2.1 Referencing Other Playbooks

A playbook may reference (invoke) other playbooks in such a manner that allows composition from smaller, more specific functional playbooks, similar to how software application development leverages modular libraries of common functions shared across different applications.

1.3 Playbook Types

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

Playbook Type	Description
Attack Playbook	A playbook that is primarily focused on the orchestration steps required to execute a penetration test or perform adversarial emulation. These playbooks can help an organization test and verify the security controls in a specific environment and potentially identify vulnerabilities or other changes necessary to improve defensive posture within that environment. For example, an attack playbook can contain the specific actions that a red-team should perform that are within the scope and rules of engagement for a specific penetration test. An attack playbook may also be used to capture, in a structured way, the sequence of an adversary's behavior as described in a text-based cyber threat intelligence (CTI) report.
Detection Playbook	A playbook that is primarily focused on the orchestration steps required to detect a known security event, known or expected security-relevant activity, or for threat hunting. For example, a detection playbook can contain the actions needed to help organizations detect a specific attack or campaign.
Engagement Playbook	A playbook that is primarily focused on the orchestration steps required to engage in denial, deception, strategic planning, and analysis activity to support adversary engagement. Whereas attack playbooks leverage actions against known defenders to test an environment, engagement playbooks define actions and countermeasures against adversaries to increase their cost to operate and decrease the value of their operations. For example, an engagement playbook can contain the actions needed to provide misinformation about data or systems to decrease the value an adversary places on the assets, or it can contain the actions needed to disrupt network access to increase the adversary’s operational costs. See [Engage].
Investigation Playbook	A playbook that is primarily focused on the orchestration steps required to investigate what affects a security event, incident, or other security-relevant activity has caused. Investigation playbooks will likely inform other subsequent actions upon completion of the investigation. For example, an investigation playbook can contain the actions needed to check various systems for suspicious activity.
Mitigation Playbook	A playbook that is primarily focused on the orchestration steps required to mitigate a security event or incident that has occurred when remediation is not initially possible. Organizations often choose to mitigate a security event or incident until they can actually remediate it. Mitigation playbooks are designed to reduce or limit the impact of suspicious or confirmed malicious

	activity. For example, a mitigation playbook can contain the specific actions to 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.
Notification Playbook	A playbook that is primarily focused on the orchestration steps required to notify and disseminate information and other playbooks about a security event, incident, or threat. For example, a notification playbook can be used to notify multiple entities about a new attack or campaign and disseminate information or playbooks to deal with it as quickly as possible.
Prevention Playbook	A playbook that is primarily focused on the orchestration steps required to prevent a known or expected security event, incident, or threat from occurring. Prevention playbooks are often designed and deployed as part of best practices to safeguard organizations from known and perceived threats and behaviors associated with suspicious activity. For example, a prevention playbook can contain the specific actions that need to be deployed on certain systems to prevent a new attack or campaign.
Remediation Playbook	A playbook that is primarily focused on the orchestration steps required to remediate, resolve, or fix the resultant state of a security event or incident and return the system, device, or network back to a nominal operating state. Remediation playbooks can fix affected assets by selectively correcting problems due to malicious activity by reverting the system or network to a known good state. For example, a remediation playbook can contain the specific actions that need to be deployed to ensure that a system or device is no longer infected with some malware. If mitigation steps were previously applied, they might need to be undone during remediation; however, this is all implementation specific and dependent on how the playbooks were created and executed.

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 CTI. This enables organizations not only to know and understand threats, behaviors, and associated intelligence, but also to 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 redefining those schemas or constructs within CACAO itself. This version of the specification uses the following concepts from other standards:

- *Identity* object from STIX 2.1 [STIX-v2.1]
- *Relationship* object from STIX 2.1 [STIX-v2.1]
- *Patterning* from STIX 2.1 [STIX-v2.1]

1.6 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 (45_{dec} or $2D_{hex}$), which in Unicode is "hyphen-minus", U+002D.
- The IDs used in examples are notional and for illustrative purposes, they do not represent real objects.
- Some URLs have been defanged. This specification gives no guidance on how to defang or re-fang content. It is done to help ensure that the example URLs cannot be used directly.

Several objects in this specification reference a STIX Identity object for the CACAO TC in their created_by property. That object is as follows and can be found in the CACAO Common Objects GitHub repository (https://github.com/oasis-tcs/cacao/tree/master/Common-Objects/Identity).

```
{
  "type": "identity",
  "spec_version": "2.1",
  "id": "identity--5abe695c-7bd5-4c31-8824-2528696cdbf1",
  "created_by_ref": "identity--8ce3f695-d5a4-4dc8-9e93-a65af453a31a",
  "created": "2023-06-20T10:00:00.000Z",
  "modified": "2023-06-20T10:00:00.000Z",
  "name": "OASIS Collaborative Automated Course of Action Operations (CACAO) for Cyber
Security TC",
  "description": "A global community of cyber security experts creating standards enabling
increased automation, collaboration, and effectiveness of the global response to cyber
threats.",
  "roles": [ "Technical Committee" ],
  "identity_class": "organization",
  "sectors": [ "non-profit" ],
  "contact_information": "cacao-comment@lists.oasis-open.org"
}
```

1.7 Changes From The Previous Version

- Added three command types (caldera-cmd, elastic, and yara), added industry sector to the playbook metadata
- Made many improvements based on public review
- Made improvements and simplifications to the signing process
- Made a lot of editorial changes
- Ensured IDs are consistently represented

- Made changes to the UUIDv5 aspects
- Changed all logic actions to only allow a single action not a list of actions so we consistently use the parallel action
- Added playbook activities, removed playbook-templates
- Changed the TLP data markings to support TLPv2.
- Variables now use two underscores instead of two dollar signs, this is to make it work with the STIX Patterning Grammar
- Playbook Features was renamed to Playbook Processing Summary
- Renamed Targets to "Agents and Targets"
- Renamed Security Infrastructure Category to just Security Category
- Renamed Agents and Targets in the playbook properties to agent_definitions and target_definitions
- Added support for quantums safe algorithms in the signature object
- Changed the signature vocab to be an open-vocabulary instead of an enum
- Added an authentication information object to abstract that away from the Agents and Targets

1.8 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
SOC - Security Operation Center
STIX - Structured Threat Information Expression
TLP - Traffic Light Protocol
ℕ - Natural numbers / counting numbers as defined as the set {1,2,3…} or {x: x > 0}
𝕎 - Whole numbers as defined as the set {0,1,2,3…} or {x: x >= 0}

2 Key Concepts and Features

This section explains some of the key concepts and features used in CACAO.

2.1 Vocabularies

Some properties in this specification use defined vocabularies. These vocabularies can be either open or closed. An open vocabulary (see section ~~9.11~~10.13) 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. Open vocabulary types have an `-ov` suffix. A closed vocabulary is effectively an enumeration and **MUST** be used as defined. Enumeration types have an `-enum` suffix.

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.

2.2 Playbook Creator

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

Entities that re-publish an object from another entity without making any changes to the object, and thus maintaining the original value in the **`id`** property, are not considered the object creator and **MUST NOT** change the **`created_by`** property. An entity that accepts objects and republishes them with modifications, additions, or omissions **MUST** create a new **`id`** and **MUST** change the **`created_by`** property for the object as they are now considered the object creator of the new object for purposes of versioning (see section 2.3 versioning for more information).

2.3 Versioning

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

Playbooks **MAY** be versioned in order to update, add, or remove information. A version of a playbook is identified uniquely by the combination of its **`id`** and **`modified`** properties. The first version of a playbook **MUST** have the same timestamp for both the **`created`** and **`modified`** properties. More recent values of the **`modified`** property indicate later versions of the playbook. Implementations **MUST** consider the version of the playbook with the most recent **`modified`** value to be the most recent version of the playbook. For every new version of a playbook, the **`modified`** property **MUST** be updated to represent the time that the new version was created. This specification does not define how to handle a consumer receiving two objects that are different, but have the same **`id`** and **`modified`** timestamp. This specification does not address how implementations should handle versions of the object that are not current.

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

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

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

2.3.1 Versioning Timestamps

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

2.3.2 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 any relationships (e.g., STIX 2.1 relationship objects) to the playbook from other data when deciding if a change is material. If the change would invalidate the usefulness of relationships to the playbook, then the change is considered material and a new playbook **id** **SHOULD** be used.

2.4 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 they must not be re-shared, or that they 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.

Any change to the **`markings`** property (e.g., adding or removing a marking) is treated the same as a change to any other property on the object and thus follows the same rules for versioning. It is important to note that data markings themselves **MUST NOT** be versioned.

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

2.5 Signing Playbooks

The process of digitally signing CACAO playbooks is conformant with the JSON Signature Scheme (X.jss) and allows digitally signed playbooks to remain in JSON format. CACAO playbooks can be signed and or countersigned depending on the needs of the ecosystem, trust group, or organization. The CACAO signature design supports both including the signature in the playbook itself and storing or sharing the signature separately as a detached signature. When signing a CACAO playbook, the signer **MUST NOT** include any existing signatures in the playbook data, unless they are specifically countersigning. Adding a signature to a playbook does not constitute a revision or change to the playbook and as such, the **`modified`** `timestamp` **MUST NOT** be updated. See Appendix A for a detailed example.

2.5.1 ~~Signing~~Create Signature Steps

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

Step 1: Create or parse the playbook to be signed
Step 2: Temporarily remove existing signature from the playbook
Step 3: Create and add signature object to the playbook from step 2
Step 4: Create a JCS [RFC8785] canonical version of entire playbook from step 3
Step 5: Create a hash based on the hash algorithm defined in the signature object (e.g., sha-256 in hex) of the JCS version of the entire playbook from step 4
Step 6: Sign the hash from step 5 using the algorithm (e.g., RS256) defined in the signature object and base64URL.encode it
Step 7: Append the new b64 digital signature from step 6 to the signatures value property (also include any existing signatures, that were removed in step 2)

2.5.2 Verify ~~Signing~~Signature Steps

The steps involved in verifying a signed CACAO playbook are as follows (see Appendix A for more details):

Step 1: Parse the signed playbook to verify
Step 2: Capture and remove the digital signature from the value property of the signature you want to verify (also remove any other signatures that may be included in the playbook)
Step 3: Parse or fetch the public key from step 2

Step 4: Create a JCS [RFC8785] canonical version of entire playbook from step 2
Step 5: Create a hash based on the hash algorithm defined in the signature object (e.g., sha-256 in hex) of the JCS version of the entire playbook from step 4
Step 6: Verify the signature received using the public key and algorithm that is defined in the signature object

2.5.3 Verify Countersigned Signature Steps

The steps involved in verifying a countersigned CACAO playbook are as follows (see Appendix A for more details):

Step 1: Parse the signed playbook to verify
Step 2: NOTE: A signature can have a countersigned signature. This countersigned signature exists inside the top-level signature and can be found in a property called "signature". During this step, identify the signature that has been countersigned that you want to verify and remove any other signatures that may be included with the playbook. Then capture and remove the countersigned digital signature from the value property of the signature you want to verify.
Step 3: Parse or fetch the public key from step 2
Step 4: Create a JCS [RFC8785] canonical version of entire playbook from step 2
Step 5: Create a hash based on the hash algorithm defined in the signature object (e.g., sha-256 in hex) of the JCS version of the entire playbook from step 4
Step 6: Verify the signature received using the public key and algorithm that is defined in the signature object

3 Playbooks

CACAO playbooks are made up of six parts~~;~~ playbook metadata, the workflow logic, a list of object definitions used in the workflow logic (agents and targets), a list of extensions, a list of data markings, and a list of any digital signatures. Playbooks **MAY** refer to other playbooks in the workflow, similar to how programs refer to function calls or modules that comprise the program. The definition and normative requirements for all data types listed in the property table below and other property tables in this document can be found in Section ~~9~~10.

3.1 Playbook Properties

Property Name	Data Type	Details
type (required)	`string`	The value of this property **MUST** be `playbook`.
spec_version (required)	`string`	The version of the specification used to represent this playbook. The value of this property **MUST** be `cacao-2.0` to represent the version of this specification.
id (required)	`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 (required)	`string`	A name for this playbook. Playbook names often follow a naming convention that is unique within an organization, community, or trust group and as such this name **SHOULD** be unique.
description (optional)	`string`	More details, context, and possibly an explanation about what this playbook does and tries to accomplish. Producers **SHOULD** populate this property.
playbook_types (optional)	`list` of `open-vocab`	A list of playbook types that specifies the operational roles that this playbook addresses. This property **SHOULD** be populated. The values for this property **SHOULD** come from the `playbook-type-ov` vocabulary (see section 3.1.1).
playbook_activities (optional)	`list` of `open-vocab`	A list of activities pertaining to the playbook. This property **SHOULD** be populated. If the **playbook_types** property is populated and comes from the `playbook-type-ov` then this property **MUST** have at least one assigned activity. This property allows an author to define more detailed descriptions for the various activities that a playbook performs. This property provides a much richer and verbose method to describe all aspects of a playbook than just the **playbook_types** property.

		The values for this property **SHOULD** come from the `playbook-activity-type-ov` vocabulary (see section 3.1.2). Each listed activity **MUST** be reflected in a CACAO workflow step object and that object **MUST** be included in the **`workflow`** property.
`playbook_processing_summary` (optional)	`playbook-processing-summary`	This property contains a summarized list of processing features that are defined within this playbook. This property enables the content of a playbook to be assessed without requiring the entire content to be parsed or understood. See section ~~9.12~~10.14.
`created_by` (required)	`identifier`	An ID that represents the entity that created this playbook. The ID **MUST** represent a STIX 2.1+ identity object.
`created` (required)	`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 given 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` (required)	`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 given 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. If **`created`** and **`modified`** properties are the same, then this is the first version of the playbook.
`revoked` (optional)	`boolean`	A boolean that identifies if the playbook creator deems that this playbook is no longer valid. The default value is `false`.
`valid_from` (optional)	`timestamp`	The time from which this playbook is considered valid and the workflow 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`**. If the **revoked** property is `true` then this property **MUST** be ignored.
`valid_until` (optional)	`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. If the **revoked** property is `true` then this property **MUST** be ignored.
derived_from (optional)	`list` of `identifier`	The ID of one or more CACAO playbooks that this playbook was derived from. The ID **MUST** represent a CACAO playbook object.
~~**derived_from**~~**related_to** (optional)	`list` of `identifier`	The ID of one or more ~~playbooks~~related STIX or CACAO objects that this playbook ~~was derived from.~~ is related to. The ID **SHOULD** represent a CACAO playbook object, but **MAY** represent any STIX v2.1 CTI object or ~~other playbook (so long as there is~~ a ~~unique identifier for that object) that this playbook was derived from~~TC approved extension.
priority (optional)	`integer`	A number (𝕎 - whole number) that represents the priority of this playbook relative to other defined playbooks. Priority in the context of CACAO is a subjective assessment; thus, producers of playbooks, sharing organizations, and marketplaces **MAY** define rules on how priority should be assessed and assigned. This specification does not address how this assessment is determined. This property is primarily to allow such usage without requiring the addition of a custom property 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. The values of 1-100 in this property are inverted from **severity** and **impact** based on how the concept of priority is used today. For example, in a SOC a P1 ticket is a higher priority than a P4 ticket.
severity (optional)	`integer`	A number (𝕎 - whole number) that represents the seriousness of the conditions that this playbook addresses. This is highly dependent on whether the playbook is a response to an incident (in which case the severity could be mapped to an incident category defined in some solution), a response to a threat (in which case the severity would likely be mapped to the severity of the threat faced or captured by threat intelligence), or a response to something else. Marketplaces and sharing organizations **MAY** define

		additional rules for how this property should be assigned. This specification does not address how this assessment is determined. 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 (optional)	`integer`	A number (𝕎 - whole number) from 0 to 100 that represents the potential impact (as determined subjectively by the producer) the execution of the playbook might have on the organization and its infrastructure. 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 or benign. Impact values range from 1, the lowest impact, to a value of 100, the highest. Marketplaces and sharing organizations **MAY** define additional rules for how this property should be assigned. This specification does not address how this assessment is determined. NOTE: The value of this property is not related to what triggered the playbook in the first place, such as a threat or an incident. Executing a playbook with a higher impact score may increase the likelihood of an effect on the organization. For example, a purely investigative playbook that is non-invasive could have a low impact value of 1. In contrast, a playbook that performs firewall changes, IPS changes, moves laptops to a quarantine VLAN etc., would have a higher impact value.
industry_sectors (optional)	`list` of `open-vocab`	A list of industry sectors that this playbook may be related or applicable to. Any industry sectors that are used in other parts of this playbook **MUST** also be included in this property. Any industry sectors that are used in other referenced playbooks **MAY** also be included in this property. The values for this property **SHOULD** come from the `industry-sector-ov` vocabulary.
labels (optional)	`list` of `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	An optional list of external references for this playbook or

(optional)	`external-reference`	content found in this playbook. Any external references that are used in other parts of this playbook **MUST** also be included in this property. Any external references that are used in other referenced playbooks **MAY** also be included in this property.
`markings` (optional)	`list` of `identifier`	An optional list of data marking objects that apply to this playbook. In some cases, though uncommon, data markings themselves may be marked with sharing or handling guidance. In this case, this property **MUST NOT** contain any references to the same data marking object (i.e., it cannot contain any circular references). Each ID **MUST** represent a CACAO data marking object.
`playbook_variables` (optional)	`dictionary`	This property contains the global variables (see section ~~9.16~~10.18.1 for information about variable scope) that can be used within the playbook, workflow steps~~,~~ (including conditional steps), commands, agents, and targets defined within the playbook. See section ~~9.16~~10.18 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 ~~9.16~~10.18).
`workflow_start` (required)	`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 start step object and that object **MUST** be included in the **`workflow`** property. This property is an implementation helper so that the entire workflow does not need to be parsed to find the start step.
`workflow_exception` (optional)	`identifier`	The workflow step invoked whenever a playbook execution exception condition occurs. If defined, the ID **MUST** represent a CACAO workflow step object and that object **MUST** be included in the **`workflow`** property.
`workflow` (required)	`dictionary`	The **`workflow`** property contains the processing logic for the playbook as workflow steps. All playbooks **MUST** contain at least the following three steps: a start step, an action/playbook-action step, and an end step. The key for each entry in the dictionary **MUST** be an `identifier` that uniquely identifies the workflow step (see section ~~9.9~~10.10 for more information on identifiers). The value for each key **MUST** be a CACAO workflow step object (see section 4).

playbook_extensions (optional)	dictionary	This property declares the extensions that are in use on this playbook and contains the properties and values that are to be used by the extensions. The key for each entry in the dictionary **MUST** be an identifier (see section 10.10 for more information on identifiers) that uniquely identifies the extension. The value for each key is a JSON object that contains the structure as defined in the extension definition's schema property. The actual property extension definition is located in the extension_definitions property.
~~agent~~**authentication_info_definitions** (optional)	dictionary	A dictionary of ~~agents~~ authentication information that can be referenced from agents and targets in workflow steps found in the **workflow** property. The authentication information can be used by agents and targets when performing interactions that require authentication. The key for each entry in the dictionary **MUST** be an identifier that uniquely identifies the ~~agent~~authentication information (see section ~~9.9~~10.10 for more information on identifiers). The value for each key **MUST** be a CACAO ~~agent-target~~authentication-info object (see section 6). Any ~~agents~~authentication information that ~~are~~is used in other parts of this playbook **MUST** also be included in this property. ~~Any agents that are used in other referenced playbooks **MAY** also be included in this property.~~
agent_definitions (optional)	dictionary	A dictionary of agents 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 agent (see section 10.10 for more information on identifiers). The value for each key **MUST** be a CACAO agent-target object (see section 7). Any agents that are used in other parts of this playbook **MUST** also be included in this property.
target_definitions (optional)	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 (see section ~~9.9~~10.10 for more information on identifiers). The value for each key **MUST** be a CACAO agent-target object (see section ~~6~~7). Any targets that are used in other parts of this playbook **MUST** also be included in this property. ~~Any targets that~~

		~~are used in other referenced playbooks **MAY** also be included in this property.~~
extension_definitions (optional)	`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 (see section ~~9.9~~10.10 for more information on identifiers). The value for each key **MUST** be a CACAO extension object (see section 7). Any extensions that are used in other parts of this playbook **MUST** also be included in this property. ~~Any extensions that are used in other referenced playbooks **MAY** also be included in this property.~~
data_marking_definitions (optional)	`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 (see section ~~9.9~~10.10 for more information on identifiers). The value for each key **MUST** be a CACAO data marking object (see section ~~8~~9). Any data markings that are used in other parts of this playbook **MUST** also be included in this property. ~~Any data markings that are used in other referenced playbooks **MAY** also be included in this property.~~
signatures (optional)	`list` of `signature`	An optional list of digital signatures for this playbook. Adding a signature to a playbook does not represent a version change of the playbook. See sections 2.5, ~~9.13~~10.15, and Appendix A for more information and a detailed example.

Example 3.1

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

```
{
  "type": "playbook",
  "spec_version": "cacao-2.0",
  "id": "playbook--61a6c41e-6efc-4516-a242-dfbc5c89d562",
  "name": "Find Malware FuzzyPanda",
  "description": "This playbook will look for FuzzyPanda on the network and in a SIEM",
  "playbook_types": [ "investigation" ],
  "playbook_activities": [ "analyze-collected-data", "identify-indicators" ],
  "playbook_processing_summary": {
    "data_markings": true
  },
  "created_by": "identity--5abe695c-7bd5-4c31-8824-2528696cdbf1",
  "created": "2023-02-19T08:00:24.918Z",
  "modified": "2023-02-19T08:00:24.918Z",
  "valid_from": "2023-02-19T08:00:24.918Z",
  "valid_until": "2023-12-31T23:59:59.999Z",
  "derived_from": [ "playbook--00ee41a2-c2ca-41da-8ea9-681344eb3926" ],
```

```
  "priority": 3,
  "severity": 70,
  "impact": 5,
  "industry_sectors": [ "aerospace", "defense" ],
  "labels": [ "malware", "fuzzypanda", "apt" ],
  "external_references": [
    {
      "name": "ACME Security FuzzyPanda Report",
      "description": "ACME security review of FuzzyPanda 2021",
      "source": "ACME Security Company, Solutions for FuzzyPanda 2021, January 2021. Available
online: hxxp://www[.]example[.]com/info/fuzzypanda2021.html",
      "url": "hxxp://www[.]example[.]com/info/fuzzypanda2021.html",
      "external_id": "fuzzypanda 2023.01",
      "reference_id": "malware--2008c526-508f-4ad4-a565-b84a4949b2af"
    }
  ],
  "markings": [
    "marking-statement--6424867b-0440-4885-bd0b-604d51786d06",
    "marking-tlp--bab4a63c-aed9-4cf5-a766-dfca5abac2bb"
  ],
  "playbook_variables": {
    "__data_exfil_site__": {
      "type": "ipv4-addr",
      "description": "The IP address for the data exfiltration site",
      "value": "1.2.3.4"
    }
  },
  "workflow_start": "start--07bea005-4a36-4a77-bd1f-79a6e4682a13",
  "workflow_exception": " ... ",
  "workflow": {
    "start--07bea005-4a36-4a77-bd1f-79a6e4682a13": {
      "type": "start",
      "name": "Start Playbook Example 1",
      "on_completion": "action--7f40f9d7-de39-4027-ab97-15035beff2ff"
    },
    "action--7f40f9d7-de39-4027-ab97-15035beff2ff": {
      "type": "action",
      "name": "IP Lookup",
      "description": "Lookup the IP address in the SIEM",
      "on_completion": "end--6b23c237-ade8-4d00-9aa1-75999738d557",
      "commands": [
        {
          "type": "manual",
          "command": "Look up IP __data_exfil_site__:value in SIEM",
          "playbook_activity": "identify-indicators"
        }
      ]
    },
    "end--6b23c237-ade8-4d00-9aa1-75999738d557": {
      "type": "end",
      "name": "End Playbook Example 1"
    }
  },
  "playbook_extensions": { ... },
  "authentication_info_definitions": { ... },
  "agent_definitions": { ... },
  "target_definitions": { ... },
  "extension_definitions": { ... },
  "data_marking_definitions": {
    "marking-statement--6424867b-0440-4885-bd0b-604d51786d06": {
      "type": "marking-statement",
      "id": "marking-statement--6424867b-0440-4885-bd0b-604d51786d06",
```

```
      "created_by": "identity--5abe695c-7bd5-4c31-8824-2528696cdbf1",
      "created": "2023-02-19T08:00:24.918Z",
      "modified": "2023-02-19T08:00:24.918Z",
      "statement": "Copyright 2023 ACME Security Company"
    },
    "marking-tlp--bab4a63c-aed9-4cf5-a766-dfca5abac2bb": {
      "type": "marking-tlp",
      "id": "marking-tlp--bab4a63c-aed9-4cf5-a766-dfca5abac2bb",
      "created_by": "identity--5abe695c-7bd5-4c31-8824-2528696cdbf1",
      "created": "2022-10-01T00:00:00.000Z",
      "modified": "2022-10-01T00:00:00.000Z",
      "tlpv2_level": "TLP-:GREEN"
    }
  },
  "signatures": [ ... ]
}
```

3.1.1 Playbook Type Vocabulary

A playbook may be categorized as having multiple types defined from this vocabulary.

Vocabulary Name: `playbook-type-ov`

Vocabulary Value	Description
`attack`	See section 1.3 for an explanation.
`detection`	See section 1.3 for an explanation.
`engagement`	See section 1.3 for an explanation.
`investigation`	See section 1.3 for an explanation.
`mitigation`	See section 1.3 for an explanation.
`notification`	See section 1.3 for an explanation
`prevention`	See section 1.3 for an explanation.
`remediation`	See section 1.3 for an explanation.

3.1.2 Playbook Activity Type Vocabulary

Playbook activity differs according to playbook type. Activity type values and descriptions are given below, organized by playbook type: Notification (N), Detection (D), Investigation (I), Prevention (P), Mitigation (M), Remediation (R), Attack (A), Engagement (E). Required activities (indicated by **M**'s) in **bold face** font) uniquely identify a playbook type and **MUST** be defined; optional activities **SHOULD** (indicated by S's) or **MAY** (indicated by O's) be associated with one or more playbook types.

Enumeration Name: `playbook-activity-type-ov`

ActivityType / Description	N	D	I	P	M	R	A	E
`compose-content` - This activity composes the notification text that will be distributed. This activity **MUST** be used with notification playbooks.	**M**							
`deliver-content` - This activity delivers notification content to the intended audience. This activity **SHOULD** be used with notification playbooks.	S							
`identify-audience` - This activity identifies the audience of a notification. This activity **SHOULD** be used with notification playbooks.	S							
`identify-channel` - This activity identifies the method by which notification content will be sent. This activity **SHOULD** be used with notification playbooks.	S							
`scan-system` - This activity scans a system (workstation, server, network device) to identify potential compromises. This activity **SHOULD** be used with detection, investigation, and mitigation playbooks.		S	S		S			
`match-indicator` - This activity matches on an indicator through traffic monitoring, system scans, and log analysis. This activity **MUST** be used with detection playbooks.		**M**						
`analyze-collected-data` - This activity analyzes historical output from security devices (e.g., logs and network traffic capture). This activity **SHOULD** be used with investigation playbooks.			S					
`identify-indicators` - This activity identifies one or more indicators that can be used to detect a security event. This activity **MUST** be used with investigation playbooks.			**M**					
`scan-vulnerabilities` - This activity identifies vulnerabilities of a system. This activity **SHOULD** be used with prevention playbooks and **MAY** be used with attack playbooks.				S			O	
`configure-systems` - This activity confirms secure configuration and if necessary, updates or configures systems or security devices to be resistant to a security event. This activity **MUST** be used with prevention playbooks.				**M**				
`restrict-access` - This activity blocks applications and network traffic (ports/IP addresses/URLs) to mitigate a security event. This activity **SHOULD** be used with mitigation playbooks.					S			
`disconnect-system` - This activity disconnects a compromised system from the network. This activity **MAY** be used with mitigation playbooks.					O			
`eliminate-risk` - This activity eliminates the risk that a threat will affect a network by restricting capabilities. This activity **MUST** be used with mitigation playbooks.					**M**			
`revert-system` - This activity reimages a system returning it to a known-good state. This activity **MAY** be used with remediation playbooks.						O		
`restore-data` - This activity restores data after a compromise (e.g., ransomware). This activity **MAY** be used with remediation playbooks.						O		

`restore-capabilities` - This activity restarts services and opens network access. This activity **MUST** be used with remediation playbooks.						**M**		
`map-network` - This activity maps a network to identify components that may be subject to compromise and to ensure the environment meets requirements for subsequent actions, such as a penetration test or attack simulation. This activity **MAY** be used with attack playbooks.							O	
`identify-steps` - This activity identifies steps (tactics, techniques, and protocols) for use in a penetration test, attack simulation, or adversary emulation plan. These steps will become the step sequence. This activity **MAY** be used with attack playbooks (alternatively, an attack playbook may comprise only the steps pertaining to the operation).							O	
`step-sequence` - This activity is a sequence of TTPs or steps that represents an adversary emulation plan, penetration test, or attack simulation. This activity **MUST** be used with attack playbooks.							**M**	
`prepare-engagement` - This activity identifies what the defender wants to accomplish with respect to engaging (and misleading) an adversary and determines and aligns an operation with a desired end-state. This activity **MUST** be used with engagement playbooks.								**M**
`execute-operation` - This activity implements and deploys denial and deception activities designed for adversary engagement. This activity **MUST** be used with engagement playbooks.								**M**
`analyze-engagement-results` - This activity turns operational engagement outputs into actionable intelligence. Assessment of the intelligence enables capture of lessons learned and refinement of future adversary engagements. This activity **MUST** be used with engagement playbooks.								**M**

3.2 Playbook Activity Metadata

The following three playbook-level properties provide specific metadata about a playbook; these properties facilitate and support the searching and indexing of playbooks. Below, we describe the metadata purpose of each property and give search examples that a search engine might support.

3.2.1 playbook_types Property

`playbook_types`	See actual property definition in section 3.1.

Metadata Purpose

The **`playbook_types`** property allows an author to define what operational roles the playbook supports. In many cases, a single playbook may support multiple roles or phases of a response and as such can combine multiple operational roles.

This property is considered very important and **SHOULD** be defined by authors.

A typical search engine might support (using pseudo-SQL):

- To find all playbooks that are **investigative** in nature
  - `Select * playbooks where playbook_types contains "investigation"`
- To find all playbooks that include **both** mitigation and remediation
  - `Select * playbooks where playbook_types contains "mitigation" && playbook_types contains "remediation"`
- To find all playbooks that **either** include prevention or mitigation
  - `Select * playbooks where (playbook_types contains "mitigation" || playbook_types contains "prevention")`
- To find all playbooks that **are** mitigation playbooks
  - `Select * playbooks where playbook_types contains "mitigation"`

3.2.2 playbook_activities Property

playbook_activities	See actual property definition in section 3.1.

Metadata Purpose

The **playbook_activities** property allows an author to define more detailed descriptions for the various activities that the playbook performs. This property provides a much richer and verbose method to describe all aspects of a playbook including parts that are particularly focused on kill-chain activities.

This property is considered important for systems that require a detailed description of a playbook's activities and **SHOULD** be defined by authors. This property **MUST** be defined if the **playbook_types** property is populated.

A typical search engine might support:

- To find all playbooks that are ~~are~~ **investigative,** and include **scan systems** activities
  - `Select * playbooks where playbook_types contains "investigation" && playbook_activities contains "scan-system"`
- To find all playbooks that include **both** mitigation and remediation, and include activities **configure systems** and **restrict access**.
  - `Select * playbooks where playbook_types contains "mitigation" && playbook_activities contains "configure-systems" && playbook_activities contains "restrict-access"`
- To find all playbooks that are **either** mitigation or prevention playbook types, and include **identifying indicators** activities
  - `Select * playbooks where (playbook_types contains "mitigation" || playbook_types contains "prevention") && playbook_activities contains "identify-indicators"`
- To find all playbooks that include **identifying indicators** activities
  - `Select * playbooks where playbook_activities contains "identify-indicators"`

3.2.3 playbook_processing_summary Property

playbook_processing_summary	See actual property definition in section 3.1.

Metadata Purpose
The **playbook_processing_summary** property allows an author to define what specific types of processing features and logical constructs are used within a playbook. Typically this property can help determine whether a playbook system has the capability to support the logical constructs defined within the playbook.

This property is considered important for systems that require an understanding of the logical constructs (see section ~~9.12~~10.14) used in the playbook to determine whether they support the playbook, and **SHOULD** be defined by authors.

A typical search engine might support:

- To find all playbooks that are **investigative,** and contain **while logic** and **temporal logic.**
  - `Select * playbooks where playbook_types contains "investigation" && playbook_processing_summary contains "while_logic" && playbook_processing_summary contains "temporal_logic"`
- To find all playbooks that support **mitigation** that includes activity for **configure systems** and **restrict access,** including programming logic using **while logic** and **temporal logic.**
  - `Select * playbooks where playbook_types contains "mitigation" && playbook_activities contains "configure-systems" && playbook_activities contains "restrict-access" && playbook_processing_summary contains "while_logic" && playbook_processing_summary contains "temporal_logic"`
- To find all playbooks that include activities for **configure systems** regardless of purpose (i.e. no need to choose the playbook_types inclusion), using **while logic.**
  - `Select * playbooks where playbook.type eq "playbook~~-template~~" playbook_activities contains "configure-systems" playbook_processing_summary contains "while_logic"`
- To find all playbooks that are **investigative,** but do NOT use **parallel_processing.**
  - `Select * playbooks where playbook_types contains "investigation" && playbook_processing_summary does not contains "parallel_processing"`

3.3 Playbook Constants & Variables

Each playbook has a set of constants and variables that **MAY** be used throughout the execution of a playbook and its associated workflow (see section ~~9.16~~10.18).

Name	Description	Mutable	Type	Default Value
`__LOCAL_AGENT__`	A constant that defines ~~a~~an agent is local to the machine instance executing the current playbook.	No	`string`	"local_agent"
`__ACTION_TIMEOUT__`	A timeout variable in milliseconds that may be used to assign to a specific step timeout. Each	Yes	`integer`	60000 milliseconds

	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 step identified in the **`on_failure`** property of that step.			
`__RETURN_CALLER__`	This constant tells the executing program to return to the step that started the current branch. NOTE: this is similar to rolling back the stack in a computer program.	No	`string`	"return_caller"
`__RETURN_CALLER_ID__`	This constant defines a step to call upon completion or failure of a sub-step. This is typically used with parallel steps that define a tree of sub-steps to execute. This constant tells the executing program exactly which step ID it **MUST** return to.	yes	`identifier`	n/a

4 Workflows

Workflows contain a series of steps that are stored in a dictionary (see the **`workflow`** property in section 3.1), 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 steps are processed either sequentially, in parallel, or both depending on the type of steps required by the playbook. In addition to simple processing, workflow steps **MAY** also contain conditional and/or temporal operations to control the execution of the playbook.

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

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

4.1 Workflow Step Common Properties

Each workflow step contains some base properties that are common across all steps. These common properties are defined in the following table. The determination of a step being successful, failing, or completing is implementation specific and is out of scope for this specification, but details **MAY** be included in the **`description`** property.

Property Name	Data Type	Details
`type` (required)	`enum`	The type of workflow step being used. The value for this property **MUST** come from the `workflow-step-type-enum` enumeration.
`name` (optional)	`string`	A name for this step that is meant to be displayed in a user interface or captured in a log message.
`description` (optional)	`string`	More details, context, and possibly an explanation about what this step does and tries to accomplish.
`external_references` (optional)	`list` of `external-reference`	An optional list of external references for this step.
`delay` (optional)	`integer`	A number (𝕎 - whole number) that represents the amount of time in milliseconds that this step **SHOULD** wait before it starts processing. If specified, the value for this property **MUST** be greater than or equal to 0. If this property is omitted, then the workflow step executes immediately without delay.
`timeout` (optional)	`integer`	A number (𝕎 - whole number) that represents the amount of time in milliseconds that this step **MUST** wait before considering the step has failed.

		When a timeout has occurred for a step, the **`on_failure`** step pointer is invoked (if defined) and the information included in that call states that an ACTION_TIMEOUT occurred including the **`id`** of the step that timed out. If specified, the value of this property **MUST** be greater than or equal to 0. If this property 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` (optional)	`dictionary`	This property contains the variables that can be used within this workflow step or within commands, agents, and targets referenced by this workflow step. See section ~~9.16~~10.18.2 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 ~~9.16~~10.18.3).
`owner` (optional)	`identifier`	An ID that represents the entity that is assigned as the owner or responsible party for this step. The value of this property **MUST** represent a STIX 2.1+ Identity object.
`on_completion` (optional)	`identifier`	The ID of the next step to be processed upon completion of the defined commands. The value of this property **MUST** represent a CACAO workflow step object. If this property is defined, then **`on_success`** and **`on_failure`** **MUST NOT** be defined.
`on_success` (optional)	`identifier`	The ID of the next step to be processed if this step completes successfully. The value of this property **MUST** represent a CACAO workflow step object. If this property is defined, then **`on_completion`** **MUST NOT** be defined. This property **MUST NOT** be used on ~~an~~the `start` or `end` ~~step~~steps.

on_failure (optional)	identifier	The ID of the next step to be processed if this step fails to complete successfully. The value of this property **MUST** represent a CACAO workflow step object. If omitted and a failure occurs, then the playbook's exception handler found in the **workflow_exception** property at the Playbook level will be invoked. If this property is defined, then **on_completion** **MUST NOT** be defined. This property **MUST NOT** be used on ~~an~~the start or end ~~step~~steps.
step_extensions (optional)	dictionary	This property declares the extensions that are in use on this step and contains any of the properties and values that are to be used by that extension. The key for each entry in the dictionary **MUST** be an identifier (see section ~~9.9~~10.10 for more information on identifiers) that uniquely identifies the extension. The value for each key is a JSON object that contains the structure as defined in the extension definition's schema property. The actual step extension definition is located in the **extension_definitions** property found at the Playbook level.

4.2 Workflow Step Type Enumeration

Enumeration Name: workflow-step-type-enum

This section defines the following types of workflow steps.

Workflow Step Type	Description
start	This workflow step is the start of a playbook. See section 4.3.
end	This workflow step is the end of a playbook or branch of workflow steps. See section 4.4.
action	This workflow step contains the actual commands to be executed. See section 4.5.
playbook-action	This workflow step executes a named playbook from within the current playbook. See section 4.6.

`parallel`	This workflow step contains a list of one or more steps that execute in parallel. See section 4.7.
`if-condition`	This workflow step contains an if-then-else statement. See section 4.8.
`while-condition`	This workflow step contains a while loop. See section 4.9.
`switch-condition`	This workflow step contains a switch statement. See section 4.10.

4.3 Start Step

The Start Step workflow step is the starting point of a playbook and represents an explicit entry in the workflow to signify the start of a playbook. While this type inherits all of the common properties of a workflow step it does not define any additional properties. This workflow step **MUST NOT** use the **`on_success`** or **`on_failure`** properties.

Property Name	Data Type	Details
`type` (required)	`string`	The value of this property **MUST** be `start`.

Example 4.1
The IDs used in this example are notional and for illustrative purposes, they do not represent real objects.

```
{
  "workflow": {
    "start--27e1d0c1-2e08-4aa1-a062-176879a533e7": {
      "type": "start",
      "name": "Start Playbook Example 1",
      "on_completion": "action--ff537b6b-eca7-4137-aed5-331a750b7cbc"
    },
    ...
  }
}
```

4.4 End Step

The end step workflow step is the ending point of a playbook or branch of step (e.g., a list of steps that are part of a parallel processing branch) and represents an explicit point in the workflow to signify the end 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. This workflow step **MUST NOT** use the **`on_completion`**, **`on_success`**, or **`on_failure`** properties. While an end step **MUST** exist for the overall workflow, additional end steps **MAY** be present for workflow branches.

Property Name	Data Type	Details
`type` (required)	`string`	The value of this property **MUST** be `end`.

Example 4.2
The IDs used in this example are notional and for illustrative purposes, they do not represent real objects.

```
{
  "workflow": {
    ...,
    "end--09a02ab7-5e77-408b-a3e2-b468524329ac": {
      "type": "end",
      "name": "End Playbook Example 1"
    }
  }
}
```

4.5 Action Step

The action step workflow step contains the actual commands to be executed by an agent against a set of targets. These commands are intended to be processed sequentially. In addition to the inherited properties, this section defines five more specific properties that are valid for this type.

Property Name	Data Type	Details
type (required)	`string`	The value of this property **MUST** be `action`.
commands (required)	`list` of `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 (see section 5). All commands in a given step **MUST** be applicable to all the agents and all the targets defined in that step.
agent (required)	`identifier`	This property contains an ID reference to a CACAO `agent-target` object that is stored at the playbook level in the **agent_definitions** property. This agent **MUST** execute the commands defined in this step. As stated in section ~~6~~7, agents are the entities that execute commands on or against targets. The ID **MUST** reference a CACAO `agent-target` object (see section ~~6~~7).
targets (optional)	`list` of `identifier`	This property contains a list of ID references to CACAO `agent-target` objects that are stored at the playbook level in the target_definitions property. Each ID **MUST** reference a CACAO `agent-target` object (see section ~~6~~7). If defined, this list **MUST** have at least one identifier.
in_args (optional)	`list` of `string`	The list of variable names from the local **step_variables** dictionary (see section 4.1) or passed into this step from the global **playbook_variables** dictionary (see section 3.1) that are used in either an agent or one of the target(s) associated with this step. See section ~~9.16~~10.18 for more information about variables.
out_args (optional)	`list` of `string`	The ~~optional~~ list of variable names from the local **step_variables** dictionary (see section 4.1) or global **playbook_variables** dictionary (see section 3.1) that are

		to be returned ~~from~~to this step after execution of the commands by the agent(s~~) and stored in~~). Implementations **SHOULD** strongly discourage the use of overloading of variable names, meaning using the same variable name at the step level that is also in use at the playbook level. This is to avoid issues with implementations. However, if the variable name is reused at the step level, then only that instance of the variable **SHOULD** be updated. ~~dictionary.~~ See section ~~9.16~~10.18 for more information about variables.

Example 4.3

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

```
{
  "workflow": {
    ...,
    "action--ba23c1b3-fdd2-4264-bc5b-c056c6862ba2": {
      "type": "action",
      "commands": [
        {
          "type": "manual",
          "command": "Disconnect the machines from the network",
        }
      ],
      "agent": "individual--328a89ab-3b8f-40c4-a491-24a40bcd3cd4",
      "delay": 5000,
      "timeout": 60000,
      "on_success": "action--e5f0d00d-3047-4432-8964-cc0c56e5ab12",
      "on_failure": "action--e7e93280-ab26-45aa-baa8-423c44f917a6"
    },
    ...
  }
}
```

This workflow step has a user block a machine (192.168.0.100) at the firewall using the firewall's HTTP API interface.

```
{
  "workflow": {
    ...,
    "action--ba23c1b3-fdd2-4264-bc5b-c056c6862ba2": {
      "type": "action",
      "commands": [
        {
          "type": "http-api",
          "command": "hxxps://www[.]firewall-example[.]com/v1/blockSystem?id=192.168.0.100",
        }
      ],
      "agent": "individual--328a89ab-3b8f-40c4-a491-24a40bcd3cd4",
      "delay": 5000,
      "timeout": 60000,
      "on_completion": "action--e5f0d00d-3047-4432-8964-cc0c56e5ab12"
    },
    ...
  }
}
```

4.6 Playbook Action Step

The playbook action step workflow step executes a referenced playbook using the agents and targets defined in the referenced playbook. In addition to the inherited properties, this section defines four more specific properties that are valid for this type.

Property Name	Data Type	Details
type (required)	`string`	The value of this property **MUST** be `playbook-action`.
playbook_id (required)	`identifier`	The referenced playbook to execute. The playbook ID **SHOULD** be defined such that it is locally relevant to the system that will execute the playbook.
playbook_version (optional)	`timestamp`	The version of the CACAO playbook that this step references. The value of this property **MUST** be the **modified** `timestamp` from the CACAO playbook that this step references. If this property is not defined then the latest version that is known or available is considered valid.
in_args (optional)	`list` of `string`	The list of variable names from the local **step_variables** dictionary (see section 4.1) or passed into this step from the global **playbook_variables** dictionary (see section 3.1) that are used in this playbook. See section ~~9.16~~10.18 for more information about variables.
out_args (optional)	`list` of `string`	The ~~optional~~ list of variable names from the local **step_variables** dictionary (see section 4.1) or global **playbook_variables** dictionary (see section 3.1) that are to be returned from this playbook after execution and stored in the same variable dictionary. See section ~~9.16~~10.18 for more information about variables.

Example 4.4

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

```
{
  "workflow": {
    ...,
    "playbook-action--ba23c1b3-fdd2-4264-bc5b-c056c6862ba2": {
      "type": "playbook-action",
      "playbook_id": "playbook--b71ac91e-fa00-40e8-bd1b-d863c89e6b6b",
      "playbook_version": "2023-01-11T05:11:36.152Z",
      "delay": 5000,
      "timeout": 60000,
      "on_completion": "action--c28b1bfa-5d0e-45ac-98b7-342a7fd8ae7d",
      "in_args": [ __vuln_sys_id_1__, __vuln_sys_id_2__ ],
      "out_args": [ __result_1__, __result_2__ ]
    },
    ...
  }
}
```

4.7 Parallel Step

The parallel step workflow step defines how to create steps that are processed in parallel. This workflow step 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 a response to a cyber attack on an operational technology (OT) environment that requires releasing air / steam / water pressure simultaneously. In addition to the inherited properties, this section defines one additional specific property that is valid for this type. Implementations **MUST** wait for all steps referenced in the **`next_steps`** property to complete before moving on.

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

Property Name	Data Type	Details
`type` (required)	`string`	The value of this property **MUST** be `parallel`.
`next_steps` (required)	`list` of `identifier`	A list of two or more workflow steps to be processed in parallel. The **`next_steps`** **MUST** contain at least two values. If there is only one value, then the parallel step **MUST NOT** be used. Each entry in the **`next_steps`** property forms a branch of steps that are to be executed, even if there is only one workflow step in the branch. Each branch **MUST** reference a unique end step when that branch has completed processing. This allows implementations to know when to return to the original parallel step that started that branch to look for any **`on_completion`**, **`on_success`**, or **`on_failure`** actions. 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. Each ID **MUST** represent a CACAO workflow step object.

Example 4.5

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

```
{
  "workflow": {
    ...,
    "parallel--cd0da652-c34e-4f8c-89a3-d90b97385988": {
      "type": "parallel",
```

```
      "name": "Parallel Step Example 1",
      "on_completion": "action--0f376d66-1241-4de3-b3c6-1b2566959bee",
      "next_steps": [
        "action--7877447e-6533-4c08-a5cc-0c4ac3258476",
        "action--58e8e332-64c8-4c60-95f9-f8760724e853"
      ]
    },
    "action--7877447e-6533-4c08-a5cc-0c4ac3258476": {
      "type": "action",
      "on_completion": "end--a854cc82-bb3b-4aca-904c-4e8895c37838"
    },
    "end--a854cc82-bb3b-4aca-904c-4e8895c37838": {
      "type": "end",
      "name": "End of this branch in this example"
    },
    "action--58e8e332-64c8-4c60-95f9-f8760724e853": {
      "type": "action",
      "on_completion": "end--4a10a184-5b91-4645-8bd2-34e1cf3f0e44"
    },
    "end--4a10a184-5b91-4645-8bd2-34e1cf3f0e44": {
      "type": "end",
      "name": "End of this other branch in this example"
    },
    "action--0f376d66-1241-4de3-b3c6-1b2566959bee": {
      "type": "action",
      "on_completion": "end--0350e827-25f2-40f6-b5dc-97da152c04d8"
    },
    "end--0350e827-25f2-40f6-b5dc-97da152c04d8": {
      "type": "end",
      "name": "End of the workflow"
    }
  }
}
```

4.8 If Condition Step

The if condition step workflow step 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	Data Type	Details
type (required)	`string`	The value of this property **MUST** be `if-condition`.
condition (required)	`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 (required)	`identifier`	The step ID to be processed if the condition evaluates as true. The entry in the **on_true** property forms a branch of steps that are to be executed, even if there is only one workflow step in the branch. This branch **MUST** reference a unique end step when that branch has completed processing.

		This allows implementations to know when to return to the original if condition step that started that branch to look for any **`on_completion`**, **`on_success`**, or **`on_failure`** actions. The ID **MUST** represent a CACAO workflow step object.
`on_false` (optional)	`identifier`	The step ID to be processed if the condition evaluates as false. The entry in the **`on_false`** property forms a branch of steps that are to be executed, even if there is only one workflow step in the branch. This branch **MUST** reference a unique end step when that branch has completed processing. This allows implementations to know when to return to the original if condition step that started that branch to look for any **`on_completion`**, **`on_success`**, or **`on_failure`** actions. The ID **MUST** represent a CACAO workflow step object.

Example 4.6

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

```
{
  "workflow": {
    ...,
    "if-condition--f00f0acb-a2ad-4832-b7c2-27d12506c6d7": {
      "type": "if-condition",
      "delay": "5000",
      "timeout": "60000",
      "on_completion": "action--9257d475-f046-48c6-85eb-b045a5f45a51",
      "condition": "__variable__:value == '10.0.0.0/8'",
      "on_true": "action--7feadae2-2863-442b-a366-e5f2fd3c9b3b",
      "on_false": "action--5c31ba4d-f1d8-4bf7-a118-1a9c43cc7650"
    },
    "action--7feadae2-2863-442b-a366-e5f2fd3c9b3b": {
      "type": "action",
      "on_completion": "end--fdea7439-d738-4cad-a4a1-dd1a55560870"
    },
    "end--fdea7439-d738-4cad-a4a1-dd1a55560870": {
      "type": "end",
      "name": "End of true branch for if-condition--f00f0acb-a2ad-4832-b7c2-27d12506c6d7"
    },
    "action--5c31ba4d-f1d8-4bf7-a118-1a9c43cc7650": {
      "type": "action",
      "on_completion": "end--6c48a6a3-0ab5-47ec-aa04-b055cdd8a77c"
    },
    "end--6c48a6a3-0ab5-47ec-aa04-b055cdd8a77c": {
      "type": "end",
      "name": "End of false branch for if-condition--f00f0acb-a2ad-4832-b7c2-27d12506c6d7"
    },
    "action--9257d475-f046-48c6-85eb-b045a5f45a51: {
       "type: "action",
       "name": "step after if-condition--f00f0acb-a2ad-4832-b7c2-27d12506c6d7"
    },
    ...
  }
}
```

Example when on_false, which is optional, is not present

```
{
  "workflow": {
    ...,
    "if-condition--9c453ae1-cb0c-48c4-a60e-5c591fa99376": {
      "type": "if-condition",
      "delay": "5000",
      "timeout": "60000",
      "condition": "__variable__:value == '10.0.0.0/8'",
      "on_completion": "action--71db9284-affe-4010-9ac7-b520955bce86"
      "on_true": "action--de796a9c-e24e-48db-83ce-15b6c12fc904",
    },
    "action--de796a9c-e24e-48db-83ce-15b6c12fc904": {
      "type": "action",
      "on_completion": "end--72e5e401-4b89-4a0e-8643-c9cf57936333"
    },
    "end--72e5e401-4b89-4a0e-8643-c9cf57936333": {
      "type": "end",
      "name": "End of true branch for if-condition--9c453ae1-cb0c-48c4-a60e-5c591fa99376"
    },
    "action--71db9284-affe-4010-9ac7-b520955bce86: {
      "type: "action",
      "name": "step after if-condition--9c453ae1-cb0c-48c4-a60e-5c591fa99376"
    },
    ...
  }
}
```

4.9 While Condition Step

The while condition step workflow step defines the 'while' conditional 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	Data Type	Details
type (required)	`string`	The value of this property **MUST** be `while-condition`.
condition (required)	`string`	A boolean expression as defined in the STIX Patterning Grammar that ***while it is true*** executes the workflow step identified by `on_true` otherwise it exits the while conditional part of the workflow step.
on_true (required)	`identifier`	The step ID to be processed every time the loop condition evaluates as true. The entry in the **on_true** property forms a branch of steps that are to be executed, even if there is only one workflow step in the branch. This branch **MUST** reference a unique end step when that branch has completed processing. This allows implementations to know when to return to the original While Step that started that branch to look for any **on_completion**, **on_success**, or **on_failure** actions.

		The ID **MUST** represent a CACAO workflow step object.

Example 4.7

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

```
{
  "workflow": {
    ...,
    "while-condition--8c0872af-a071-48af-9bf9-a8ad7e137289": {
      "type": "while-condition",
      "delay": "5000",
      "timeout": "60000",
      "on_completion": "action--a30d41f9-2fa4-4315-aa19-e58f0b37bb3b",
      "condition": "__variable__:value == '10.0.0.0/8'",
      "on_true": "action--75c25526-f00e-4cec-9fcc-ea1c996ef384"
    },
    ...
  }
}
```

4.10 Switch Condition Step

The switch condition step workflow step 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	Data Type	Details
type (required)	`string`	The value of this property **MUST** be `switch-condition`.
switch (required)	`string`	A variable that is evaluated to determine which key in the cases dictionary is matched against to execute the associated step.
cases (required)	`dictionary`	This property is a dictionary that defines one or more case values (as dictionary keys) and a step ID (as a key value) to be processed when the case value is matched against the switch value. The value for each entry in the dictionary **MUST** be an `identifier` and it **MUST** represent a CACAO workflow step object. This value uniquely identifies the steps to be processed when that key/value is chosen (see section ~~9.9~~10.10 for more information on identifiers). Each entry in the **cases** property forms a branch of steps that are to be executed, even if there is only one workflow step in the branch. Each branch **MUST** reference a unique end step when that branch has completed processing. This allows implementations to know when to return to the original switch condition step that started that branch to

		look for any **on_completion**, **on_success**, or **on_failure** actions. This dictionary **MAY** have a "default" case value.

Example 4.8

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

```
{
  "type": "playbook",
  "spec_version": "cacao-2.1",
  ...,
  "playbook_variables": {
    "  userseln  ": {
      "type": "integer",
      "description": "User supplied value for switch action"
    }
  },
  ...,
  "workflow": {
    ...,
    "action--fc7bda8b-c2d8-4533-b022-19a68e150e44" : {
      "type": "action",
      "name": "Ask the user to choose the next step via cli command"
      "commands": [
        {
          "type": "bash",
          "command": "/opt/bin/ask cli question(\"Enter next step choice (1, 2, other) ?\",
\"  userseln  \")",
        }
      ],
      "on_completion": "switch-condition--33ba061e-193d-41db-b40b-0e8373997dc9",
      "out_args": [ "  userseln  " ]
    },
    "switch-condition--33ba061e-193d-41db-b40b-0e8373997dc9": {
      "type": "switch-condition",
      "delay": "5000",
      "timeout": "60000",
      "on_completion": "action--c6728da5-f96a-4ba8-a4eb-fda6f24c9d7f",
      "switch": "__variableuserseln__:value",
      "cases": {
        "1": "action--fc7bda8b-c2d8-4533-b022-19a68e150e5b",
        "2": "action--c4f77ed2-b82d-4285-ae04-541366b1cfc8",
        "default": "action--cf5f14bd-c84f-4016-89ce-f1539a34c76d"
      }
    },
    "action--fc7bda8b-c2d8-4533-b022-19a68e150e5b" : {
      "type": "action",
      "name": "Action to handle Switch Option 1"
      "on_completion": "action--c6728da5-f96a-4ba8-a4eb-fda6f24c9d7f"
    },
    "action--c4f77ed2-b82d-4285-ae04-541366b1cfc8" : {
      "type": "action",
      "name": "Action to handle Switch Option 2"
      "on_completion": "action--c6728da5-f96a-4ba8-a4eb-fda6f24c9d7f"
    },
    "action--cf5f14bd-c84f-4016-89ce-f1539a34c76d" : {
      "type": "action",
      "name": "Action to handle Switch Option Default"
      "on_completion": "action--c6728da5-f96a-4ba8-a4eb-fda6f24c9d7f"
    },
```

```
    "end--c6728da5-f96a-4ba8-a4eb-fda6f24c9d7f" : {
      "type": "end",
      "name": "Next step upon completion of the switch statement"
    },
    ...
  }
}
```

5 Commands

The CACAO command object (`command-data`) contains detailed information about the commands that are to be executed or processed automatically or manually as part of an action step (see section 4.5). Each command listed in an action 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 agents defined in that step.

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

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

5.1 Command Data

Property Name	Data Type	Details
`type` (required)	`open-vocab`	The type of command being used. The value of this property **SHOULD** come from the `command-type-ov` vocabulary.
`description` (optional)	`string`	An optional description about this command.
`command` (optional)	`string`	A string-based command as defined by the type. Commands can be simple strings, JSON blobs (normal plain/text JSON data), or type specific command IDs (see example 5.5). The command **MUST** be valid for the defined type and version.
`command_b64` (optional)	`string`	A base64 encoded (see section 4 of [RFC 4648]) 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` (optional)	`string`	The version of the command language being used. If no version is specified then the most current version of the command language **SHOULD** be used.
`playbook_activity` (optional)	`open-vocab`	A meta data description of the playbook activity that the command provides that enables summarization at the playbook level of all activities defined within the playbook. This property **SHOULD** be populated. The value for this property **SHOULD** come from the `playbook-activity-type-ov` vocabulary.

5.2 Command Type Vocabulary

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.
`bash`	A Bash command. Bash is just a shell without a login/remote connection. Therefore, if someone wants to define shell commands as part of a playbook but the login process is performed using some other mechanism then bash would be the way to do this. An example might be using a remote terminal emulator ~~(RDP)~~ that supports login to a shell and then the commands are defined in bash.
`http-api`	An HTTP API command.
`ssh`	An SSH command. SSH combines both the login aspect (i.e. the agent part) and the command part (the shell commands to execute via ssh) once logged in. It would be unnatural to define the agent as ssh and then define the shell/command as bash. One would SSH to a agent and then execute shell commands in SSH or they would remote login to a agent (ie. telnet or otherwise) and then execute shell commands in bash.
`caldera-cmd`	A caldera command that is used by an attack orchestration system to attack or simulate an attack against an agent. These can include attacks from vulnerability assessment or penetration systems. An example would be a Caldera ability included in the command when the agent specifies a Caldera agent or group. (See [CalderaAbility]). If the **`command`** property is used, then that property **SHOULD** only contain the ID of the Caldera ability that is to be executed. However, if the entire Caldera ability is to be shared, implementers **SHOULD** use the **`command_b64`** property to contain the base64 encoded version of the ability itself.
`elastic`	An Elasticsearch query, such as Query DSL (Domain Specific Language) [QDSL: https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl.html] or EQL (Event Query Language) [EQL: https://www.elastic.co/guide/en/elasticsearch/reference/current/eql.html].
`jupyter`	A Jupyter notebook (expressed in JSON).
`kestrel`	A Kestrel command (expressed in THL) describes a threat hunt step [KESTREL]. Kestrel commands are defined in the Kestrel Threat Hunting Language (THL). THL is a whitespace-indented language and **SHOULD** use the **command_b64** property to contain the base64 encoded version of the Kestrel command or threat hunt step.

`openc2-json`	An OpenC2 command (expressed in JSON).
`sigma`	A Sigma rule (expressed in YAML) [SIGMA]. Sigma rules are defined in YAML. YAML is a whitespace-indented language, meaning that indentation is used to denote structure. Due to this requirement, implementers **SHOULD** use the **command_b64** property to contain the base64 encoded version of the Sigma rule.
`yara`	A YARA rule [YARA: https://yara.readthedocs.io/en/stable/writingrules.html].

Example 5.1 (HTTP API Command)

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

Example 5.2 (Manual Command)

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

Example 5.3 (SSH Command)

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

Example 5.4 (Attack Command Caldera Ability as specified in the command_b64 property)

```
{
  "type": "attack-cmd",
  "description": "A Caldera Ability that scans wifi",
  "command_b64":
"aWQ6IDlhMzA3NDBkLTNhYTgtNGMyMy04ZWZhLWQ1MTIxNWU4YTViOQ0KbmFtZTogU2NhbiBXSUZJIG5ldHdvcmtzDQpkZ
XNjcmlwdGlvbjogVmlldyBhbGwgcG90ZW50aWFsIFdJRkkgbmV0d29ya3Mgb24gaG9zdA0KdGFjdGljOiBkaXNjb3Zlcnk
NCnRlY2huaXF1ZToNCiAgYXR0YWNrX2lkOiBUMTAxNg0KICBuYW1lOiBTeXN0ZW0gTmV0d29rayBDb25maWd1cmF0aW9uI
ERpc2NvdmVyeQ0KcGxhdGZvcm1zOg0KICBkYXJ3aW46DQogICAgc2g6DQogICAgICBjb21tYW5kOiB8DQogICAgICAgIC4
vd2lmaS5zaCBzY2FuDQogICAgICBwYXlsb2FkOiB3aWZpLnNoDQogIGxpbnV4Og0KICAgIHNoOg0KICAgICAgY29tbWFuZ
DogfA0KICAgICAgICAuL3dpZmkuc2ggc2Nhbg0KICAgICAgcGF5bG9hZDogd2lmaS5zaA0KICB3aW5kb3dzOg0KICAgIHB
zaCxwd3NoOg0KICAgICAgY29tbWFuZDogfA0KICAgICAgICAuXHdpZmkucHMxIC1TY2FuDQogICAgICBwYXlsb2FkOiB3a
WZpLnBzMQ=="
}
```

The content of the above base64 command (**`command_b64`**) is the encoded version of the caldera ability that is shown below (decoded version). The content of the command is shown as text for illustration purposes only.

```
id: 9a30740d-3aa8-4c23-8efa-d51215e8a5b9
name: Scan WIFI networks
description: View all potential WIFI networks on host
tactic: discovery
technique:
  attack_id: T1016
  name: System Network Configuration Discovery
platforms:
```



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

Example 5.5 (Attack Command Caldera Ability as specified in the command property)

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

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


66 Authentication Information

In a CACAO playbook, authentication information is used by agents and targets when they need to authenticate against a resource. Authentication information is stored in a dictionary where the ID is the key and the value is an `authentication-info` object (see section 10.2). Common properties for an authentication information object are defined in section 6.1.

Authentication information can use and refer to variables just like other parts of the playbook. For any authentication information property value, the producer may define a variable substitution such that the actual property value is determined at runtime based on the variable assigned to the authentication information. In Example 6.1, authentication information is referenced within an agent, but the authentication information's actual values are based on variables instead of being hard-coded in the data itself. It is generally best practice to always use variables for authentication information.

Example 6.1

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

```
"authentication_info_definitions": {
  "http-basic--8899e52a-6c7b-4a10-b197-d1efb57e075b": {
    "type": "http-basic",
    "user_id": "__username__:value",
    "password": "__password__:value",
  }
}
```

6.1 Authentication Information Common Properties

Each authentication information object contains some base properties that are common across all objects. These properties are defined in the following table. The ID for each object is stored as the key in the **`authentication_info_definitions`** dictionary at the Playbook level.

Property Name	Data Type	Details
`type` (required)	`open-vocab`	The type of object being used. The value of this property **SHOULD** come from the `authentication-info-type-ov` vocabulary.
`name` (optional)	`string`	The name that represents this object that is meant to be displayed in a user interface or captured in a log message. This property **SHOULD** be populated
`description` (optional)	`string`	More details, context, and possibly an explanation about this object.
`authentication_info_extensions` (optional)	`dictionary`	This property declares the extensions that are in use on this authentication information object and contains any of the properties and values that are to be used by that extension. The key for each entry in the dictionary **MUST** be an

		`identifier` (see section 9.9 for more information on identifiers) that uniquely identifies the extension. The value for each key is a JSON object that contains the structure as defined in the extension definition's schema property. The actual step extension definition is located in the **`extension_definitions`** property found at the Playbook level.

6.2 Authentication Information Type Vocabulary

Vocabulary Name: `authentication-info-type-ov`

This section contains the types of authentication information defined in this specification. Each object is further defined in the following sections.

Type	Description
`http-basic`	Describes that the authentication data is for an HTTP basic authentication. This refers to the HTTP basic authentication scheme as defined in [RFC7617].
`oauth2`	Describes that the authentication data refers to the OAuth 2.0 specification as defined in [RFC5849] and [RFC6750].
`private-key`	Describes that the authentication data uses a username and private key.
`user-auth`	Describes that the authentication data uses a generic username and password and can be used for many different types of logins, such as system or console level logins.

6.3 HTTP Basic Authentication

This type defines the HTTP basic authentication information object and is used by agents and targets when performing HTTP basic authentication. This object inherits all of the authentication information common properties (see section 6.1). In addition to the inherited properties, this section defines four additional specific properties that are valid for this type. If the **`kms`** and **`kms_key_identifier`** properties are used the **`user_id`** and **`password`** properties **MUST NOT** be populated.

Property Name	Data Type	Details
`type` (required)	`string`	The value of this property **MUST** be `http-basic`.
`user_id` (optional)	`string`	The **`user_id`** property used in HTTP Basic authentication as defined by [RFC7617].

password (optional)	string	The password property used in HTTP Basic authentication as defined by [RFC7617]. This value **SHOULD** be passed in via a variable.
kms (optional)	boolean	If this property is **true** the key material associated with this authentication information is stored in a key management store and the **kms_key_identifier** property contains the identifier used to look up the key material associated with the **kms_key_identifier**. If this property is set to true then the value of the **kms_key_identifier** property **MUST** be defined and the value of the **password** property **MUST** be ignored
kms_key_identifier (optional)	string	This property contains the key identifier for the authentication information value stored in a key management service (KMS) used by the orchestration system executing the playbook. The specific KMS used by the orchestration system is out of scope of this specification. If this property is defined then the **kms** property **MUST** be set to true.

6.4 OAuth2 Authentication

This type defines the Oauth2 authentication information object and is used by agents and targets when performing oauth2 authentication. This object inherits all of the authentication information common properties (see section 6.1). In addition to the inherited properties, this section defines four additional specific properties that are valid for this type. If the **kms** and **kms_key_identifier** properties are used the **token** property **MUST NOT** be populated.

Property Name	Data Type	Details
type (required)	string	The value of this property **MUST** be oauth2.
oauth_header (required)	string	The OAuth header used in OAuth authentication as defined in section 3.5.1 of [RFC5849] and [RFC6750].
token (required)	string	The bearer token used in HTTP Bearer Token authentication as defined by [RFC6749] and [RFC6750]. This value **SHOULD** be passed in via a variable.
kms (optional)	boolean	If this property is **true** the key material associated with this authentication information is stored in a key management store and the **kms_key_identifier** property contains the

		identifier used to look up the key material associated with the **kms_key_identifier**. If this property is set to `true` then the value of the **kms_key_identifier** property **MUST** be defined and the value of the **token** property **MUST** be ignored
kms_key_identifier (optional)	`string`	This property contains the key identifier for the authentication information value stored in a key management service (KMS) used by the orchestration system executing the playbook. The specific KMS used by the orchestration system is out of scope of this specification. If this property is defined then the **kms** property **MUST** be set to `true`.

6.5 Private Key Authentication

This type defines the private-key authentication information object and is used by agents and targets when performing cryptographic authentication. This object inherits all of the authentication information common properties (see section 6.1). In addition to the inherited properties, this section defines four additional specific properties that are valid for this type. If the **kms** and **kms_key_identifier** properties are used the **username** and **private_key** properties **MUST NOT** be populated.

Property Name	Data Type	Details
type (required)	`string`	The value of this property **MUST** be `private-key`.
username (optional)	`string`	The username to access this agent.
private_key (optional)	`string`	The private key associated with the username to access this agent. This value **SHOULD** be passed in via a variable.
kms (optional)	`boolean`	If this property is **true** the key material associated with this authentication information is stored in a key management store and the **kms_key_identifier** property contains the identifier used to look up the key material associated with the **kms_key_identifier**. If this property is set to `true` then the value of the **kms_key_identifier** property **MUST** be defined and the value of the **private_key** property **MUST** be ignored.

kms_key_identifier (optional)	string	This property contains the key identifier for the authentication information value stored in a key management service (KMS) used by the orchestration system executing the playbook. The specific KMS used by the orchestration system is out of scope of this specification. If this property is defined then the **kms** property **MUST** be set to true.

6.6 User Authentication

This type defines a generic username / password authentication information object and is used by agents and targets when performing authentication. This object inherits all of the authentication information common properties (see section 6.1). In addition to the inherited properties, this section defines four additional specific properties that are valid for this type. If the **kms** and **kms_key_identifier** properties are used the **username** and **password** properties **MUST NOT** be populated.

Property Name	Data Type	Details
type (required)	string	The value of this property **MUST** be user-auth.
username (optional)	string	A username for this authentication.
password (optional)	string	The password for this authentication. This value **SHOULD** be passed in via a variable.
kms (optional)	boolean	If this property is **true** the key material associated with this authentication information is stored in a key management store and the **kms_key_identifier** property contains the identifier used to look up the key material associated with the **kms_key_identifier**. If this property is set to true then the value of the **kms_key_identifier** property **MUST** be defined and the value of the **password** property **MUST** be ignored.
kms_key_identifier (optional)	string	This property contains the key identifier for the authentication information value stored in a key management service (KMS) used by the orchestration system executing the playbook. The specific KMS used by the orchestration system is out of scope of this specification.

		If this property is defined then the **`kms`** property **MUST** be set to `true`.

7 Agents and Targets

In a CACAO playbook, agents are the entities that execute commands (see section 5) on or against targets. Agents are stored in a dictionary where the ID is the key and the value is an `agent-target` object (see section ~~9~~10.1). Targets are stored in a dictionary where the ID is the key and the value is an `agent-target` object (see section ~~9~~10.1). Common properties for agents and targets are defined in section ~~6~~7.1.

Agents can involve either manual or automated processing. For example, an individual may process a command manually, while a firewall may process a command automatically. An agent type vocabulary is defined in section ~~6~~7.2, and each agent type is further defined in the rest of the sections. Types include security infrastructure such as firewalls, routers, and threat intelligence platforms, as well as specific network addressable agents like URLs and IPv4/IPv6/MAC addresses.

Agents can use and refer to variables just like other parts of the playbook. For any agent property value, the producer may define a variable substitution such that the actual property value is determined at runtime based on the variable assigned to the agent. In Example ~~6~~7.1, an agent is referenced within a workflow step, but the agent's actual values are based on variables (e.g., name, email, phone, location) instead of being hard-coded by the agent itself.

Example 7.1

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

~~**Example 6.1**~~

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

```
"agent_definitions": {
  "individual--4486f3d9-e5c3-43e8-9019-ae48899679a4": {
    "type": "individual",
    "name": "John Doe",
    "name": "__INDIVIDUALS_NAME__:value",
    "contact": {
      "email": {
        "work": "__INDIVIDUALS_EMAIL__:value"
      },
      "phone": {
        "work": "__INDIVIDUALS_PHONE__:value"
      }
    }
  }
}
```

~~6~~7.1 Agent and Target Common Properties

Each object (agent or target) contains some base properties that are common across all objects. These properties are defined in the following table. The ID for each object is stored as the key in the **`agent_definitions`** dictionary or the **`target_definitions`** dictionary.

Property Name	Data Type	Details

type (required)	`open-vocab`	The type of object being used. The value of this property **SHOULD** come from the `agent-target-type-ov` vocabulary.
name (required)	`string`	The name that represents this object that is meant to be displayed in a user interface or captured in a log message. This property **MUST** be populated
description (optional)	`string`	More details, context, and possibly an explanation about this object. This property **SHOULD** be populated.
location (optional)	`civic-location`	Physical address information for this object.
agent_target_extensions (optional)	`dictionary`	This property declares the extensions that are in use on this action or target and contains any of the properties and values that are to be used by that extension. The key for each entry in the dictionary **MUST** be an `identifier` (see section ~~9.9~~10.10 for more information on identifiers) that uniquely identifies the extension. The value for each key is a JSON object that contains the structure as defined in the extension definition's schema property. The actual step extension definition is located in the **extension_definitions** property found at the Playbook level.

~~6~~7.2 Agent-Target Type Vocabulary

Vocabulary Name: `agent-target-type-ov`

This section contains the types of agents and targets defined in this specification. Each object listed below can be either an agent or a target and each object is further defined in the following sections. From the description in section ~~6~~7, agents are the entities that execute commands on or against targets. Each example in the following sections show these types as agents, but they could also be targets.

Type	Description
People and Places	
`group`	A group typically associated with a team or organizational group
`individual`	An individual human-being
`location`	An identified location (either physical or logical)
`organization`	A named organization or business entity
`sector`	A business or government sector, includes industrial categories

Devices and Equipment	
`http-api`	An HTTP API interface.
`linux`	A generic Linux system
`net-address`	A general identified network addressable entity
`security-category`	A named security infrastructure category such as Firewall, IPS, TIP, etc
`ssh`	An SSH service running on some device

~~6~~7.3 Group

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

Property Name	Data Type	Details
`type` (required)	`string`	The value of this property **MUST** be `group`.
`contact` (optional)	`contact`	Contact information for this agent.

Example ~~6~~7.2 (Group Agent)

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

```
"agent_~~defintions~~definitions": {
  "group--edec4af0-0657-4f76-9f5b-b2b980e5698b": {
    "type": "group",
    "name": "SOC",
    "description": "The SOC for company example dot com",
    "location": {
      "name": "ACME Company HQ"
    },
    "contact": {
      "email": {
        "work": "soc@example.com"
      }
    }
  }
}
```

~~6~~7.4 Individual

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

Property Name	Data Type	Details
type (required)	`string`	The value of this property **MUST** be `individual`.
contact (optional)	`contact`	Contact information for this agent.

Example ~~6~~7.3 (Individual Agent)

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

```
"agent_definitions": {
  "individual--8380aa57-6920-4beb-8abf-a32eea918823": {
    "type": "individual",
    "name": "John Doe",
    "description": "John Does is the CISO of Example dot com",
    "location": {
      "name": "ACME Company",
      "description": "ACME campus located next to airport",
      "building_details": "Building H, First floor, Suite 110, Room 110-4",
      "region": "northern-america",
      "country": "US",
      "administrative_area": "California",
      "city": "San Francisco",
      "street_address": "123 Main Street",
      "postal_code": "90123"
    },
    "contact": {
      "email": {
        "work": "doej@example.com",
        "home": "john@examplehome.com"
      },
      "phone": {
        "work": "+1-123-123-1234"
      },
      "contact_details": "John works remotely most days"
    }
  }
}
```

~~6~~7.5 Location

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

Property Name	Data Type	Details
type (required)	`string`	The value of this property **MUST** be `location`.
logical (optional)	`list` of `string`	An optional list of logical location names as defined by the playbook creator (e.g., wiring closet, network segment, room number, etc)

Example ~~6~~7.4 (Location Agent)

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

```
"agent_definitions": {
```

```
  "location--9b8e6304-670c-4e87-84ce-b37a9afe887c": {
    "type": "location",
    "name": "Example.com HQ",
    "description": "The HQ site for company example dot com",
    "location": {
      "name": "ACME Company",
      "description": "ACME campus located next to airport",
      "building_details": "Building H, First floor, Suite 110, Room 110-4",
      "network_details": "Network closet 201, rack location:38U",
      "region": "northern-america",
      "country": "US",
      "administrative_area": "California",
      "city": "San Francisco",
      "street_address": "123 Main Street",
      "postal_code": "90123"
    },
  "logical": [ "room 213" ]
  }
}
```

~~6~~7.6 Organization

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

Property Name	Data Type	Details
type (required)	`string`	The value of this property **MUST** be `organization`.
contact (optional)	`contact`	Contact information for this agent.

Example ~~6~~7.5 (Organization Agent)

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

```
"agent_definitions": {
  "organization--0416efe1-6427-4153-844e-8a1b08921ec6": {
    "type": "organization",
    "name": "ACME Company",
    "location": {
      "name": "ACME Company",
      "description": "ACME campus located next to airport",
      "building_details": "Building H, First floor, Suite 110, Room 110-4",
      "network_details": "Network closet 201, rack location:38U",
      "region": "northern-america",
      "country": "US",
      "administrative_area": "California",
      "city": "San Francisco",
      "street_address": "123 Main Street",
      "postal_code": "90123"
    },
    "contact": {
      "email": {
        "work": "doej@example.com",
        "home": "john@examplehome.com"
      },
      "phone": {
```

```
        "work": "+1-123-123-1234"
      },
      "contact_details": "John works remotely most days"
    }
  }
}
```

67.7 Sector

This type defines a sector object and is used for commands that need to be processed or executed by a sector. This object inherits the common agent properties. In addition to the inherited properties, this section defines one additional specific property that is valid for this type.

Property Name	Data Type	Details
type (required)	`string`	The value of this property **MUST** be `sector`.
sector (required)	`string`	The values this property **SHOULD** come from the `industry-sector-ov` vocabulary, see section 67.7.1.

Example 67.6 (Sector Agent)

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

```
"agent_definitions": {
  "sector--5aa10ecd-c367-4157-82b1-2b4891d4ae3e": {
    "type": "sector",
    "name": "Healthcare Sector",
    "sector": "healthcare"
  }
}
```

67.7.1 Industry Sector Vocabulary

Vocabulary Name: `industry-sector-ov`

This section defines the industrial and commercial sectors sectors

Sector Type	Description
`aerospace`	The aerospace sector/industry comprises entities that research, design, manufacture, operate, and maintain aircraft and spacecraft technology. Aerospace activity is very diverse, with a plethora of commercial, industrial, and military applications. Subclasses (sector/industry): `aviation`
`aviation`	The aviation sector/industry is similar to the aerospace sector/industry, but its applications are focused within the earth's atmosphere.

agriculture	The agriculture sector/industry comprises entities primarily engaged in farming animals (animal husbandry) and plants (agronomy, horticulture, and forestry in part).
automotive	The automotive sector/industry comprises entities involved in the manufacture of motor vehicles, including most components, such as engines and bodies, but excluding tires, batteries, and fuel [VocabAuto].
biotechnology	The biotechnology sector/industry comprises entities involved in utilizing biotechnology to develop products. Biotechnology is known to highly overlap with the pharmaceutical sector/industry, but generally, it has a plethora of use cases like in agriculture, food, and chemical sectors/industries.
chemical	The chemical sector/industry comprises entities producing petrochemicals, polymers, basic inorganics, specialties, and consumer chemicals [VocabChem]. The products (chemicals) produced by the chemical sector/industry have a broad range of uses, such as in the food industry, pharmaceutical, agriculture, manufacturing, and industries involved in consumer goods.
commercial	The commercial sector/industry comprises entities involved in wholesale and retail trade, generally without making any changes to the goods. The commercial sector, in this case, does not include professional services.
consulting	The consulting sector/industry comprises entities that provide expert advice and possibly implementation services in exchange for a fee.
construction	The construction sector/industry comprises entities involved in building construction (residential and non-residential) , infrastructure construction (e.g., large public works, dams, bridges, roads, airports, railways, and tramlines), and industrial construction (e.g., energy installations, manufacturing plants).
cosmetics	The cosmetics sector/industry comprises entities that manufacture and distribute cosmetic products (e.g., hygiene products such as soap, shampoo, deodorant, and toothpaste to luxury beauty items including perfumes and makeup).
critical-infrastructure	Critical infrastructure comprises sectors with assets or systems that are essential for maintaining vital societal functions. It is the case that different nations may define more or fewer assets as critical infrastructures. For example, the U.S.A defines sixteen sectors as critical infrastructures, whereas Norway defines six, namely, communication

	networks, energy, water and wastewater, transportation, oil and gas, and satellite communications.
`dams`	The dams sector/industry comprises entities involved in operating and maintaining dams. Based on its purpose, a dam can be considered critical infrastructure for a nation. Dams provide a wide range of economic, environmental, and social benefits, including hydroelectric power, river navigation, water supply, wildlife habitat, waste management, flood control, and recreation [VocabDams].
`defense`	The defense sector comprises government and commercial entities involved in research and development, design, production, delivery, and maintenance of weapons, weapon systems, subsystems, and components or parts, to meet military requirements. The defense sector covers everything from land, sea, and air defense capabilities and cybersecurity.
`education`	The education sector/industry comprises entities that facilitate learning, such as schools, colleges, and universities.
`emergency-services`	The emergency services sector/industry provides a wide range of prevention, preparedness, response, and recovery services during both day-to-day operations and incident response. The emergency services sector includes geographically distributed facilities and equipment in both paid and volunteer capacities organized primarily at the federal, state, local, tribal, and territorial levels of government, such as city police departments and fire stations, county sheriff's offices, Department of Defense police and fire departments, and town public works departments. The emergency services sector also includes private sector resources, such as industrial fire departments, private security organizations, and private emergency medical services providers [VocabEmSrv].
`energy`	The energy sector/industry comprises entities involved in electricity generation and distribution, such as energy generation infrastructure (e.g., power plants), operators (e.g., grid operators), transmission and distribution lines, and utility providers. Electricity can be generated from renewable resources (e.g., hydro, wind, solar, biomass, geothermal), fossil fuels (such as coal and gas), or nuclear power. The energy sector/industry overlaps with the utilities sector/industry. Subclasses (sector/industry): `non-renewable-energy`, `renewable-energy`.

non-renewable-energy	The non-renewable energy sector/industry comprises entities involved in electricity generation and distribution from non-renewable resources such as oil and petroleum products, gasoline, natural gas, diesel fuel, and nuclear.
renewable-energy	The renewable energy sector/industry comprises entities involved in electricity generation and distribution from renewable resources such as hydro, wind, solar, biomass, and geothermal.
media	The media sector/industry consists of film, print, radio, and television, also when provided in electronic form via the Internet.
financial	The financial sector/industry is a broad term used to describe a range of activities that manage money, encompassing everything from insurance companies and stock brokerages to investment funds and banking.
food	The food sector/industry comprises entities involved in the processing, preparation, preservation, and packaging of food and beverages. The raw materials used are generally of vegetable or animal origin and produced by agriculture like farming, breeding, and fishing. Foodservice is also included and comprises entities that serve food to people, such as restaurants.
gambling	The gambling or betting sector/industry includes online or offline gambling entities like casinos and other online betting games.
government	The government sector includes a wide variety of entities involved in governmental nature activities. It includes facilities owned or leased by federal, state, local, and tribal governments. Many government facilities are open to the public for business activities, commercial transactions, or recreational activities. In contrast, others that are not open to the public contain highly sensitive information, materials, processes, and equipment. These facilities include general-use office buildings and special-use military installations, embassies, courthouses, national laboratories, and structures that may house critical equipment, systems, networks, and functions. In addition to physical structures, the sector includes cyber elements that contribute to the protection of sector assets (e.g., access control systems and closed-circuit television systems) as well as individuals who perform essential functions or possess tactical, operational, or strategic knowledge [VocabGov]. The government sector overlaps with multiple other sectors/industries like the defense sector/industry that comprises government entities

	that support a nation's national security capability, and the emergency services sector that provides a wide range of prevention, preparedness, response, and recovery services during both day-to-day operations and incident response. Subclasses (sector/industry): `local-government`, `national-government`, `regional-government`, `public-services`.
`local-government`	A city or municipality level of government. See `government` above for more details.
`national-government`	A national level of government. See `government` above for more details.
`regional-government`	A regional, state, or area level of government. See `government` above for more details.
`public-services`	The public services sector/industry includes services provided by a government to people living within its jurisdiction, either directly through public sector agencies or by financing provision of services by private businesses or voluntary organizations [VocabPServ]. The public services sector may overlap with other sectors like healthcare, education, transportation, and utilities. See `government` above for more details.
`healthcare`	The healthcare sector/industry comprises entities that provide healthcare, meaning services to assess, maintain or restore a patient's state of health, including the prescription, dispensation, and provision of medicinal products and medical devices [VocabHealth].
`information-communications-technology`	The information and communications technology (ICT) sector/industry, also known as the information technology sector or just technology sector, comprises entities that produce and provide information technology services and products such as software, hardware, electronics, and telecommunications. Subclasses (sector/industry): `electronics-hardware`, `software`, `telecommunications`
`electronics-hardware`	The electronics and hardware sector/industry comprises entities that produce electronic equipment and components and computer hardware.
`software`	The software sector/industry comprises entities dedicated to producing software.

telecommunications	The telecommunications industry within the ICT sector comprises entities that produce and provide telecommunications equipment and services. Examples are Internet Service Providers (ISPs), wired and mobile telephony providers, and satellite communications operators. A satellite research and production facility would fall optimally under the aerospace sector/industry.
legal-services	The legal services sector/industry, otherwise known as the legal industry, comprises entities that provide services of lawyers and other legal practitioners to individuals, businesses, government agencies, and nonprofits.
lodging	The lodging sector/industry is a segment of the hospitality sector/industry specializing in providing customers with accommodation services (e.g., hotels, motels, resorts, and bed and breakfasts).
manufacturing	The manufacturing sector/industry comprises entities involved in the creation of products from raw materials and commodities. It includes all foods, chemicals, textiles, machines, and equipment, it includes all refined metals and minerals derived from extracted ores, and it includes all lumber, wood, and pulp products. As a best practice, the manufacturing sector/industry element **SHOULD** be used when none of the taxonomy elements is adequate for tagging or classifying a case. In addition, like the rest of the elements, manufacturing can be used in combination with another element to provide extra precision in classification/tagging. For example, an incident that affects a company in the automotive sector/industry can be tagged as both automotive and manufacturing to indicate that the manufacturing process or a manufacturing plant was targeted.
maritime	The maritime sector/industry involves a plethora of organizations and activities related to the ocean and ships and other floating entities. Examples are maritime transportation, shipyards, maritime equipment manufacturers, and commercial fishing. The maritime sector/industry highly overlaps with the transportation sector/industry.
metals	The metals sector/industry comprises entities involved in the processing of non-ferrous metals such as aluminum, copper, zinc, and ferrous materials such as steel [VocabMetals].
mining	The mining sector/industry comprises entities dedicated to locating and extracting metal and mineral reserves. Oil and natural gas extraction are not included in this industry, but they can be referenced using the Petroleum sector/industry.

non-profit	The nonprofit sector/industry comprises entities organized and operated for a collective, public, or social benefit compared to for-profit organizations that aim to generate a profit. The nonprofit sector/industry may overlap with other sectors/industries that also accommodate nonprofit entities, like public universities that are part of the education sector/industry, but they have a nonprofit cause. Further, non-governmental organizations (NGOs) are not distinguished and thus are included in the nonprofit sector/industry. Subclasses (sector/industry): humanitarian-aid, human-rights.
humanitarian-aid	The humanitarian aid industry or humanitarian aid organizations provide material and logistic assistance to people who need help (e.g., homeless, refugees, and victims of natural disasters, wars, and famines).
human-rights	The human rights industry comprises establishments primarily engaged in promoting causes associated with human rights either for a broad or specific constituency. Establishments in this industry address issues, such as protecting and promoting the broad constitutional rights and civil liberties of individuals and those suffering from neglect, abuse, or exploitation; promoting the interests of specific groups, such as children, women, senior citizens, or persons with disabilities; improving relations between racial, ethnic, and cultural groups; and promoting voter education and registration [VocabHumanRights].
nuclear	The nuclear sector/industry includes nuclear infrastructure, and in this taxonomy, it is unrelated to how the nuclear power is used, such as for energy generation, radioactive materials for healthcare, or for developing nuclear weapons.
petroleum	The petroleum sector/industry, also known as the oil sector/industry or the oil and gas sector/industry, comprises entities involved in the global processes of exploration, extraction, refining, and transporting (often by oil tankers and pipelines) petroleum. The petroleum industry overlaps with the chemical sector/industry in the sense that many finished products derived from petrochemical processing.
pharmaceuticals	The pharmaceutical sector/industry comprises entities that research, develop, produce, and distribute pharmaceutical products like medications.
research	The research industry comprises entities that solely specialize in research like research institutions, think tanks, and research groups/divisions within organizations.

`transportation`	The transportation sector/industry comprises entities that provide services to move people or goods, including transportation infrastructure. The transportation sector consists of several industries that focus on transportation, including air freight and logistics, airlines, marine, road and rail, and transportation infrastructures like railroads and marine ports. Subclasses (sector/industry): `logistics-shipping`
`logistics-shipping`	The logistics and shipping sector/industry comprise entities responsible for planning, implementing, and controlling procedures for the efficient and effective transportation and storage of goods.
`utilities`	The utilities sector comprises entities that provide basic amenities, such as water, sewage services, electricity, dams, and natural gas [VocabUtils].
`video-game`	The video game industry comprises entities involved in the development, marketing, and monetization of video games. The video game industry overlaps with other sectors/industries like the ICT and, in particular, the software sector/industry.
`water`	The water sector/industry comprises entities that provide water and wastewater services, including sewage treatment. The water sector/industry does not include manufacturers and suppliers of bottled water, which is part of the beverage production and belongs to the food sector/industry [VocabWater].

~~6~~7.8 HTTP API

This type defines an HTTP API object and is used for commands that need to be processed or executed by an HTTP API. In addition to the inherited properties, this section defines ~~six additional specific properties that are valid for this type. In addition to the inherited properties, this section defines seven~~three additional specific properties that are valid for this type.

Property Name	Data Type	Details
`type` (required)	`string`	The value of this property **MUST** be `http-api`.
`http_url` (required)	`string`	A full URL of the HTTP API service that should be called.
~~**`http_auth_type`** (optional)~~	~~`string`~~	~~The authentication type required to access this HTTP API (e.g., "basic", "oauth2", etc.)~~

~~**user_id**~~**authentication_info** (optional)	~~string~~identifier	~~The user_id~~This property ~~used in HTTP Basic~~contains an ID reference to a CACAO `authentication`~~ as defined by [RFC7617].~~`-info` object that is stored at the Playbook level in the **`authentication_info_definitions`** property. The ID **MUST** reference a CACAO `authentication-info` object (see section 6).
~~**password** (optional)~~	~~string~~	~~The password property used in HTTP Basic authentication as defined by [RFC7617].~~
~~**token** (optional)~~	~~string~~	~~The bearer token used in HTTP Bearer Token authentication as defined by [RFC6750].~~
~~**oauth_header** (optional)~~	~~string~~	~~The OAuth header used in OAuth authentication as defined in section 3.5.1 of [RFC5849].~~
category (optional)	`list` of `open-vocab`	One or more identified categories of security infrastructure types that this agent represents (see section ~~6~~7.11.1). The value for this property **SHOULD** come from the `security-category-type-ov` vocabulary.

Example ~~6~~7.7 (HTTP-API Agent)

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

```
"agent_definitions": {
  "http-api--7e9174ec-a293-43df-a72d-471c79e276bf": {
    "type": "http-api",
    "name": "Firewall 1",
    "http_url": "hxxp://example.com/v1/",
    "authentication_info": "http~~-auth-type": "~~-basic--c2557066-1ffe-440e-b474-fea8158ab202",
~~    "user_id": "__username__:value",~~
~~    "password": "__password__:value",~~
    "category": [ "firewall" ],
    "location": { ... }
  }
}
```

~~6~~7.9 Linux System

This type defines a Linux system object and is used for commands that need to be processed or executed by a Linux system. In addition to the inherited properties, this section defines ~~five~~four additional specific properties that are valid for this type.

Property Name	Data Type	Details
type (required)	`string`	The value of this property **MUST** be `linux`.

address (required)	dictionary	The key for each entry in the dictionary **MUST** be a string that uniquely identifies one or more address types. The key(s) **MUST** be one of the following values ipv4, ipv6, l2mac, vlan, or url. The dictionary value associated with each key **MUST** be a list of string that contains the corresponding address(es) for that particular key type.
port (optional)	string	The TCP port number for the Linux system. The default value is 22 based on standard port number services [PortNumbers].
~~**username** (optional)~~	~~string~~	~~The username to access this system.~~
~~**password**~~**authentication_info** (optional)	~~string~~identifier	~~The password associated with the username to access this system. This value will most often be passed in via a variable.~~This property contains an ID reference to a CACAO authentication-info object that is stored at the playbook level in the **authentication_info_definitions** property. The ID **MUST** reference a CACAO authentication-info object (see section 6).
~~**private_key** (optional)~~	~~string~~	~~The private key associated with the username to access this system. This value will most often be passed in via a variable.~~
category (optional)	list of open-vocab	One or more identified categories of security infrastructure types that this agent represents (see section ~~6~~7.11.1). The value for this property **SHOULD** come from the security-category-type-ov vocabulary.

Example ~~6~~7.8 (Linux Agent)

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

```
"agent_definitions": {
  "linux--075c6ab4-c8ff-4fe6-8569-44894e5a52a9": {
    "type": "linux",
    "name": "Linux Server 1",
    "address": {
      "ipv4": [ "10.1.2.3" ]
    },
    "authentication_info": "user~~_id": "__username__:value",
    ~~"password": "__password__:value~~-auth--c409bc40-01df-4cb2-b11f-a7cf5358659b",
    "category": [ "kali" ],
    "location": { ... }
  }
}
```

~~6~~7.10 Network Address

This type defines a network address object and is used for commands that need to be processed or executed by a device at a network address. In addition to the inherited properties, this section defines ~~five~~three additional specific properties that are valid for this type.

Property Name	Data Type	Details
type (required)	`string`	The value of this property **MUST** be `net-address`.
address (required)	`dictionary`	The key for each entry in the dictionary **MUST** be a `string` that uniquely identifies one or more address types. The key(s) **MUST** be one of the following values `ipv4`, `ipv6`, `l2mac`, `vlan`, or `url`. The dictionary value associated with each key **MUST** be a `list` of `string` that contains the corresponding address(es) for that particular key type.
~~**username** (optional)~~	~~`string`~~	~~The username to access this agent.~~
~~**password**~~**authentication_info** (optional)	~~`string`~~`identifier`	~~The password associated with the username to access this agent. This value **SHOULD** be passed in via a variable.~~This property contains an ID reference to a CACAO `authentication-info` object that is stored at the playbook level in the **`authentication_info_definitions`** property. The ID **MUST** reference a CACAO `authentication-info` object (see section 6).
~~**private_key** (optional)~~	~~`string`~~	~~The private key associated with the username to access this agent. This value **SHOULD** be passed in via a variable.~~
category (optional)	`list` of `open-vocab`	One or more identified categories of security infrastructure types that this agent represents (see section ~~6~~7.11.1). The value for this property **SHOULD** come from the `security-category-type-ov` vocabulary.

Example ~~6~~7.9 (General Network Agent)

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

```
"agent_definitions": {
  "net-address--6f6f9814-5982-4322-9a9c-0ef25d33ef2a": {
    "type": "net-address",
    "name": "Firewall 1",
    "address": {
      "url": [ "hxxps://someorg[.]com/tellmetoorchestratewhat/amethod" ]
    },
~~    "username": "someusername",~~
~~    "password": "__password__:value",~~
    "authentication_info": "user-auth--c409bc40-01df-4cb2-b11f-a7cf5358659b",
    "category": [ "firewall" ~~],~~]
~~    "location": { ... }~~
~~  }~~
~~}~~

~~6~~ }
}
```

7.11 Security Category

This type defines a security (infrastructure) category object and is used for commands that need to be processed or executed by a piece of security infrastructure. In addition to the inherited properties, this section defines one additional specific property that is valid for this type.

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

Example ~~6~~7.10 (Security Infrastructure Agent)

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

```
"agent_definitions": {
  "security-category--f81aa730-2c59-4190-b8d5-3f2b4beecd95": {
    "type": "security-category",
    "name": "IPS 1",
    "category": [ "ips" ]
  }
}
```

~~6~~7.11.1 Security Category Type Vocabulary

Vocabulary Name: `security-category-type-ov`

This section defines the infrastructure types that an agent may relate to. These infrastructure types capture the key characteristics of the infrastructure and it includes values from the very general to the more specific. It is not intended to be exhaustive nor binary.

Infrastructure Type	Description
`aaa`	An authentication, authorization and accounting services system
`analytics`	An analytical processing system such as flow processing, anomaly detection, machine-learning, behavioral detection, etc
`caldera`	Caldera allows organizations to test endpoint security solutions and assess a network's security posture against the common post-compromise adversarial techniques contained in the MITRE ATT&CK model.
`content-gateway`	A gateway inspection and mitigation system
`desktop`	A desktop system

endpoint	A general computer device with no specific constraints or requirements
firewall	A L3, L4 or above firewall
handset	A handset device
ids	An intrusion detection system
ips	An intrusion prevention system
kali	Kali Linux is an open-source, Debian-based Linux distribution geared towards various information security tasks, such as Penetration Testing, Security Research, Computer Forensics and Reverse Engineering.
manx	The Manx plugin supplies shell access into Caldera, along with reverse-shell payloads for entering/exiting agents manually.
orchestrator	An orchestration system
os-linux	A Linux operating system
os-mac	A Mac-OS operating system
os-windows	A Windows operating system
redcanary-atomicred	Atomic Red Team is a collection of small, highly portable detection tests mapped to MITRE ATT&CK. This gives defenders a highly actionable way to immediately start testing their defenses against a broad spectrum of attacks.
ragdoll	The Ragdoll plugin gets instructions by scraping the decoy web page, it then sends results through GET URL parameters (encoded).
router	A L3 or above routering system
sandcat	The Sandcat plugin is the default agent used in a Caldera operation.
server	A generic server system common in deployments such as the cloud or services supporting multiple client devices and applications
siem	A SIEM
switch	A L2, L3, or above switching system
ticketing	A trouble-ticketing system, workload processing system, etc
tip	A threat intelligence platform
wireless	A wireless communications system typically associated with 802.11 radio communications

~~6~~7.12 SSH CLI

This type defines a SSH CLI object and is used for commands that need to be processed or executed by an SSH CLI. In addition to the inherited properties, this section defines ~~six~~four additional specific properties that are valid for this type.

Property Name	Data Type	Details
type (required)	`string`	The value of this property **MUST** be `ssh`.
address (required)	`dictionary`	The key for each entry in the dictionary **MUST** be a `string` that uniquely identifies one or more address types. The key(s) **MUST** be one of the following values `ipv4`, `ipv6`, `l2mac`, `vlan`, or `url`. The dictionary value associated with each key **MUST** be a `list` of `string` that contains the corresponding address(es) for that particular key type.
port (optional)	`string`	The TCP port number for the SSH service. The default value is 22 based on standard port number services [PortNumbers].
~~**username** (optional)~~	~~`string`~~	~~The username to access this system.~~
~~**password**~~**authentication_info** (optional)	~~`string`~~`identifier`	~~The password associated with the username to access this system. This value will most often be passed in via a variable.~~This property contains an ID reference to a CACAO `authentication-info` object that is stored at the playbook level in the **`authentication_info_definitions`** property. The ID **MUST** reference a CACAO `authentication-info` object (see section 6).
~~**private_key** (optional)~~	~~`string`~~	~~The private key associated with the username to access this system. This value will most often be passed in via a variable.~~
category (optional)	`list` of `open-vocab`	One or more identified categories of security infrastructure types that this agent represents (see section ~~6~~7.11.1). The value for this property **SHOULD** come from the `security-category-type-ov` vocabulary.

Example ~~6~~7.11 (SSH CLI Agent)

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

```
"agent_definitions": {
  "ssh--e75ad630-ac34-4e90-9dab-406c378cfb98": {
    "type": "ssh",
    "name": "SSH Server 1",
    "address": {
      "ipv4": [ "192.168.1.100" ]
    },
    "username": "someusername",
    "password": "__password__:value",
    "authentication_info": "user-auth--c409bc40-01df-4cb2-b11f-a7cf5358659b",
    "category": [ "router" ],
    "location": { ... }
```

```
    }
}
```


~~7~~8 Extension Definition

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 definition object is the value. Workflow steps, agents, 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.

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 fulfills that need.

~~7~~8.1 Extension Definition Properties

Property Name	Data Type	Details
type (required)	`string`	The value of this property **MUST** be `extension-definition`.
name (required)	`string`	A name used to identify this extension for display purposes during execution, development or troubleshooting.
description (optional)	`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 property. ~~Note that the~~NOTE: The schema property is the normative definition of the extension, and this property, if present, is for documentation purposes only.
created_by (required)	`identifier`	An ID that represents the entity that created this extension. The ID **MUST** represent a STIX 2.1+ identity object.
schema (required)	`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 (required)	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.

Example ~~7~~8.1

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

```
"extension-definition--0a727eb7-f699-4e20-a182-2db4b18b084a": {
  "type": "extension-definition",
  "name": "Extension Foo",
  "description": "This schema adds foo to bar for steps",
  "created_by": "identity--5abe695c-7bd5-4c31-8824-2528696cdbf1",
  "schema": "hxxps://www[.]example[.]com/schema-foo/v1/",
  "version": "1.2.1"
}
```

Example ~~7~~8.2

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

```
{
  "type": "playbook",
  ...
  "workflow": {
    "action--ba6c42d2-8563-4a0a-ba2c-af1764808513": {
      "type": "action",
      "delay": 5000,
      "timeout": 60000,
      "on_success": "action--91ed4639-cfa5-401d-a55a-6428afc90f98",
      "on_failure": "action--259b0dac-4b3b-4e1b-8686-950ecc7a3b17",
      "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--5abe695c-7bd5-4c31-8824-2528696cdbf1",
      "schema": "hxxps://www[.]example[.]com/schema-bar/v3/",
      "version": "3.2.3"
    }
  }
}
```

Example ~~7~~8.3

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

```
{
  "type": "playbook",
```



```
  ...
  "agent_definitions": {
    "net-address--bd546f32-637f-4b69-8161-ade5b5b53cbc": {
      "type": "net-address",
      "address": {
        "url": [ "hxxps://someorg[.]com/tellmetoorchestratewhat/amethod" ],
        "vlan": [ "vlan1" ]
      },
      "username": "someusername",
      "password": "apassword",
      "agent_extensions": {
        "extension-definition--45c72acc-d124-481e-8b12-57ab1fd4c144": {
          "l2_address": "010203040506"
        }
      }
    }
  },
  "extension_definitions": {
    "extension-definition--45c72acc-d124-481e-8b12-57ab1fd4c144": {
      "type": "extension-definition",
      "name": "Network Agent with Mac",
      "description": "This schema adds L2 mac address to network agents",
      "created_by": "identity--5abe695c-7bd5-4c31-8824-2528696cdbf1",
      "schema": "hxxps://www[.]example[.]com/schema-something/v2/",
      "version": "2.1.2"
    }
  }
}
```


~~8~~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. This specification does not define rules for how multiple markings applied to the same object should be interpreted.

~~8~~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	Data Type	Details
`type` (required)	`enum`	The type of data marking being used. The value for this property **MUST** come from the `data-marking-type-enum` enumeration.
`id` (required)	`identifier`	A value that uniquely identifies the data marking definition.
`name` (optional)	`string`	A name used to identify this data marking.
`description` (optional)	`string`	More details, context, and possibly an explanation about what this data marking does and tries to accomplish.
`created_by` (required)	`identifier`	An ID that represents the entity that created this data marking. The ID **MUST** represent a STIX 2.1+ identity object.
`created` (required)	`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`** (required)~~	~~`timestamp`~~	~~Data markings **MUST NOT** be versioned. This property **MUST** always equal the timestamp of the **`created`**~~

		~~property.~~
revoked (optional)	`boolean`	A boolean that identifies if the creator deems that this data marking is no longer valid. The default value is `false`. Processing of data that has been previously shared with an associated data marking that is subsequently revoked is unspecified and dependent on the implementation of the consuming software.
valid_from (optional)	`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**. If the **revoked** property is `true` then this property **MUST** be ignored.
valid_until (optional)	`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. If the **revoked** property is `true` then this property **MUST** be ignored.
labels (optional)	`list` of `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 (optional)	`list` of `external-reference`	An optional list of external references for this data marking.
marking_extensions (optional)	`dictionary`	This property declares the extensions that are in use on this data marking and contains any of the properties and values that are to be used by that extension. The key for each entry in the dictionary **MUST** be an `identifier` (see section ~~9.9~~10.10 for more information on identifiers) that uniquely identifies the extension. The value for each key is a JSON object that contains the structure as defined in the extension definition's schema property. The actual step extension definition is located in the **extension_definitions** property found at the Playbook level.

~~8~~9.2 Data Marking Type Enumeration

Enumeration Name: `data-marking-type-enum`

This section defines the following types of data markings.

Data Marking Type	Description
`marking-statement`	The statement marking definition defines the representation of a textual marking statement (e.g., copyright, terms of use). See section ~~8~~9.3.
`marking-tlp`	The TLP marking definition. See section ~~8~~9.4.
`marking-iep`	The IEP marking definition. See section ~~8~~9.5.

~~8~~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	Data Type	Details
type (required)	`string`	The value of this property **MUST** be `marking-statement`.
statement (required)	`string`	A statement (e.g., copyright, terms of use) applied to the content marked by this marking definition.

Example ~~8.1~~

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

9.1

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

```
"data_marking_definitions": {
  "marking-statement--6372bbfe-5aa1-4225-b866-846265cc8689": {
    "type": "marking-statement",
    "id": "marking-statement--6372bbfe-5aa1-4225-b866-846265cc8689",
    "created_by": "identity--5abe695c-7bd5-4c31-8824-2528696cdbf1",
    "created": "2020-04-01T00:00:00.000Z",
    "modified": "2020-04-01T00:00:00.000Z",
    "statement": "Copyright 2020, Example Corp"
  }
}
```

~~8~~9.4 TLP Marking

The TLP marking object defines the representation of a FIRST TLP V2 marking statement.

Property Name	Data Type	Details

type (required)	`string`	The value of this property **MUST** be `marking-tlp`.
tlpv2_level (required)	`enum`	The value of this property is the name of the TLP V2 level as defined by FIRST [TLP]. The value **MUST** be one of the following: `TLP:RED`, `TLP:AMBER`, `TLP:AMBER+STRICT`, `TLP:GREEN`, `TLP:CLEAR`

The following standard data marking definitions **MUST** be used to represent `TLP:RED`, `TLP:AMBER`, `TLP:AMBER+STRICT`, `TLP:GREEN`, ~~`TLP:CLEAR` TLP markings.~~and `TLP:CLEAR` TLP markings (NOTE: these are not examples, but the actual markings). These can also be found in the CACAO Common Objects GitHub repository (https://github.com/oasis-tcs/cacao/tree/master/Common-Objects/Data-Markings).

`TLP:CLEAR`	{
	"data_marking_definitions": {
	"marking-tlp--94868c89-83c2-464b-929b-a1a8aa3c8487": {
	"type": "marking-tlp",
	"id": "marking-tlp--94868c89-83c2-464b-929b-a1a8aa3c8487",
	"created_by": "identity--5abe695c-7bd5-4c31-8824-2528696cdbf1",
	"created": "2022-10-01T00:00:00.000Z",
	~~"modified": "2022-10-01T00:00:00.000Z",~~
	"tlpv2_level": "TLP:CLEAR"
	}
	}
	}

`TLP:GREEN`	{
	"data_marking_definitions": {
	"marking-tlp--bab4a63c-aed9-4cf5-a766-dfca5abac2bb": {
	"type": "marking-tlp",
	"id": "marking-tlp--bab4a63c-aed9-4cf5-a766-dfca5abac2bb",
	"created_by": "identity--5abe695c-7bd5-4c31-8824-2528696cdbf1",
	"created": "2022-10-01T00:00:00.000Z",
	~~"modified": "2022-10-01T00:00:00.000Z",~~
	"tlpv2_level": "TLP:GREEN"
	}
	}
	}

`TLP:AMBER`	{
	"data_marking_definitions": {
	"marking-tlp--55d920b0-5e8b-4f79-9ee9-91f868d9b421": {
	"type": "marking-tlp",
	"id": "marking-tlp--55d920b0-5e8b-4f79-9ee9-91f868d9b421~~.~~",
	"created_by": "identity--5abe695c-7bd5-4c31-8824-2528696cdbf1",
	"created": "2022-10-01T00:00:00.000Z",
	~~"modified": "2022-10-01T00:00:00.000Z",~~
	"tlpv2_level": "TLP:AMBER"
	}
	}
	}

`TLP:AMBER+STRICT`	{
	"data_marking_definitions": {
	"marking-tlp--939a9414-2ddd-4d32-a0cd-375ea402b003": {

	<pre>"type": "marking-tlp",
	"id": "marking-tlp--939a9414-2ddd-4d32-a0cd-375ea402b003",
	"created_by": "identity--5abe695c-7bd5-4c31-8824-2528696cdbf1",
	"created": "2022-10-01T00:00:00.000Z",
	~~"modified": "2022-10-01T00:00:00.000Z",~~
	"tlpv2_level": "TLP:AMBER+STRICT"
	}
	}
	}

TLP:RED	<pre>{
	"data_marking_definitions": {
	"marking-tlp--e828b379-4e03-4974-9ac4-e53a884c97c1": {
	"type": "marking-tlp",
	"id": "marking-tlp--e828b379-4e03-4974-9ac4-e53a884c97c1",
	"created_by": "identity--5abe695c-7bd5-4c31-8824-2528696cdbf1",
	"created": "2022-10-01T00:00:00.000Z",
	~~"modified": "2022-10-01T00:00:00.000Z",~~
	"tlpv2_level": "TLP:RED"
	}
	}
	}

~~8~~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	Data Type	Details
type (required)	string	The value of this property **MUST** be marking-iep.
name (required)	string	The name of the IEP policy.
~~**tlp_level** (optional)~~**description** (required)	string	See IEP Specification [IEP].
~~**description** (optional)~~**tlp** (required)	string	TLPv1. See IEP Specification [IEP].
iep_version (~~optional~~required)	~~string~~integer	See IEP Specification [IEP].
start_date (~~optional~~required)	timestamp	See IEP Specification [IEP].
end_date (~~optional~~required)	timestamp or a JSON null	See IEP Specification [IEP].
encrypt_in_transit (~~optional~~required)	string	See IEP Specification [IEP].

permitted_actions (~~optional~~required)	`string`	See IEP Specification [IEP].
affected_party_notifications (required)	`string`	See IEP Specification [IEP].
attribution (~~optional~~required)	`string`	See IEP Specification [IEP].
unmodified_resale (~~optional~~required)	`string`	See IEP Specification [IEP].
external_references (required)	`list` of `string`	See IEP Specification [IEP].

Example ~~8~~9.3

```
"data_marking_definitions": {
  "marking-iep--1eb9a8a4-e35a-4b8d-8f13-f27a4bced5aa": {
    "type": "marking-iep",
    "id": "marking-iep--1eb9a8a4-e35a-4b8d-8f13-f27a4bced5aa",
    "created_by": "identity--5abe695c-7bd5-4c31-8824-2528696cdbf1",
    "created": "2020-04-01T00:00:00.000Z",
    "modified": "2020-04-01T00:00:00.000Z",
    "name": "FIRST IEP TLP-AMBER",
    "description": "IEP policy",
    "iep_version": 2.0,
    "start_date": "2020-04-02T00:00:00.000Z",
    "end_date": null,
    "tlp_level": "TLP:AMBER"": "amber",
  }
}
```

```
    "encrypt in transit": "must",
    "permitted actions": "internally-visible-actions",
    "affected party notifications": "may",
    "attribution": "may",
    "unmodified resale": "must-not",
    "external references": ["URL1", "URL2"]
  }
}
```

10 Data Types

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

~~9~~10.1 Agent and Target

The `agent-target` data type captures detailed information about the entities or devices that accept, receive, process, or execute one or more commands as defined in a workflow step and uses the JSON object type [RFC8259] for serialization (see section ~~6~~7).

~~9.2~~10.2 Authentication Information

The `authentication-info` data type captures authentication information that is used by agents and targets when they need to authenticate against a resource (see section 6).

10.3 Boolean

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

~~9.3~~10.4 Civic Location

The `civic-location` data type captures civic location information and uses the JSON object type [RFC8259] for serialization. Implementations need to be mindful when including physical address information and GPS information into the same civic location to ensure that they reference the same actual physical location. However, in the event that the physical address information and the GPS information do not match, then the physical address information **SHOULD** be considered correct.

Property Name	Data Type	Details
`name` (optional)	`string`	A name for this location.
`description` (optional)	`string`	A detailed description about this location.
`building_details` (optional)	`string`	Additional details about the location within a building including things like floor, room, etc.
`network_details` (optional)	`string`	Additional details about this network location including things like wiring closet, rack number, rack location, and VLANs.
`region` (optional)	`enum`	The geographical region for this location. The value for this property **MUST** come from the `region-enum` enumeration (see section ~~9.3~~10.4.1).

country (optional)	`string`	The country for this location. This property **MUST** contain a valid ISO 3166-1 ALPHA-2 Code [ISO3166-1].
administrative_area (optional)	`string`	The state, province, or other sub-national administrative area for this location. This property **SHOULD** contain a valid ISO 3166-2 Code [ISO3166-2].
city (optional)	`string`	The city for this location.
street_address (optional)	`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 (optional)	`string`	The postal code for this location.
latitude (optional)	`string`	The GPS latitude of the location 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. NOTE: Some systems like Google Maps have the following rules. "Latitude ranges between -90 and 90 degrees, inclusive. Values above or below this range will be clamped to the range [-90, 90]. This means that if the value specified is less than -90, it will be set to -90. And if the value is greater than 90, it will be set to 90." [Google Maps]
longitude (optional)	`string`	The GPS longitude of the location 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. NOTE: Some systems like Google Maps have the following rules. "Longitude ranges between -180 and 180 degrees, inclusive. Values above or below this range will be wrapped so that they fall within the range. For example, a value of -190 will be converted to 170. A value of 190 will be converted to -170. This reflects the fact that longitudes wrap around the globe." [Google Maps]

precision (optional)	string	Defines the precision of the coordinates specified by the **latitude** and **longitude** properties. This is measured in meters. The actual agent 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.

Example ~~9~~10.1

```
{
  "name": "ACME Company",
  "description": "ACME campus located next to airport",
  "building_details": "Building H, First floor, Suite 110, Room 110-4",
  "network_details": "Network closet 201, rack location:38U",
  "region": "northern-america",
  "country": "US",
  "administrative_area": "California",
  "city": "San Francisco",
  "street_address": "123 Main Street",
  "postal_code": "90123"
}
```

~~9.3~~10.4.1 Region Enumeration

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

Enumeration Name: region-enum

Vocabulary Value
africa
eastern-africa
middle-africa
northern-africa
southern-africa
western-africa
americas
caribbean
central-america
latin-america-caribbean
northern-america

south-america
asia
central-asia
eastern-asia
southern-asia
south-eastern-asia
western-asia
europe
eastern-europe
northern-europe
southern-europe
western-europe
oceania
antarctica
australia-new-zealand
melanesia
micronesia
polynesia

~~9.4~~10.5 Command Data

The `command-data` data type contains detailed information about the commands that are to be executed or processed automatically or manually as part of an action step (see section 4.5) and uses the JSON object type [RFC8259] for serialization (see section 5).

~~9.5~~10.6 Contact Information

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

Property Name	Data Type	Details

email (optional)	`dictionary`	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 (optional)	`dictionary`	A phone number for this contact. The key for each entry in the dictionary **MUST** be a `string` that uniquely identifies the type (e.g., the keys could be things like "work", "home", "personal", etc). The value for each key **MUST** be a `string`.
contact_details (optional)	`string`	Additional contact information.

Example ~~9~~10.2

```
{
  "email": {
    "work": "person1@example.com",
    "home": "home@example.com"
  },
  "phone": {
    "work": "+1-123-123-1234"
  },
  "contact_details": "Some details about this contact"
}
```

~~9.6~~10.7 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 an ASCII string
- **MUST** only contain the characters: a-z (lowercase ASCII), A-Z (uppercase ASCII), 0-9, a hyphen (-) (see section 1.6), and an underscore (_)
- **MUST** be no longer than 250 ASCII characters in length and **SHOULD** be lowercase
- **MUST** start with a letter, number, or the underscore character

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

~~9.7~~10.8 Enum

The `enum` data type represents a defined hardcoded list of `string` values and uses the JSON string type [RFC8259] for serialization.

An enum contains a list of known values that apply to a specific property. Any normative rules for the use of that enum with a given property are defined in the definition of that property. When an enum is defined

for a property the value or values for that property **MUST** come from the enum. Additional values **MUST NOT** be added to an enum by an implementation.

10.~~9.8~~ 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`**, **`external_id`**, or **`reference_id`**.

Property Name	Data Type	Description
`name` (required)	`string`	The name of the author or title of the source of this external reference.
`description` (optional)	`string`	A detailed description of this external reference.
`source` (optional)	`string`	A textual citation of this source. The citation source **MAY** use a standard citation format like Chicago, MLA, APA, or similar style.
`url` (optional)	`string`	A URL [RFC3986] for this external reference.
`external_id` (optional)	`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` (optional)	`string`	An identifier that represents the data that this content is referring to. This property is especially useful when referencing content that already exists in a graph dataset or can be referenced via some ID. When referencing STIX content, this would be the STIX-based UUID.

Example 10.1

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

~~**9.1**~~

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

```
"external_references": [
    {
      "name": "ACME Security FuzzyPanda Report",
      "description": "ACME security review of FuzzyPanda 2021",
      "source": "ACME Security Company, Solutions for FuzzyPanda 2021, January 2021.
         Available online: hxxp://www[.]example[.]com/info/fuzzypanda2021.html",
      "url": "hxxp://www[.]example[.]com/info/fuzzypanda2021.html",
      "external_id": "fuzzypanda 2023.01",
      "reference_id": "malware--2008c526-508f-4ad4-a565-b84a4949b2af"
    }
]
```

~~9.9~~10.10 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. 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 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.

All CACAO objects **SHOULD** use UUIDv4 for the UUID portion of the identifier. A CACAO `playbook` object **MAY** use UUIDv5 for the UUID portion of the identifier. All CACAO `step` objects **MUST** use UUIDv4.

Using a UUIDv5 for the `playbook` **MAY** allow producers and consumers using the same namespace and contributing properties to generate the same identifier for that playbook. When using UUIDv5 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 a series of properties from the object that will ensure a globally unique identifier and those properties **SHOULD** be stringified according to the JSON Canonicalization Scheme [RFC8785] to ensure a canonical representation of the JSON data
- The contributing properties to the `playbook` object's UUIDv5 name portion **SHOULD** be the **`name`** and **`playbook_types`** properties.
- Producers not following these rules **MUST NOT** use a namespace of aa7caf3a-d55a-4e9a-b34e-056215fba56a

~~9.~~10.11 Integer

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

~~9.11~~10.12 List

The `list` data type defines a sequence of values ordered based on how they appear in the list. The phrasing "`list` of <type>" is used to indicate that all values within the list **MUST** conform to the specified type. For instance, `list` of type `integer` means that all values of the `list` **MUST** be of the `integer` type. This specification does not specify the maximum number of allowed values in a `list`; however, every instance of a `list` **MUST** have at least one value. Specific CACAO Object properties may define more restrictive upper and/or lower bounds for the length of the list.

Empty lists are prohibited in CACAO and **MUST NOT** be used as a substitute for omitting the property if it is optional. If the property is required, the list **MUST** be present and **MUST** have at least one value.

10.13 Open Vocabularies

The `open-vocab` data type represents a defined list of suggested `string` values and uses the JSON string type [RFC8259] for serialization.

An open vocabulary contains a list of known values that apply to a specific property. Any normative rules for the use of that vocabulary with a given property are defined in the definition of that property. When an open vocabulary is defined for a property the value or values for that property **SHOULD** come from the suggested vocabulary, but **MAY** be any other `string` value. Values that are not from the suggested vocabulary ~~SHOULD~~**MUST** be all lowercase and ~~SHOULD~~**MUST** use hyphens instead of spaces or underscores as word separators.

~~9.12~~10.14 Playbook Processing Summary

The `playbook-processing-summary` data type represents the major processing features and functionality of a playbook and contains a summarized list of the processing features that can be implemented for a specific playbook and is defined at the playbook metadata level. This is done to help implementations ~~understand~~identify the concepts and features used within a specific playbook without having to parse the entire playbook.

Property Name	Data Type	Details
`manual_playbook` (optional)	`boolean`	This type of playbook contains only manual commands and simple text-based descriptions or tasks.
`external_playbooks` (optional)	`boolean`	See section 4.6.
`parallel_processing` (optional)	`boolean`	See section 4.7.
`if_logic` (optional)	`boolean`	See section 4.8.
`while_logic` (optional)	`boolean`	See section 4.9.
`switch_logic` (optional)	`boolean`	See section 4.10.
`temporal_logic` (optional)	`boolean`	See section 4.1 **`delay`** and **`timeout`** properties.
`data_markings` (optional)	`boolean`	See section 2.4 and section ~~8~~9.
`digital_signatures` (optional)	`boolean`	See section ~~9.13~~10.15.
`countersigned_signatures` (optional)	`boolean`	See section ~~9.13~~10.15.
`extensions` (optional)	`boolean`	See section ~~7~~8.

~~9.13~~10.15 Signature

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

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

Property Name	Data Type	Description
`type` (required)	`string`	The value of this property **MUST** be `jss`.
`id` (required)	`identifier`	A value that uniquely identifies the signature. All signatures with the same ID are considered different versions of the same signature and the version of the signature is identified by its **`modified`** property.
`created_by` (optional)	`identifier`	An ID that represents the entity that created this signature. The ID **MUST** represent a STIX 2.1+ identity object.
`created` (required)	`timestamp`	The time at which this signature was originally created. The creator can use any time it deems most appropriate as the time the signature was created, but it **MUST** be precise to the nearest millisecond (exactly three digits after the decimal place in seconds). The created property **MUST NOT** be changed when creating a new version of the signature.
`modified` (required)	`timestamp`	The time that this particular version of the signature was last modified. The creator can use any time it deems most appropriate as the time that this version of the signature was modified, but it **MUST** be precise to the nearest millisecond (exactly three digits after the decimal place in seconds). The modified property **MUST** be later than or equal to the value of the created property. If the **`created`** and **`modified`** properties are the same, then this is the first version of the signature.
`revoked` (optional)	`boolean`	A boolean that identifies if the signature creator deems that this signature is no longer valid. The default value is `false`.
`signee` (required)	`string`	The name of the entity or organization that produced this signature. This property is similar to the X.509 fields.
`valid_from` (optional)	`timestamp`	The time from which this signature is considered valid. If omitted, the signature is valid at all times or until the timestamp defined by **`valid_until`**. If the **`revoked`** property is `true` then this property **MUST** be ignored.
`valid_until` (optional)	`timestamp`	The time at which this signature should no longer be considered valid.

		If the **valid_until** property is omitted, then there is no constraint on the latest time for which the signature is valid. This property **MUST** be greater than the timestamp in the **valid_from** property if the **valid_from** property is defined. If the **revoked** property is `true` then this property **MUST** be ignored.
related_to (required)	`identifier`	~~The CACAO~~A value that can identify the original playbook object that was signed with this signature ~~is for~~. The value of this property **MUST** be a CACAO playbook id. If the signature is detached from the original playbook object then this property **MUST** be populated.
related_version (required)	`timestamp`	~~The~~ A value that can identify the version of the ~~CACAO~~original playbook object that was signed with this signature ~~is for~~. The value of this property **MUST** be the **modified** `timestamp` from the CACAO playbook that this signature is for.
hash_algorithm (required)	`string`	This property identifies the hashing algorithm, as defined by IANA, that was used to hash the JCS version of the full playbook object (~~playbook~~Playbook Object + X.jss ~~signature~~Signature) and is a case-sensitive ASCII string. As of this writing, implementations **SHOULD** use `sha-256` or `sha-512` but **MAY** use any current and widely accepted hashing algorithm that is defined in the IANA registry. The actual signing process, defined in the **algorithm** property, sometimes uses an internal hashing algorithm inside the signing process itself, this property **MAY** identify the same hashing algorithm as the signing process or **MAY** identify a different hashing algorithm.
algorithm (required)	`open-vocab`	This property identifies the algorithm that was used to sign the playbook and is a case-sensitive ASCII string. The value for this property **SHOULD** come from the `signature-algorithm-type-ov` vocabulary and **SHOULD** be a current and widely accepted quantum safe algorithm, but **MAY** be any currently accepted safe algorithm. At the time of this writing quantum safe algorithms could come from those defined in XMSS [RFC 8391] section 5.3 or LMS [RFC 8554] section 5.1 and other algorithms could come from those defined in JWA [RFC 7518] section 3.1 or [RFC 8037] section 3.1. While JWA [RFC7518] section 3.1 defines the following symmetric algorithms: `HS256`, `HS384`, and `HS512` these algorithms **SHOULD NOT** be used, as CACAO playbooks are intended to be shared across system and organizational boundaries. If one of these three symmetric algorithms or some other symmetric algorithm is used, then the sharing and transmission of those keys is out of scope for this specification.

public_key (optional*)	string	This property contains a PEM encoded public key without the header and footer for the algorithm selected in the **algorithm** property.
public_cert_chain (optional*)	list of string	This property contains a public key certificate for the algorithm selected in the **algorithm** property and **MUST** follow the requirements defined in section 4.7 of [RFC7517] as quoted here. This property "contains a chain (X.509 certificate chain) of one or more PKIX certificates [RFC5280]. The certificate chain is represented as a JSON array of certificate value strings. Each string in the array is a base64-encoded (Section 4 of [RFC4648] -- not base64URL.encoded) DER [ITU.X690.1994] PKIX certificate value. The PKIX certificate containing the key value **MUST** be the first certificate. This **MAY** be followed by additional certificates, with each subsequent certificate being the one used to certify the previous one. The key in the first certificate **MUST** match the public key." This property is called "x5c" in section 4.7 of [RFC7517].
cert_url (optional*)	string	This property contains a URI [RFC3986] that refers to a resource for an X.509 public key certificate or certificate chain [RFC5280] for the algorithm selected in the **algorithm** property and **MUST** follow the requirements defined in section 4.6 of [RFC7517] as quoted here. "The identified resource **MUST** provide a representation of the certificate or certificate chain that conforms to RFC 5280 [RFC5280] in PEM-encoded form, with each certificate delimited as specified in section ~~6~~7.1 of RFC 4945 [RFC4945]. The key in the first certificate **MUST** match the public key. The protocol used to acquire the resource **MUST** provide integrity protection; an HTTP GET request to retrieve the certificate **MUST** use TLS [RFC2818] [RFC5246]; the identity of the server **MUST** be validated, as per section 6 of RFC 6125 [RFC6125]." This property is called "x5u" in section 4.6 of [RFC7517].
thumbprint (optional*)	string	This property contains a fingerprint of a public key or public key certificate for the algorithm selected in the **algorithm** property and **MUST** follow the requirements defined in section 4.9 of [RFC7517] as quoted here. This property "is a base64URL.encoded SHA-256 thumbprint (a.k.a. digest, X.509 certificate SHA-256 thumbprint) of the DER encoding of an X.509 certificate [RFC5280]. Note that certificate thumbprints are also sometimes known as certificate fingerprints. The key in the certificate **MUST** match the public key." This property is called "x5t#S256" in section 4.9 of [RFC7517].
value (required)	string	A base64URL.encoded signature that was created using the signature algorithm defined in the **algorithm** property and a key. In pseudo code it is defined as:

		base64URL.encode(sign(algorithm, key, hash(jcs(<JSONObject with Signature Object>)))).
signature (optional)	`signature`	This property enables a signature to be countersigned, meaning a signature can be signed by another signature.

~~9.13~~10.15.1 Signature Algorithm Type ~~Enumeration~~Vocabulary

~~Enumeration~~Vocabulary **Name:** `signature-algorithm-type-~~enum~~ov`

Vocabulary Value	Description
`RS256`	RSASSA-PKCS1-v1_5 using SHA-256. See section 3.3 of JWA [RFC 7518] for more information. This method is recommended per JWA [RFC7518].
`RS384`	RSASSA-PKCS1-v1_5 using SHA-384. See section 3.3 of JWA [RFC 7518] for more information.
`RS512`	RSASSA-PKCS1-v1_5 using SHA-512. See section 3.3 of JWA [RFC 7518] for more information.
`ES256`	ECDSA using P-256 and SHA-256. See section 3.4 of JWA [RFC 7518] for more information. This method is recommended per JWA [RFC7518].
`ES384`	ECDSA using P-384 and SHA-384. See section 3.4 of JWA [RFC 7518] for more information.
`ES512`	ECDSA using P-521 and SHA-512. See section 3.4 of JWA [RFC 7518] for more information.
`PS256`	RSASSA-PSS using SHA-256 and MGF1 with SHA-256. See section 3.5 of JWA [RFC 7518] for more information.
`PS384`	RSASSA-PSS using SHA-384 and MGF1 with SHA-384. See section 3.5 of JWA [RFC 7518] for more information.
`PS512`	RSASSA-PSS using SHA-512 and MGF1 with SHA-512. See section 3.5 of JWA [RFC 7518] for more information.
`Ed25519`	See [RFC 8037] and [RFC 8032] for more information. NOTE: Unlike RFC8037 [RFC8037] this specification requires explicit Ed* algorithm names (e.g. Ed25519) instead of generic versions like "EdDSA".
`Ed448`	See [RFC 8037] and [RFC 8032] for more information. NOTE: Unlike RFC8037 [RFC8037] this specification requires explicit Ed* algorithm names (e.g. Ed448) instead of generic versions like "EdDSA".
`XMSS-SHA2_10_256`	See section 5.3 of XMSS [RFC 8391] for more information.
`XMSS-SHA2_16_256`	See section 5.3 of XMSS [RFC 8391] for more information.

`XMSS-SHA2_20_256`	See section 5.3 of XMSS [RFC 8391] for more information.
`LMS_SHA256_M32_H5`	See section 5.1 of ~~LBS~~LMS [RFC 8554] for more information.
`LMS_SHA256_M32_H10`	See section 5.1 of ~~LBS~~LMS [RFC 8554] for more information.
`LMS_SHA256_M32_H15`	See section 5.1 of ~~LBS~~LMS [RFC 8554] for more information.
`LMS_SHA256_M32_H20`	See section 5.1 of ~~LBS~~LMS [RFC 8554] for more information.
`LMS_SHA256_M32_H25`	See section 5.1 of ~~LBS~~LMS [RFC 8554] for more information.

~~9.14~~10.16 String

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

~~9.15~~10.17 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. Additional requirements may be defined where this data type is used.

~~9.16~~10.18 Variables

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

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

- **MUST** be unique within the contextual scope they are declared
- **MUST** be prefixed and suffixed with __ (two underscore characters) for both declaration and use
- **MUST** contain the keyword ":value" when using the variable (e.g., __ipaddress__:value)
- **MUST** include the double underscore "__" for the variable name prefix and suffix (a total of four characters) as part of the 250 ASCII character length limit.
- **MUST** start with a letter after the variable prefix __
- Are case-sensitive (age, Age and AGE are three different variables) but **SHOULD** be lowercase

~~9.16~~10.18.1 Variable Scope

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

object used or referenced by that object (e.g., a playbook variable can be assigned at the playbook level but also reassigned a different value within a workflow step).

~~9.16~~10.18.2 Using Variables

Variables are referenced by using the key name from the dictionary with a suffix (keyword) of ":value". For example, if you had a variable in the dictionary called "__ip_addresses__", one could reference this and use it in a playbook by using "__ip_addresses__:value". Variables **MAY** be passed to and from external playbooks provided that system supports passing of arguments when the system function is invoked or returns its results.

~~9.16~~10.18.3 Variable

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

Property Name	Data Type	Details
`type` (required)	`open-vocab`	The type of variable being used. The value for this property **SHOULD** come from the `variable-type-ov` vocabulary.
`description` (optional)	`string`	An optional detailed description of this variable.
`value` (optional)	`string`	The value **MUST** be defined as one of the ~~variable represented by a~~ following JSON types: a string~~. The value **MAY be populated with**~~, a ~~string value (or~~ number or boolean encoded as a JSON string~~),~~, an empty string "", ~~or with~~ the special JSON NULL value~~.~~, or a JSON object. 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` (optional)	`boolean`	~~Is~~This property defines if this variable is immutable~~ or mutable?~~. If true, the variable is immutable and **MUST NOT** be changed. If false, the variable is mutable and can be updated later on in the playbook. The default value is `false`. If this property is not present then the value is `false`.
`external` (optional)	`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.

Example ~~9~~10.2

```
{
  "type": "playbook",
  …,
  "playbook_variables": {
    "__variable name__": {
```

```
      "type": "<variable_type>",
      "description": "<details about variable>",
      "value": "<variable_value>",
      "constant": false,
      "external": false
    }
  }
}
```

Example ~~9~~10.3

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

~~9.16~~10.18.4 Variable Type Vocabulary

Vocabulary Name: `variable-type-ov`

Vocabulary Value	Description	Examples
`bool`	The value is a true or false value encoded as a string	`"type": "bool", "value": "true"`
`dictionary`	The value contains a dictionary of values.	`"type": "dictionary", "value": {"a":"fun"}`
`float`	A floating point number encoded as a string	`"type": "float", "value": "3.14159265"`
`hexstring`	Some string encoded in hexadecimal with a leading 0x.	`"type": "hexstring", "value": "0xFCA1"`
`integer`	An integer encoded as a string	`"type": "integer", "value": "123"`
`ipv4-addr`	An IPv4 network address (e.g., 127.0.0.1)	`"type": "ipv4-addr", "value": "127.0.0.1"`
`ipv6-addr`	An IPv6 network address (e.g, fe80::8785:b894:75aa:c16f) See [RFC 5952] for information about normalizing the address.	`"type": "ipv6-addr", "value": "2001:db8::1"`
`long`	A long number value encoded as a string	`"type": "long", "value": "-2147483647"`

mac-addr	A layer 2 network MAC address (e.g., bc:d0:74:7a:3a:31)	`"type": "mac-addr", "value": "bc:d0:74:7a:3a:31"`
hash	A hash encoded as a string	`"type": "hash", "value": "ef9e7175fe883e3dc0d77dfad982846b"`
md5-hash	An MD5 hash encoded as a string	`"type": "md5-hash", "value": "ef9e7175fe883e3dc0d77dfad982846b"`
sha256-hash	A SHA 256 hash encoded as a string	`"type": "sha256-hash", "value": "c85406dd4c0b149a519a7715d5b8b17afe855594ef16eaada9be23a5aada155c"`
string	A normal string value	`"type": "string", "value": "some text"`
uri	A URI address	`"type": "uri", "value": "http://www.example.com/"`
uuid	An RFC 4122-compliant UUID [RFC4122].	`"type": "uuid", "value": "ded7157d-ba68-48c5-a093-b6c7b6bafd5f"`

~~10~~

11 Conformance

~~10~~11.1 CACAO Playbook Producers and Consumers

A "CACAO 2.0 Producer" is any software that can create CACAO 2.0 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~~11.2, Mandatory Features.
- It **MAY** support any features listed in section ~~10~~11.3, 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.
- It **MAY** produce content in an earlier CACAO version if it determines no loss of semantic functionality.

A "CACAO 2.0 Consumer" is any software that can consume CACAO content and conforms to the following normative requirements:

- It **MUST** support parsing of all required properties for the content that it consumes from this version of the Specification.
- It **MUST** support all features listed in section ~~10~~11.2, Mandatory Features.
- It **MAY** support any features listed in section ~~10~~11.3, 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.
- It **MUST** support parsing of all required properties and mandatory to implement features as defined in previous minor versions of this version of the CACAO Specification.
- It **MAY** support parsing of optional properties and optional to implement features as defined in previous minor versions of this version of the CACAO Specification.
- It **MAY** support parsing of required and optional properties and mandatory and optional to implement features as defined in newer versions of the CACAO Specification.

~~10~~11.1.1 CACAO 2.0 to 1.1 Version Compatibility

Prior to exchanging CACAO playbook content between a CACAO Producer and a CACAO Consumer it is recommended to verify version compatibility between the various systems involved to ensure improved interoperability.

- CACAO Consumers and Producers **SHOULD** verify version compatibility prior to exchanging content.

~~10~~11.2 CACAO Mandatory Features

~~10~~11.2.1 Versioning

CACAO 2.0 Producers and CACAO 2.0 Consumers **MUST** support versioning by following the normative requirements listed in section 2.3.

~~10~~11.2.2 Variables

CACAO 2.0 Producers and CACAO 2.0 Consumers **MUST** support variables by following the normative requirements listed in sections 3.3 and ~~9.16~~10.18.

~~10~~11.2.3 Playbooks

CACAO 2.0 Producers and CACAO 2.0 Consumers **MUST** support the playbook object defined in this specification by following the normative requirements listed in section 3

~~10~~11.2.4 Workflow Steps

CACAO 2.0 Producers and CACAO 2.0 Consumers **MUST** support the workflow steps defined in this specification by following the normative requirements listed in sections 3.1 and 4.

~~10~~11.2.5 Commands

CACAO 2.0 Producers and CACAO 2.0 Consumers **MUST** support the command object as defined in this specification by following the normative requirements listed in sections 3.1 and 5. However, a CACAO 2.0 Producer or CACAO 2.0 Consumer **MAY** support only a subset of command object types.

~~10~~11.2.6 ~~Agents~~Authentication Types

CACAO 2.0 Producers and CACAO 2.0 Consumers **MUST** support the ~~agents defined in this specification by following the normative requirements listed in sections 3.1 and 6.~~

~~10.2.7 Targets~~

~~CACAO 2.0 Producers and CACAO 2.0 Consumers **MUST** support the targets~~authentication types defined in this specification by following the normative requirements listed in sections 3.1 and 6.

~~10~~11.2.7 Agents

CACAO 2.0 Producers and CACAO 2.0 Consumers **MUST** support the agents defined in this specification by following the normative requirements listed in sections 3.1 and 7.

11.2.8 Targets

CACAO 2.0 Producers and CACAO 2.0 Consumers **MUST** support the targets defined in this specification by following the normative requirements listed in sections 3.1 and 7.

11.3 CACAO Optional Features

~~10~~11.3.1 Data Markings

CACAO 2.0 Producers and CACAO 2.0 Consumers **MAY** support Data Markings. Software that supports Data Markings **MUST** follow the normative requirements listed in sections 2.4, 3.1, and ~~8~~9.

~~10~~11.3.2 Extensions

CACAO 2.0 Producers and CACAO 2.0 Consumers **MAY** support Extensions. Software that supports Extensions **MUST** follow the normative requirements listed in sections 3.1 and ~~7~~8.

~~10~~11.3.3 Digital Signatures

CACAO 2.0 Producers and CACAO 2.0 Consumers **MAY** support Digital Signatures. Software that supports Digital Signatures **MUST** follow the normative requirements listed in sections 2.5, 3.1 and ~~9.13~~10.15.

Appendix A. Examples

Please see the following Github repository for various CACAO 2.0 examples:

https://github.com/oasis-tcs/cacao/tree/master/Examples.

The examples found in this repo are based on various scenarios and are included to help readers understand how CACAO playbooks can be developed. In these examples it is common to not actually use UUIDs, but rather simple IDs to make it easier for visual human inspection. These simple IDs will have a form of "<object-type>--uuid1". 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.

Appendix B. Security and Privacy Considerations

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

B.1 Security Considerations

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

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

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

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

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

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

CACAO provides a graph-based object 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=1.1". The parameter value is of the form 'n.m', where n is the major version and m the minor version, both unsigned integer values.

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

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

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

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

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

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

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

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

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

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

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

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

Published specification:
CACAO Version 2.0 OASIS Committee Specification 01

https://docs.oasis-open.org/cacao/security-playbooks/v2.0/cs01/security-playbooks-v2.0-cs01.html

Cited in the "OASIS Standards" document:

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

https://www.oasis-open.org/standards#security-playbooks2.0

Applications which use this media:
Collaborative Automated Course of Action Operations (CACAO) defines a language and serialization format used to exchange cybersecurity playbooks. CACAO enables organizations to share playbooks with one another in a consistent and machine-readable manner, allowing security communities to better

understand how to respond to computer-based attacks and to anticipate and/or respond to those attacks faster and more effectively. CACAO is designed to improve many different capabilities, such as collaborative threat analysis, automated threat exchange, automated detection and response, and more.

Fragment identifier considerations: None

Restrictions on usage: None

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

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

3. File extension(s): cacao

4. Macintosh file type code: TEXT [RFC8259]

5. Object Identifiers: None

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

Intended usage: COMMON

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

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

Change controller: OASIS

Provisional registration: No

Appendix D. References

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

D.1 Normative References

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

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

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

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

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

[KESTREL]
Kestrel Threat Hunting Language. [Online]. Available: https://github.com/opencybersecurityalliance/kestrel-lang

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

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

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

[RFC4122]
A Universally Unique IDentifier (UUID) URN Namespace, RFC 4122, July 2005. [Online]. Available: https://www.rfc-editor.org/info/rfc4122

[RFC4648]
The Base16, Base32, and Base64 Data Encodings, RFC 4648, October 2006. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc4648#section-4

[~~RFC5849~~RFC5280]
Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, RFC 5280, DOI 10.17487/RFC5280, May 2008. [Online]. Available: https://www.rfc-editor.org/info/rfc5280

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

[RFC 5952]
A Recommendation for IPv6 Address Text Representation, RFC 5952, August 2010, [Online]. Available: https://www.rfc-editor.org/info/rfc5952

[RFC6749]
The OAuth 2.0 Authorization Framework, RFC 6749, October 2012. [Online]. Available: https://www.rfc-editor.org/info/rfc6749

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

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

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

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

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

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

[RFC8032]
Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, https://www.rfc-editor.org/info/rfc8032.

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

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

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

[RFC8391]
XMSS: eXtended Merkle Signature Scheme, RFC 8391, May 2018. [Online]. Available: https://www.rfc-editor.org/info/rfc8391.

[RFC8554]
Leighton-Micali Hash-Based Signatures, RFC 8554, April 2019. [Online]. Available: https://www.rfc-editor.org/info/rfc8554.

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

[SIGMA]
SIGMA - Generic Signature Format for SIEM Systems. [Online]. Available: https://github.com/SigmaHQ/sigma

[STIX-v2.1]
STIX™ Version 2.1. Edited by Bret Jordan, Rich Piazza, and Trey Darley. 10 June 2021. OASIS Standard. https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.html. Latest stage: https://docs.oasis-open.org/cti/stix/v2.1/stix-v2.1.html

[TLP]
FIRST. "Traffic Light Protocol, Version 2.0 (TLP)". 2022, Aug. 5. [Online]. Available: https://first.org/tlp/

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

D.2 Informative References

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

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

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

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

[Engage]
MITRE Engage™. Accessed: September 2022. [Online]. Available: https://engage.mitre.org/starter-kit/

[Google Maps]
Google Maps Platform. "Coordinates". Accessed: January 12, 2023. Available: https://developers.google.com/maps/documentation/javascript/reference/coordinates#:~:text=Latitude%20is%20specified%20in%20degrees,range%20%5B%2D180%2C%20180).

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Appendix E. Acknowledgments

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

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

Bret Jordan, Individual
Allan Thomson, Individual
Marlon Taylor, DHS Office of Cybersecurity and Communications (CS&C)
Stephanie Hazlewood, IBM
Emily Ratliff, IBM
Desiree Beck, MITRE Corporation
Richard Piazza, MITRE Corporation
Andrew Storms, New Context Services, Inc.
Lior Kolnik, Palo Alto Networks
Marco Caselli, Siemens AG
Luca Morgese Zangrandi, TNO
Mateusz Zych, University of Oslo
Vasileios Mavroeidis, University of Oslo & Sekoia.io

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

Curtis Kostrosky, Accenture
Anup Ghosh, Accenture
Patrick Maroney, AT&T
Dean Thompson, Australia and New Zealand Banking Group (ANZ Bank)
Arnaud Taddei, Broadcom
Naasief Edross, Cisco Systems
Michael Simonson, Cisco Systems
Omar Santos, Cisco Systems
Jyoti Verma, Cisco Systems
Andrew Storms, Copado
Jane Ginn, CTIN
Ryan Hohimer, CTIN
Christian Hunt, CTIN
Ben Ottoman, CTIN
Christopher Robinson, CTIN
Arsalan Iqbal, CTM360
Avkash Kathiriya, Cyware Labs
Ryan Joyce, DarkLight, Inc.
Paul Patrick, DarkLight, Inc.

Michael Rosa, DHS Office of Cybersecurity and Communications (CS&C)
Marlon Taylor, DHS Office of Cybersecurity and Communications (CS&C)
Aukjan van Belkum, EclecticIQ
Gerald Stueve, Fornetix
Chris O'Brien, Google Inc.
Stephanie Hazlewood, IBM
Jason Keirstead, IBM
Emily Ratliff, IBM
John Morris, IBM
Mahbod Tavallaee, IBM
Srinivas Tummalapenta, IBM
Francisco de AndrÈs PÈrez, Individual
Joerg Eschweiler, Individual
Bret Jordan, Individual
Terry MacDonald, Individual
Anil Saldanha, Individual
Frans Schippers, Individual
Allan Thomson, Individual
Rodger Frank, Johns Hopkins University Applied Physics Laboratory
Karin Marr, Johns Hopkins University Applied Physics Laboratory
Chris Dahlheimer, LookingGlass
Jason Webb, LookingGlass
Desiree Beck, MITRE Corporation
Richard Piazza, MITRE Corporation
David Kemp, National Security Agency
Christian Hunt, New Context Services, Inc.
Andrew Storms, New Context Services, Inc.
Kaleb Wade, New Context Services, Inc.
Stephen Banghart, NIST
David Darnell, North American Energy Standards Board
Lior Kolnik, Palo Alto Networks
David Bizeul, SEKOIA
Duncan Sparrell, sFractal Consulting LLC
Marco Caselli, Siemens AG
Alexandre Cabrol Perales, Sopra Steria Group
Baptiste Decrand, Sopra Steria group
Manos Athanatos, Telecommunication Systems Institute
Greg Reaume, TELUS
Ryan Trost, ThreatQuotient, Inc.
Franck Quinard, TIBCO Software Inc.
Frank Fransen, TNO
Luca Morgese Zangrandi, TNO
Sebastiaan Tesink, TNO
Toby Considine, University of North Carolina at Chapel Hill
Vasileios Mavroeidis, University of Oslo & Sekoia.io
Mateusz Zych, University of Oslo

Appendix F. Revision History

Revision	Date	Editor(s)	Changes Made
01	2023-02-13	Bret Jordan, Allan Thomson	Initial 2.0 Version. Minor editorial changes found during publishing of CACAO v1.1 CSD02. These consisted of URL changes to OASIS boilerplate text. Significant changes per public review, please see the change log. Removed some of the b64 encoding that was used during the signing process and added a hash_algorithm property. Added playbook activities, removed playbook-templates, changed to TLPv2. Removed the spec_version property from data-markings and signatures. Changed targets to agents and targets. Refactored the attack agent/target. Made lots of other small changes. Changed version to 2.0. This version is going to ballot and 30-day public review to become CSD01.
02	2023-05-17	Bret Jordan, Allan Thomson	Minor changes during public review. Fixed some of the examples. Synced variable lengths to dictionary lengths. Renamed the agents and targets properties on the playbook. Lots of editorial changes. Made changes to the signature object to allow for quantum safe versions. This version is going to ballot and 15-day public review to become CSD02.
03	2023-08-15	Bret Jordan, Allan Thomson	Some editorial changes. Changed two of the SHOULDs to MUSTs in section 10.13 Change the signature vocab to be an open-vocabulary instead of an enum. Added quantum safe options to the signature methods. Created an identity object for the CACAO TC. Added authentication information objects to abstract that away from the Agents and Targets and added support for KMS

			systems. Added the extensions property to the main Playbook object. Added some clarifying text in section 10.18.3 for the value property to make it clear by what is meant by strings. Fixed the IEP to be current with the FIRST definition. Removed modified property from data markings. This version is going to ballot and 15-day public review to become CSD03.

Appendix G. Notices

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website: [https://www.oasis-open.org/policies-guidelines/ipr/]

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

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

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. OASIS AND ITS MEMBERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THIS DOCUMENT OR ANY PART THEREOF.

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

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

[OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this OASIS Standards Final Deliverable by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this OASIS Standards Final Deliverable. OASIS may include such claims on its website, but disclaims any obligation to do so.].

[OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this OASIS Standards Final Deliverable or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS

Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Standards Final Deliverable, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims~~.~~.

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

