# STIX™ 2.0 Specification

Version 2.0-rc2

## Technical Committee

OASIS Cyber Threat Intelligence (CTI) TC

## Chair

Richard Struse (Richard.Struse@hq.dhs.gov), DHS Office of Cybersecurity and Communications (CS&C)

## Editors

Bret Jordan (bret.jordan@bluecoat.com), Blue Coat Systems, Inc.
Rich Piazza (rpiazza@mitre.org), MITRE Corporation
John Wunder (jwunder@mitre.org), MITRE Corporation

## Table of Contents

# 1. Introduction

Structured Threat Information Expression (STIX™) is a language and serialization format used to exchange cyber threat intelligence (CTI). STIX enables organizations to share CTI with one another in a consistent and machine readable manner, allowing security communities to better understand what computer-based attacks they are most likely to see and to anticipate and/or respond to those attacks faster and more effectively. STIX is designed to improve many different capabilities, such as collaborative threat analysis, automated threat exchange, automated detection and response, and more.

In response to lessons learned in implementing previous versions, STIX has been significantly redesigned (TODO: add reference to STIX 1.2.1) and, as a result, omits some of the objects and properties defined in STIX 1.2.1. The objects chosen for inclusion in STIX 2.0 represent a minimally viable product (MVP) that fulfills basic consumer and producer requirements for CTI sharing. Objects and properties not included in STIX 2.0, but deemed necessary by the community, will be included in future releases.

## 1.1. Terminology

The keywords "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in RFC2119 (REF:RFC2119).

**CAPEC** - Common Attack Pattern Enumeration and Classification
**Consumer** - Any entity that receives STIX content.
**CTI** - Cyber Threat Intelligence
**CybOX** - Cyber Observable Expression (Spec Ref TODO)
**Entity** - Anything that has a separately identifiable existence (e.g., organization, person, group, etc.).
**IEP** - FIRST (Forum of Incident Response and Security Teams) Information Exchange Policy
**Instance** - A single occurrence of a STIX object version.
**MTI** - Mandatory to Implement
**MVP** - Minimally Viable Product
**Object Creator** - The entity that created a STIX object (See Section TODO).
**Object Representation** - An instance of an object version that is serialized as STIX.
**Producer** - Any entity that distributes STIX content, including object creators as well as those passing along existing content.
**SDO -** STIX Domain Object
**SRO** - STIX Relationship Object
**STIX** - Structured Threat Information Expression
**STIX Content** - STIX documents, including STIX Objects, STIX Objects grouped as bundles, etc.
**STIX Object** - A STIX Domain Object (SDO) or STIX Relationship Object (SRO)

**TAXII** - Trusted Automated Exchange of Indicator Information
**TLP** - Traffic Light Protocol

## 1.2. Overview

### 1.2.1. Graph-Based Model

STIX 2 is graph-based, in the sense of a connected graph of nodes and edges. STIX Domain Objects define the graph nodes and STIX relationships (including STIX Relationship Objects and embedded relationships) define the edges. The full set of STIX Domain Objects and STIX Relationship Objects are known as STIX Objects. This graph-based language conforms to common analysis approaches and allows for flexible, modular, structured, and consistent representations of CTI.

### 1.2.2. STIX Domain Objects

STIX 2.0 defines a set of STIX Domain Objects (SDOs): Attack Pattern, Campaign, Course of Action, Identity, Indicator, Intrusion Set, Malware, Observed Data, Report, Threat Actor, Tool, and Vulnerability. Each of these objects correspond to a concept commonly represented in CTI. Using the building blocks of SDOs alongside STIX relationships, individuals can create and share broad and comprehensive cyber threat intelligence.

STIX Domain Objects all share a common set of properties. These common properties provide standard capabilities such as versioning, data marking (representing how data can be shared and used), and extensibility.

### 1.2.3. STIX Relationships

A relationship is a link between STIX Objects that describes the way in which the objects are related. Most relationships are represented using STIX Relationship Objects (SROs), while other special embedded relationships are represented as ID references.

The generic Relationship object is one of two SROs and is used for most relationships in STIX. This generic Relationship object contains a property called **`relationship_type`** to describe more specifically what the relationship represents. This specification defines a set of known terms to use for the **`relationship_type`** property between SDOs of specific types. For example, the Indicator SDO defines a relationship from itself to Malware with a **`relationship_type`** of `indicates` to describe how the Indicator can detect or indicate the presence of that Malware. In addition to the terms defined in the specification, STIX also allows for custom terms to be used as the relationship type.

Currently the only other SRO (besides a generic Relationship) is the Sighting relationship object. The Sighting object is used to capture cases where an entity has "seen" an SDO, such as sighting an indicator. Sighting is a separate SRO because it contains additional properties such as **`count`** that are only applicable to Sighting relationships. Other SROs may be defined in future versions of STIX if new relationships are identified that also require additional properties not present on the generic Relationship object.

In addition to relationships created using the SROs (Relationship and Sighting), STIX also uses ID references to represent embedded relationships. Embedded relationships are simply ID reference properties on STIX Objects that contain the ID of a different STIX Object. Embedded relationships are used when the property is an inherent part of the object and not something that a third party might add or something that might require a confidence. Because they represent a simply inherent linkage and have no other properties, an SRO is not needed to represent them. An embedded relationship can only be asserted by the creator of the object ("object creator") it is contained in.

For example, the entity that created a STIX Object is an inherent, factual part of that object and therefore that information is captured in an embedded relationship contained in the **`created_by_ref`** property rather than through the use of an SRO.

### 1.2.4. Vocabularies

Many STIX Objects contain properties whose values can be selected from a defined set of values. These sets of values are called vocabularies and are defined in STIX in order to enhance interoperability by increasing the likelihood that different entities use the same exact string to represent the same concept. If used consistently, vocabularies make it less likely that one entity refers to the energy sector as “Energy” and another as “Energy Sector”, thereby making comparison and correlation easier.

While using predefined values from STIX vocabularies is encouraged, in some cases this is not possible or desirable. STIX supports this by defining vocabularies as “open”, where producers are permitted to use values outside of the suggested vocabulary.

### 1.2.5. Serialization

STIX is defined independent of any specific storage or serialization. However, the mandatory-to-implement (MTI) serialization for STIX 2 is JSON (TODO REF IETF). Therefore, all STIX 2-compatible tools **MUST** support JSON as a serialization format. STIX 2-compatible tools **MAY** support serializations other than JSON.

As JSON is the MTI serialization, all examples in this document are expressed in JSON.

### 1.2.6. Transporting STIX

STIX 2 is transport-agnostic, i.e. the structures and serializations do not rely on any specific transport mechanism. A companion CTI specification, TAXII (TODO REF), is designed specifically to transport STIX Objects. STIX provides a Bundle (see Section TODO) as a container for STIX Objects to allow for transportation of bulk STIX data, especially over non-TAXII communication mechanisms.

## 1.3. Conventions

### 1.3.1. Naming Conventions

All type names, property names and literals are in lowercase. Words in property names are separated with an underscore (_), while words in type names and string enumerations are separated with a dash (-). All type names, property names, object names, and vocabulary terms are between three and 250 characters long.

In the JSON serialization all property names and string literals **MUST** be exactly the same, including case, as the names listed in the property tables in this specification. For example, the SDO common property **`created_by_ref`** must result in the JSON key name "created_by_ref". Properties marked required in the property tables **MUST** be present in the JSON serialization.

### 1.3.2. Reserved Property Names

Reserved property names are marked with a type called `RESERVED` and a description text of “RESERVED FOR FUTURE USE”. Any property name that is marked as `RESERVED` **MUST NOT** be present in STIX content conforming to this version of the specification.

### 1.3.3. Font Colors and Style

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 – `threat-actor`
  - property names are in bold style – **`created_at`**
  - literals (values) are in green with a green background – `malicious-activity`
  - All relationship types” are string literals, therefore they will also appear in green with a green background – `related-to`
- In an object's 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 pt font, with straight quotes, black text and a `light blue background`. All examples have a 2 space indentation. Parts of the example may be omitted for conciseness and clarity. These omitted parts are denoted with the ellipses (...).

# 2. Common Data Types

This section defines the common types used throughout STIX. These types will be referenced by the "Type" column in other sections. This section defines the names and permitted values of common types that are used in the STIX information model; it does not, however, define the meaning of any properties using these types. These types may be further restricted elsewhere in the document.

| Type | Description |
|---|---|
| `boolean` | A value of `true` or `false`. |
| `cybox-container` | A container for CybOX content. This type is defined by the [TODO CybOX Reference]. |
| `external-reference` | A non-STIX identifier or reference to other related external content. |
| `identifier` | An identifier (ID) for a STIX Domain Object, STIX Relationship Object, Bundle, or Marking Definition. |
| `kill-chain-phase` | A name of a kill chain phase. |
| `list` | An ordered sequence of values. The phrasing "`list` of type `<type>`" is used to indicate that all values within the list must conform to a specific type. |
| `number` | A numeric value. |
| `open-vocab` | A value from a STIX open (`open-vocab`) or suggested vocabulary. |
| `string` | A series of Unicode characters. |
| `timestamp` | A time value (date and time). |
| `timestamp-precision` | The level of precision for timestamps. |

## 2.1. Boolean

**Type Name:** `boolean`

A `boolean` is a value of either true or false. Properties with this type **MUST** have a value of `true` or `false`.

The JSON MTI serialization uses the JSON boolean type <TODO: add reference>, which is a literal (unquoted) `true` or `false`.

### 2.1.1. Examples

```
{
  ...
  "summary": true,
  ...
}
```

## 2.2. CybOX Container

**Type Name:** `cybox-container`

A container for CybOX content, as defined by [TODO Ref CybOX]. The structure is duplicated here simply as a reference; normative usage is defined by the CybOX language.

<todo: When CybOX is finished, copy in>

## 2.3. External Reference

**Type Name:** `external-reference`

External references are used to describe pointers to information represented outside of STIX. For example, a Malware object could use an external reference to indicate an ID for that malware in an external database or a report could use references to represent source material.

The JSON MTI serialization uses the JSON object type <TODO: add reference> when representing `external-reference`.

### 2.3.1. Properties

| Property Name | Type | Description |
|---|---|---|
| **`source_name`** (required) | `string` | The source within which the `external-reference` is defined (system, registry, organization, etc.). |
| **`description`** (optional) | `string` | A human readable description. |
| **`url`** (optional) | `string` | A URL reference to an external resource. [TODO: Reference to URL syntax] |

|  |  |  |
|---|---|---|
| **external_id** (optional) | `string` | An identifier for the external reference content. |

## 2.3.2. Requirements

- In addition to the **source_name** property, at least one of the **external_id**, **url**, or **description** properties **MUST** be present.

## 2.3.3. Examples

An `external-reference` to a VERIS Community Database (VCDB) [TODO:Add ref?] entry

```
{
  ...
  "external_references": [
    {
      "source_name": "veris",
      "external_id": "0001AA7F-C601-424A-B2B8-BE6C9F5164E7",
      "url": "https://github.com/vz-risk/VCDB/blob/master/data/json/0001AA7F-C601-424A-B2B8-
             BE6C9F5164E7.json"
    }
  ],
  ...
}
```

An `external-reference` from the CAPEC™ (TODO add ref) repository

```
{
  ...
  "external_references": [
    {
      "source_name": "capec",
      "external_id": "CAPEC-550"
    }
  ],
  ...
}
```

An `external-reference` from the CAPEC repository with URL

```
{
  ...
  "external_references": [
    {
      "source_name": "capec",
      "external_id": "CAPEC-550",
      "url": "http://capec.mitre.org/data/definitions/550.html"
    }
  ],
  ...
}
```

An `external-reference` to ACME Threat Intel's report document

```
{
  ...
  "external_references": [
    {
      "source_name": "ACME Threat Intel",
      "description": "Threat report",
      "url": "http://www.example.com/threat-report.pdf"
    }
  ],
  ...
}
```

An `external-reference` to a Bugzilla item

```
{
  ...
  "external_references": [
    {
      "source_name": "ACME Bugzilla",
      "external_id": "1370",
      "url": "https://www.example.com/bugs/1370"
    }
  ],
  ...
}
```

An `external-reference` to a offline threat report (i.e. e-mailed, offline, etc.)

```
{
  ...
  "external_references": [
    {
      "source_name": "ACME Threat Intel",
      "description": "Threat report"
    }
  ],
  ...
}
```

## 2.4. Identifier

**Type Name:** `identifier`

An `identifier` universally and uniquely identifies a SDO, SRO, Bundle, or Marking Definition. Identifiers **MUST** follow the form `[object-type]--[UUIDv4]`, where **`[object-type]`** is the exact value from the `type` field of the object being identified or referenced and where the

**`[UUIDv4]`** is an RFC 4122-compliant Version 4 UUID. The UUID **MUST** be generated according to the algorithm(s) defined in RFC 4122, Section 4.4 (Version 4 UUID) [add reference].

The JSON MTI serialization uses the JSON string type <TODO: add reference> when representing `identifier`.

### 2.4.1. Examples

```
{
  ...
  "type": "indicator",
  "id": "indicator--e2e1a340-4415-4ba8-9671-f7343fbf0836",
  ...
}
```

## 2.5. Kill Chain Phase

**Type Name:** `kill-chain-phase`

The `kill-chain-phase` represents a phase in a kill chain, which describes the various phases an attacker may undertake in order to achieve their objectives.

The JSON MTI serialization uses the JSON object type <TODO: add reference> when representing `kill-chain-phase`.

| Property Name | Type | Description |
|---|---|---|
| **`kill_chain_name`** (required) | `string` | The name of the kill chain. The value of this property **SHOULD** be all lowercase (where lowercase is defined by the locality conventions) and **SHOULD** use dashes instead of spaces or underscores as word separators. |
| **`phase_name`** (required) | `string` | The name of the phase in the kill chain. The value of this property **SHOULD** be all lowercase (where lowercase is defined by the locality conventions) and **SHOULD** use dashes instead of spaces or underscores as word separators. |

When referencing the Lockheed Martin Cyber Kill Chain™, the **`kill_chain_name`** **MUST** be `lockheed-martin-cyber-kill-chain`.

### 2.5.1. Examples

Example specifying the “reconnaissance” phase from the Lockheed Martin Cyber Kill Chain

```
{
  ...
  "kill_chain_phases": [
    {
      "kill_chain_name": "lockheed-martin-cyber-kill-chain",
      "phase_name": "reconnaissance"
    }
  ],
  ...
}
```

Example specifying the “pre-attack” phase from the “foo” kill-chain

```
{
  ...
  "kill_chain_phases": [
    {
      "kill_chain_name": "foo",
      "phase_name": "pre-attack"
    }
  ],
  ...
}
```

## 2.6. List

**Type Name:** `list`

The `list` type defines an ordered sequence of one or more values. The phrasing “`list` of type `<type>`” is used to indicate that all values within the list must conform to a specific type. For instance, `list` of type `number` means that all values of the list must be of the number 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 STIX object properties may define more restrictive upper and/or lower bounds for the length of the list.

Empty lists are prohibited in STIX and **MUST NOT** be used as a substitute for omitting the property if it is optional.

The JSON MTI serialization uses the JSON array type [TODO: Add ref?], which is an ordered list of zero or more values.

### 2.6.1. Examples

```
{
  ...
  "observed_data_refs": [
    "observed-data--b67d30ff-02ac-498a-92f9-32f845f448cf",
    "observed-data--c96f4120-2b4b-47c3-b61f-eceaa54bd9c6",
    "observed-data--787710c9-1988-4a1b-9761-a2de5e19c62f"
  ],
  ...
}
```

## 2.7. Number

**Type Name:** `number`

The `number` type represents any number that can be expressed as a real number (e.g., -10, 0, 10, 10.1, 10.123213). Each use of the `number` type may specify the following:

- The valid range of values;
- Whether it is limited to integers or not; and
- The maximum number of decimal places.

In the JSON MTI serialization, numbers are represented by the JSON number type [REF: https://tools.ietf.org/html/rfc7159].

### 2.7.1. Examples

```
{
  ...
  "count": 8,
  ...
}
```

## 2.8. String

**Type Name:** `string`

The `string` data type represents a finite-length string of valid characters from the Unicode coded character set [REF:ISO.10646]. Unicode incorporates ASCII [REF: RFC20] and the characters of many other international character sets.

The JSON MTI serialization uses the JSON string type [TODO: Add reference], which mandates the UTF-8 encoding for supporting Unicode.

### 2.8.1. Examples

```
{
  ...
  "title": "The Black Vine Cyberespionage Group",
  ...
}
```

## 2.9. Timestamp

**Type Name:** `timestamp`

The `timestamp` type defines how timestamps are represented in STIX. Most discrete timestamps (i.e., not time ranges or relative times) have a corresponding optional field that indicates the precision of the timestamp, of type `timestamp-precision`.

In cases where the timestamp is metadata about the STIX construct, such as the **`created`** property and the **`modified`** property on STIX Objects, the `timestamp` field will not have the corresponding precision field. In these cases, the timestamp **MUST** be treated as if the precision property is `full`.

The JSON MTI serialization uses the JSON string type <TODO: add reference> when representing `timestamp`.

### 2.9.1. Requirements

- The `timestamp` field **MUST** be a valid RFC 3339-formatted timestamp [TODO add reference] using the format `YYYY-MM-DDTHH:mm:ss[.s+]Z` where the “s+” represents 1 or more sub-second values. The brackets denote that subsecond 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 timezone and **MUST** use the “Z” designation to indicate this.

### 2.9.2. Examples

A `timestamp` that does not have a corresponding precision field

```
{
  ...
  "created": "2016-01-20T12:31:12.12345Z",
  ...
}
```

Examples of timestamps with a corresponding precision field are located in the following section.

## 2.10. Timestamp Precision

**Type Name:** `timestamp-precision`

The `timestamp-precision` type represents the precision options for a given `timestamp`.

The JSON MTI serialization uses the JSON string type <TODO: add reference> when representing `timestamp-precision`.

### 2.10.1. Requirements

- If present, the `timestamp-precision` field **MUST** have a value of `year`, `month`, `day`, `hour`, `minute`, or `full`.
    - The default value for the precision field is `full`, so omitting the field is equivalent to explicitly specifying `full`.
    - A value of `full` indicates that the value in the `timestamp` field is precise to the full number of digits in the timestamp value (including any fractional seconds, such as milliseconds or microseconds).
    - A value of `year`, `month`, `day`, `hour`, or `minute` indicates that the timestamp value is precise to that as a lower bound (the precision window is the timestamp value plus one unit of the precision value).
        - *For example, if the timestamp value is `2016-04-25T13:00:00Z` and the precision value is `hour`, the time is greater than or equal to `2016-04-25T13:00:00Z` and less than `2016-04-25T14:00:00Z`.*
    - When specifying a precision other than `full`, the time portion of the `timestamp` field **MUST** contain `00` for all fields beyond the specified precision while the date portion **MUST** contain `01` for all fields beyond the specified precision.
        - *For example, if the precision field is `month`, the `timestamp` field must contain `01` for the day field and `00` for the hour, minute, and second fields such as `2016-12-01T00:00:00Z`.*
- The `timestamp-precision` property **MUST** always be at the same level as the `timestamp` property.
- The property name for the precision property **MUST** have the following suffix **`[timestamp_field_name]_precision`**.
    - *For example, if the key of the `timestamp` property is **`valid_from`**, the key of the precision field is **`valid_from_precision`**.*

### 2.10.2. Examples

**The following examples have explicitly defined the precision**

A timestamp known only to a year would look like:

```
{
  ...
  "start": "2016-01-01T00:00:00Z",
  "start_precision": "year",
  ...
}
```

A timestamp known only to an hour would look like:

```
{
  ...
  "end": "2016-01-20T12:00:00Z",
  "end_precision": "hour",
  ...
}
```

**The following examples have implicitly defined the precision**

A timestamp known to a second would look like:

```
{
  ...
  "start": "2016-01-20T12:31:12Z",
  ...
}
```

A timestamp known to 5-digit sub-second precision would look like:

```
{
  ...
  "end": "2016-01-20T12:31:12.12345Z",
  ...
}
```

## 2.11. Open Vocabulary

**Type Name:** `open-vocab`

The `open-vocab` type is represented as a `string`. For properties that use this type there will be a list of suggested values, known as the suggested vocabulary. The value of the property **SHOULD** be chosen from the suggested vocabulary but **MAY** be any other `string` value. Values that are not from the suggested vocabulary **SHOULD** be all lowercase (where lowercase is defined by the locality conventions) and **SHOULD** use dashes instead of spaces or underscores as word separators.

A consumer that receives STIX content with one or more `open-vocab` terms not defined in the suggested vocabulary **MAY** ignore those values.

The JSON MTI serialization uses the JSON string type <TODO: add reference> when representing `open-vocab`.

### 2.11.1. Examples

**Example using value from the suggested vocabulary**

In this example the Indicator **`labels`** property is an open vocabulary and we are using one of the suggested vocabulary values.

```
{
  ...,
  "labels": ["malicious-activity"],
  ...
}
```

**Example using a custom value**

In this example, for the same Indicator **`labels`** property, we are not using a value in the suggested vocabulary.

```
{
  ...,
  "labels": ["pbx-fraud-activity"],
  ...
}
```

# 3. STIX Objects

This section outlines the common properties and behavior across all SDOs and SROs.

The JSON MTI serialization uses the JSON object type <TODO: add reference> when representing all STIX Objects.

## 3.1. Common Properties

| Property Name | Type | Description |
|---|---|---|
| **`type`** (required) | `string` | The **`type`** property identifies the type of STIX Object. The value of the **`type`** field **MUST** be one of the types defined by a STIX Object (e.g., `indicator`). |
| **`id`** (required) | `identifier` | The **`id`** property universally and uniquely identifies this object. All objects with the same **`id`** are considered different versions of the same object.<br><br>Because the object type is part of the `identifier`, it is not possible for objects of different types to share the same **`id`**. |

|  |  |  |
|---|---|---|
| **`created_by_ref`** (optional) | `identifier` | The **`created_by_ref`** property specifies the ID of the Identity object that describes the entity that created this object.<br><br>If this attribute is omitted, the source of this information is undefined. This may be used by object creators who wish to remain anonymous. |
| **`created`** (required) | `timestamp` | The **`created`** property represents the time at which the first version of this object was created. The object creator can use the time it deems most appropriate as the time the object was created.<br><br>The **`created`** property **MUST** not be changed when creating a new version of the object.<br><br>The **`created`** property does not have a corresponding precision property and its precision **MUST** be treated as `full`.<br><br>See section TODO for further definition of versioning. |
| **`modified`** (required) | `timestamp` | The **`modified`** property represents the time that this particular version of the object was created. The object creator can use the time it deems most appropriate as the time this version of the object was modified. The value of the **`modified`** property for a given object version **MUST** be later than or equal to the value of the **`created`** property.<br><br>Object creators **MUST** update the **`modified`** property when creating a new version of an object.<br><br>The **`modified`** property does not have a corresponding precision property and its precision **MUST** be treated as `full`.<br><br>See section TODO for further definition of versioning. |
| **`version`** (required) | `number` | The **`version`** property indicates the |

| | | |
|---|---|---|
| | | version of this object. The value of this property **MUST** be an integer (whole number) greater than or equal to 1 and less than or equal to 999,999,999. Greater numbers indicate later versions of the object. Object creators **MUST** increase the version number (**SHOULD** increment it by exactly 1) when creating a new version of an object.<br><br>See section TODO for further definition of versioning. |
| **`revoked`** (optional) | `boolean` | The **`revoked`** property indicates whether the object has been revoked. Revoked objects are no longer considered valid by the object creator. Revoking an object is permanent; future versions of the object with this **`id`** **MUST NOT** be created.<br><br>The default value of this property is `false`.<br><br>See section TODO for further definition of versioning. |
| **`labels`** (optional) | `list` of type `string` | The **`labels`** property specifies a set of classifications.<br><br>The object definition of this property usually includes a suggested vocabulary and items in this list **SHOULD** come from that vocabulary. Additional labels **MAY** be added beyond what is in the suggested vocabulary. |
| **`external_references`** (optional) | `list` of type `external-reference` | The **`external_references`** property specifies a list of external references which refers to non-STIX information. This property is used to provide one or more URLs, descriptions, or IDs to records in other systems. |
| **`object_marking_refs`** (optional) | `list` of type `identifier` | The **`object_marking_refs`** property specifies a list of IDs of `marking-definition` objects that apply to this object. |

| | | |
|---|---|---|
| | | See the Data Markings in section TODO for further information. |
| **`granular_markings`** (optional) | `list` of type `granular-marking` | The **`granular_markings`** property specifies a list of granular markings applied to this object.<br><br>See the Data Markings in section [TODO Ref] for further information. |

## 3.2. IDs and References

The **`id`** property universally and uniquely identifies an SDO, SRO, Bundle, or Marking Definition. It **MUST** conform to the `identifier` type.

All STIX Objects (as well as Bundle and Marking Definition) use identifiers as defined by the `identifier` type. The `identifier` type is also used to define fields that are *ID references* to other constructs (such as the **`created_by_ref`** property in all STIX Objects). *Resolving* an ID reference is the process of identifying and obtaining the actual object referred to by the ID reference field. ID references resolve to an object when the value of the ID reference property (e.g., **`created_by_ref`**) is an exact match with the **`id`** property of another object. If a consumer has access to multiple versions of an object, the consumer **SHOULD** interpret any references to that object as referring to the latest version as defined in [REF: Section 3.4]. ID references **MAY** refer to objects to which the consumer/producer may not currently have. This specification does not address the implementation of ID reference resolution.

## 3.3. Object Creator

The object creator is the entity (e.g., system, organization, instance of a tool) that generates the **`id`** property for a given object. Object creators are represented as Identity objects. An embedded relationship to the Identity object representing the object creator is captured in the **`created_by_ref`** property.

Entities that re-publish an object from another entity without making any changes to the object, and thus maintaining the original **`id`**, are not considered the object creator and **MUST NOT** change the **`created_by_ref`** property. An entity that accepts objects and republishes them with modifications, additions, or omissions **MUST** create a new **`id`** for the object. They are considered the object creator of the new object for purposes of versioning.

## 3.4. Versioning

This section describes the versioning process and normative rules for performing versioning and revocation of STIX Objects. STIX Objects are versioned using the **`version`**, **`revoked`**, **`created`**, and **`modified`** properties. See the properties table in Section TODO [add reference] for full definitions and normative usage of those properties.

STIX Objects **MAY** be versioned in order to update, add, or remove information. A version of a STIX Object is identified uniquely by a combination of its **`id`** and **`version`** properties. Greater values of the **`version`** property indicate later versions of the object. Implementations **MUST** consider the version of the STIX Object with the highest version value to be the current state of the object. This specification does not address how implementations should handle versions of the object that are not current.

STIX Objects have a single *object creator*, the entity that generates the **`id`** for the object and creates the first version. Only the object creator is permitted to create new versions of a STIX Object. Producers other than the object creator **MUST NOT** create new versions of that object. If a producer other than the object creator wishes to create a new version, they **MUST** instead create a new object with a new **`id`**. They **SHOULD** additionally create a `derived-from` Relationship object to relate their new object to the original object that it was derived from.

Every representation (each time the object version is serialized and shared) of a version of an object (identified by the object's **`id`** and **`version`**) **MUST** always have the same set of properties and the same values for each property. In order to change the value of any property, or to add or remove properties, the **`version`** number **MUST** be increased to indicate a new version and the **`modified`** property **MUST** be updated to reflect the time of the change.

Objects can also be revoked, which is an indication 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 STIX Object. A value of `true` in the **`revoked`** property indicates that an object (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. The change to the **`revoked`** property to indicate that an object is revoked is an update to the object, and therefore its **`version`** and **`modified`** properties **MUST** be updated at the same time. This specification does not address how implementations should handle revoked data.

### 3.4.1. Versioning Timestamps

There are two timestamp properties used to indicate when STIX Objects were created and modified: **`created`** and **`modified`**. The **`created`** property indicates the time the first version of the object was created. The **`modified`** property indicates the time the specific version of the object was created. 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 **`created`** and **`modified`** properties.

### 3.4.2. New Version or New Object?

Eventually an implementation will encounter a case where a decision must be made regarding whether a change is a new version of an existing object or is different enough that it is a new object. 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 rules of thumb are provided. Any time a change indicates a *material change* to the meaning of the object, a new object with a different **`id`** should be used. A material change is any change that the object creator believes substantively changes the meaning of the object. As an example, an object creator might consider changing a Threat Actor from one country to another is a material change. These decisions are always made by the object creator. The object creator should also think about relationships to the object when deciding if a change is material. If the change would invalidate the usefulness of relationships to the object, then the change is considered material and a new object **`id`** should be used.

### 3.4.3. Examples

**Example of a new version**

One object creator has decided that the previous title they used for a SDO is incorrect. They consider that change as an update to the object.

| Step # | STIX Object | Object Creator Action |
|---|---|---|
| 1 | `{ "type": "example", "id": "example--1", "created": "2016-05-01T06:13:14.000000Z", "modified": "2016-05-01T06:13:14.000000Z", "version": 1, "title": "attention", "description": "this is the description" }` | Original version of an object is created. |
| 2 | N/A, STIX is not involved in this step | Object creator changes the title in their internal database. |
| 3 | `{ "type": "example", "id": "example--1", "created": "2016-05-01T06:13:14.000000Z", "modified": "2016-05-08T03:43:44.000000Z", "version": 2, "title": "Attention!", "description": "this is the description" }` | Object creator increases the **`version`** property by 1 and updates the **`modified`** property. |

**Example of derived object**
One object creator has decided that the previous title they used for a SDO is incorrect. They consider that change fundamental to the meaning of the object and therefore revoke the object and issue a new one.

| Step # | STIX Object | Object Creator Action |
|---|---|---|
| 1 | `{`<br>`  "type": "example",`<br>`  "id": "example--1",`<br>`  "created": "2016-05-01T06:13:14.000000Z",`<br>`  "modified": "2016-05-01T06:13:14.000000Z",`<br>`  "version": 1,`<br>`  "title": "attention",`<br>`  "description": "this is the description"`<br>`}` | Original object created (via new id and set **version** to *1*). |
| 2 | N/A, STIX is not involved in this step | Object creator changes the title in their internal database. |
| 3 | `{`<br>`  "type": "example",`<br>`  "id": "example--1",`<br>`  "created": "2016-05-01T06:13:14.000000Z",`<br>`  "modified": "2016-05-08T03:43:44.000000Z",`<br>`  "version": 2,`<br>`  "title": "attention",`<br>`  "description": "this is the description",`<br>`  "revoked": true`<br>`}` | Object creator revokes the existing object by setting **revoked** to `true`. The **`version`** is set to `2` and the **`modified`** property is updated. |
| 4 | `{`<br>`  "type": "example",`<br>`  "id": "example--2",`<br>`  "created": "2016-05-08T03:43:44.000000Z",`<br>`  "modified": "2016-05-08T03:43:44.000000Z",`<br>`  "version": 1,`<br>`  "title": "Something completely different",`<br>`  "description": "this is the description"`<br>`}` | Object creator creates a new object (with a new **`id`** and **`version`** set to `1`). |
| 5 | `{`<br>`  "type": "relationship",`<br>`  "id": "relationship--3",`<br>`  "created": "2016-05-08T03:43:44.000000Z",`<br>`  "modified": "2016-05-08T03:43:44.000000Z",`<br>`  "version": 1,`<br>`  "relationship_type": "derived-from",`<br>`  "source_ref": "example--1",`<br>`  "target_ref": "example--2"` | (Optional) Object creator creates a new Relationship indicating that the new object is derived from the old object. |

| | | |
|---|---|---|
| | `}` | |

**Example consumer workflow**

This section describes an example workflow where a consumer receives multiple updates to a particular object. (In this example, the STIX Objects have been truncated for brevity.)

| Step # | STIX Object | Recipient Action |
|---|---|---|
| 1 | `{`<br>`  "type": "example",`<br>`  "id": "example--1",`<br>`  "created": "2016-05-01T06:13:14.000000Z",`<br>`  "modified": "2016-05-01T06:13:14.000000Z",`<br>`  "version": 1`<br>`}` | Consumer stores example object because this is the first time they have seen the object. |
| 2 | `{`<br>`  "type": "example",`<br>`  "id": "example--1",`<br>`  "created": "2016-05-01T06:13:14.000000Z",`<br>`  "modified": "2016-05-08T03:43:44.000000Z",`<br>`  "version": 4`<br>`}` | Consumer updates example object because the received version number is greater than the object that is currently stored. |
| 3 | `{`<br>`  "type": "example",`<br>`  "id": "example--1",`<br>`  "created": "2016-05-01T06:13:14.000000Z",`<br>`  "modified": "2016-05-06T06:23:45.000000Z",`<br>`  "version": 3`<br>`}` | Consumer ignores this object because they already have a newer version of the object. Note: consumer might choose to store meta-information about received objects, including versions that were received out-of-order. The consumer also may choose to store a copy for reference. |
| 4 | `{`<br>`  "type": "example",`<br>`  "id": "example--1",`<br>`  "created": "2016-05-01T06:13:14.000000Z",`<br>`  "modified": "2016-05-11T06:41:21.000000Z",`<br>`  "version": 12,`<br>`  "revoked": true`<br>`}` | Consumer decides to delete example object, but keeps some metadata regarding the object. |
| 5 | `{`<br>`  "type": "example",`<br>`  "id": "example--1",`<br>`  "created": "2016-05-01T06:13:14.000000Z",`<br>`  "modified": "2016-05-10T17:28:54.000000Z",`<br>`  "version": 11` | Consumer ignores this object because they already have a newer version of the object (the revoked version). |

| | | |
|---|---|---|
| | `}` | |

**Example object creator workflow**

This section describes an example workflow where a object creator publishes multiple updates to a particular object. This scenario assumes a human using a STIX implementation. (In this example, the STIX Objects have been truncated for brevity.)

| Step # | STIX Object | User Action |
|---|---|---|
| 1 | N/A – STIX is not involved in this scenario.<br><br>(Tools *could* choose to create and track STIX versions for internal changes, but it is not required by the specification.) | User clicks a create button in the user interface, creates a SDO, then clicks save. This action causes information to be stored in the product’s database. |
| 2 | `{`<br>`  "type": "example",`<br>`  "id": "example--2",`<br>`  "created": "2016-05-01T06:13:14.000000Z",`<br>`  "modified": "2016-05-01T06:13:14.000000Z",`<br>`  "version": 1`<br>`}` | The user clicks the “share” button, delivering the intelligence to sharing partners. |
| 3 | N/A – STIX is not involved in this scenario.<br><br>(Tools *could* choose to create and track STIX versions for internal changes, but it is not required by the specification.) | The user performs additional analysis within the STIX implementation, performing multiple modifications and saving their work multiple times. |
| 4 | `{`<br>`  "type": "example",`<br>`  "id": "example--2",`<br>`  "created": "2016-05-01T06:13:14.000000Z",`<br>`  "modified": "2016-05-03T16:33:51.000000Z",`<br>`  "version": 2`<br>`}` | The user, happy with the status of their work, decides to provide an update to some properties of the previously published object (not shown). |
| 5 | `{`<br>`  "type": "example",`<br>`  "id": "example--2",`<br>`  "created": "2016-05-01T06:13:14.000000Z",`<br>`  "modified": "2016-05-08T13:35:12.000000Z"`<br>`  "version": 3,`<br>`  "revoked": true,`<br>`}` | The user receives lots of negative feedback regarding the quality of their work and decides to retract the object by pressing the “revoke” button. |

## 3.5. Common Relationships

Each SDO has its own set of relationships types that are specified in the definition of that SDO. The following common relationship types are defined for all SDOs. See Section <to do>[add reference] for more information about relationships.

| Relationship Type | Source | Target | Description |
|---|---|---|---|
| `derived-from` | *`<STIX Domain Object>`* | *`<STIX Domain Object of same type>`* | The information in the target object is based on information from the source object.<br><br>`derived-from` is an explicit relationship between two separate objects and **MUST NOT** be used as a substitute for the versioning process defined in section TODO. |
| `duplicate-of` | *`<STIX Domain Object>`* | *`<STIX Domain Object of same type>`* | The referenced source and target objects are semantically duplicates of each other.<br><br>This specification does not address whether the source or the target object is the duplicate object or what action, if any, a consumer should take when receiving an instance of this relationship.<br><br>As an example, a Campaign object from one organization could be marked as a `duplicate-of` a Campaign object from another organization if they both described the same campaign. |
| `related-to` | *`<STIX Domain Object>`* | *`<STIX Domain Object of any type>`* | Asserts a non-specific relationship between two SDOs. This relationship can be used when none of the other predefined relationships are appropriate.<br><br>As an example, a Malware object describing a piece of malware could be marked as a `related-to` a Tool if they are commonly used together. That relationship is not common enough to standardize on, but may |

|  |  |  |  |
|---|---|---|---|
|  |  |  | be useful to some analysts. |

## 3.6. Reserved Properties

This section defines property names that are reserved for future use in revisions of this document. The property names defined in this section **MUST NOT** be used for the name of any Custom Property.

Properties that are currently reserved across all STIX Objects are:

- `confidence`
- `severity`
- `action`
- `usernames`
- `phone_numbers`
- `addresses`

In addition, the following object names are reserved:

- `incident`
- `infrastructure`

# 4. Data Markings

Data markings represent restrictions, permissions, and other guidance for how data can be used and shared. For example, data may be shared with the restriction that it must not be re-shared, or that it must be encrypted at rest. In STIX, data markings are specified using the `marking-definition` object. These definitions are applied to complete STIX Objects using object markings and to individual properties of STIX Objects via granular markings.

Some types of marking definitions 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 object or property should be interpreted.

## 4.1. Marking Definition

**Type Name:** `marking-definition`

The `marking-definition` object represents a specific marking. Data markings typically represent handling or sharing requirements for data, and are applied in the **`object_marking_refs`** and **`granular_markings`** properties on STIX Objects, which reference a list of IDs for `marking-definition` objects.

Two marking definition types are defined in this specification: TLP, to capture TLP markings, and Statement, to capture text marking statements. In addition, it is expected that the FIRST Information Exchange Policy (IEP) will be included in a future version once a machine-usable specification for it has been defined.

The JSON MTI serialization uses the JSON object type <TODO: add reference> when representing `marking-definition`.

### 4.1.1. Properties

| Property Name | Type | Description |
|---|---|---|
| **`type`** (required) | `string` | The **`type`** property identifies the type object. The value of this property **MUST** be `marking-definition`. |
| **`id`** (required) | `identifier` | The **`id`** property universally and uniquely identifies this Marking Definition.<br><br>Because the object type is part of the `identifier`, it is not possible for objects of different types to share the same **`id`**. |
| **`created_by_ref`** (optional) | `identifier` | The **`created_by_ref`** property specifies the ID of the `identity` object that describes the entity that created this Marking Definition.<br><br>If this attribute is omitted, the source of this information is undefined. This may be used by object creators who wish to remain anonymous. |
| **`created`** (required) | `timestamp` | The **`created`** property represents the time at which the first version of this Marking Definition was created. The object creator can use the time it deems most appropriate as the time the object was created. |

| | | |
|---|---|---|
| **external_references** (optional) | list of type external-reference | The **external_references** property specifies a list of external references which refers to non-STIX information. This field is used to provide one or more URLs, descriptions, or IDs to records in other systems. |
| **object_marking_refs** (optional) | list of type identifier | The **object_marking_refs** property specifies a list of IDs of marking-definitions that apply to this Marking Definition. This property **MUST NOT** contain any references to this Marking Definition object (i.e., it cannot contain any circular references).<br><br>Though uncommon, in some cases marking definitions themselves may be marked with sharing or handling guidance. |
| **granular_markings** (optional) | list of type granular-marking | The **granular_markings** property specifies a list of granular markings applied to this. This property **MUST NOT** contain any references to this Marking Definition object (i.e., it cannot contain any circular references).<br><br>Though uncommon, in some cases Marking Definitions themselves may be marked with sharing or handling guidance. |
| **definition_type** (required) | open-vocab | The **definition_type** property identifies the type of Marking Definition. The value of the **definition_type** property **SHOULD** be one of the types defined in the subsections below: statement or tlp (REF:see Sections 4.1.3 and 4.1.4). |
| **definition** (required) | <marking object> | The **definition** property contains the marking object itself (e.g., the TLP marking as defined in Section TODO, the Statement marking as defined in Section TODO, or some |

| | | |
|---|---|---|
| | | other marking definition defined elsewhere). |

### 4.1.2. Relationships

There are no relationships explicitly defined between the Marking Definition object and other objects, other than those defined as common relationships. The first section lists the embedded relationships by property name along with their corresponding target.

Relationships are not restricted to those listed below. Relationships can be created between any objects using the `related-to` relationship name or, as with open vocabularies, user-defined names.

| **Embedded Relationships** | |
|---|---|
| **`created_by_ref`** | `identity` |
| **`object_marking_refs`** | `marking-definition` |
| **Common Relationships** | |
| `duplicate-of`, `derived-from`, `related-to` | |

### 4.1.3. Statement Marking Object Type

The Statement marking type defines the representation of a textual marking statement (e.g., copyright, terms of use, etc.) in a definition. The value of the **`definition_type`** property **MUST** be `statement` when using this marking type. Statement markings are generally not machine-readable and this specification does not define any behavior or actions based on their values.

Content may be marked with multiple Statement marking types that do not override each other. In other words, the same content can be marked both with a statement saying "Copyright 2016" and a statement saying "Terms of use are ..." and both statements apply.

| **Property Name** | **Type** | **Description** |
|---|---|---|
| **`statement`** (required) | `string` | A Statement (e.g., copyright, terms of use) applied to the content marked by this marking definition. |

#### 4.1.3.1. Examples

```
{
  "type": "marking-definition",
  "id": "marking-definition--34098fce-860f-48ae-8e50-ebd3cc5e41da",
  "created": "2016-08-01T00:00:00Z",
  "definition_type": "statement",
  "definition": {
    "statement": "Copyright 2016, Example Corp"
  }
}
```

### 4.1.4. TLP Marking Object Type

The TLP marking type defines how you would represent a Traffic Light Protocol (TLP) marking in a definition field. The value of the **`definition_type`** property **MUST** be `tlp` when using this marking type.

| Property Name | Type | Description |
|---|---|---|
| **`tlp`** (required) | `string` | The TLP level (defined by FIRST, ask Tom Millar for stable ref) of the content marked by this marking definition, as defined in this section. |

The following standard marking definitions **MUST** be used to reference or represent TLP markings. Other instances of `tlp-marking` **MUST NOT** be used (the only instances of TLP marking definitions permitted are those defined here).

| | |
|---|---|
| `white` | `{`<br>`  "type": "marking-definition",`<br>`  "id": "marking-definition--613f2e26-407d-48c7-9eca-b8e91df99dc9",`<br>`  "created": "2016-08-01T00:00:00Z",`<br>`  "definition_type": "tlp",`<br>`  "definition": {`<br>`    "tlp": "white"`<br>`  }`<br>`}` |
| `green` | `{`<br>`  "type": "marking-definition",`<br>`  "id": "marking-definition--34098fce-860f-48ae-8e50-ebd3cc5e41da",`<br>`  "created": "2016-08-01T00:00:00Z",`<br>`  "definition_type": "tlp",`<br>`  "definition": {`<br>`    "tlp": "green"`<br>`  }`<br>`}` |

| | |
|---|---|
| amber | `{`<br>`  "type": "marking-definition",`<br>`  "id": "marking-definition--f88d31f6-486f-44da-b317-01333bde0b82",`<br>`  "created": "2016-08-01T00:00:00Z",`<br>`  "definition_type": "tlp",`<br>`  "definition": {`<br>`    "tlp": "amber"`<br>`  }`<br>`}` |
| red | `{`<br>`  "type": "marking-definition",`<br>`  "id": "marking-definition--5e57c739-391a-4eb3-b6be-7d15ca92d5ed",`<br>`  "created": "2016-08-01T00:00:00Z",`<br>`  "definition_type": "tlp",`<br>`  "definition": {`<br>`    "tlp": "red"`<br>`  }`<br>`}` |

## 4.2. Object Markings

Object Markings apply data markings to an entire STIX Object or Marking Definition and all of its contents. Object Markings are specified as embedded relationships in the **object_marking_refs** property, which is an optional list of IDs for `marking-definition` objects. The referenced markings apply to that STIX Object or Marking Definition and all of its contents. Changes to the **object_marking_refs** property (and therefore the markings applied to the object) are treated the same as changes to any other properties on the object and follow the same rules for versioning.

### 4.2.1. Examples

This example marks the Indicator and all its properties with the Marking Definition referenced by the ID.

```
{
  "type": "indicator",
  "id": "indicator--b346b4b3-f4b7-4235-b659-f985f65f0009",
  ...
  "object_marking_refs": ["marking-definition--089a6ecb-cc15-43cc-9494-767639779123"],
  ...
}
```

## 4.3. Granular Markings

Whereas object markings apply to an entire STIX Object or Marking Definition and all its properties, granular markings allow data markings to be applied to individual portions of STIX Objects and Marking Definitions. Granular markings are specified in the **granular_markings** property, which is a list of `granular-marking` instances. Each of those instances contains a list of selectors to indicate what is marked and a reference to the `marking-definition` object to be applied. Granular markings can be used, for example, to indicate that the **`name`** property of an `indicator` should be handled as TLP:GREEN, the **`description`** property as TLP:AMBER, and the **`pattern`** property as TLP:RED.

### 4.3.1. Granular Marking Type

The `granular-marking` type defines how the `marking-definition` object referenced by the **`marking_ref`** property applies to a set of content identified by the list of selectors in the **`selectors`** property.

| Property Name | Type | Description |
|---|---|---|
| **`marking_ref`** (required) | `identifier` | The **`marking_ref`** property specifies the ID of the `marking-definition` object that describes the marking. |
| **`selectors`** (required) | `list` of type `string` | The **`selectors`** property specifies a list of selectors for content contained within the STIX Object in which this property appears. Selectors **MUST** conform to the syntax defined in [REF:Section 4.3.1.1].<br><br>The `marking-definition` referenced in the **`marking_ref`** field is applied to the content selected by the selectors in this list. |

#### 4.3.1.1. Selector Syntax

Selectors contained in the **`selectors`** list are strings that consist of multiple components that **MUST** be separated by the `.` character. Each component **MUST** be one of:

- A property name, e.g., `description`, or;
- A zero-based list index, specified as a non-negative integer in square brackets, e.g., `[4]`

Selectors denote path traversals: the root of each selector is the STIX Object that the **`granular_markings`** field appears in. Starting from that root, for each component in the selector, properties and list items are traversed. When the complete list has been traversed, the value of the content is considered selected.

Selectors **MUST** refer to properties or list items that are actually present on the marked object.

As an example, consider the following STIX Object:

```
{
  "id": "vulnerability--ee916c28-c7a4-4d0d-ad56-a8d357f89fef",
  "created": "2016-02-14T00:00:00Z",
  "modified": "2016-02-14T00:00:00Z",
  "version": 1,
  "type": "vulnerability",
  "name": "CVE-2014-0160",
  "description": "The (1) TLS...",
  "external_references": [{
    "source_name": "cve",
    "external_id": "CVE-2014-0160"
  }],
  "labels": ["heartbleed", "has-logo"]
}
```

Valid selectors:

- `description` selects the **`description`** property ("`The (1) TLS...`").
- `external_references.[0].source_name` selects the **`source_name`** property of the first value of the **`external_references`** list ("`cve`").
- `labels.[0]` selects the first item contained within the **`labels`** list ("`heartbleed`").
- `labels` selects the list contained in the **`labels`** property. Due to the recursive nature of the selector, that includes all items in the list `(["heartbleed", "has-logo"])`.
- `external_references` selects the list contained in the **`external_references`** property. Due to the recursive nature of the selector, that includes all list items and all properties of those list items.

Invalid selectors:

- `pattern` and `external_references.[3]` are invalid selectors because they refer to content not present in that object.
- `description.[0]` is an invalid selector because the `description` property is a string and not a list.
- `labels.name` is an invalid selector because `labels` property is a list and not an object.

This syntax is inspired by JSONPath (REF: http://goessner.net/articles/JsonPath/ ) and is in fact a strict subset of allowable JSONPath expressions (with the exception that the '$' to indicate the root is implicit). Care should be taken when passing selectors to JSONPath evaluators to ensure that the root of the query is the individual STIX Object. It is expected, however, that selectors can be easily evaluated in programming languages that implement list and key/value mapping types (dictionaries, hashmaps, etc.) without resorting to an external library.

### 4.3.2. Example

This example marks the **`description`** and **`labels`** properties with the single marking definition referenced in the list.

```
{
  ...
  "granular_markings": [
    {
      "marking_ref": "marking-definition--089a6ecb-cc15-43cc-9494-767639779123",
      "selectors": ["description", "labels"]
    }
  ],
  "description": "Some description"
  "title": "Some title",
  "labels": ["first", "second"]
}
```

# 5. STIX Domain Objects

This specification defines the set of STIX Domain Objects (SDOs), each of which corresponds to a unique concept commonly represented in CTI. Using SDOs and STIX relationships as building blocks, individuals can create and share broad and comprehensive cyber threat intelligence.

Property information, relationship information, and examples are provided for each SDO defined below. Property information includes common properties as well as properties that are specific to each SDO. Relationship information includes embedded relationships (e.g., **`created_by_ref`**), common relationships (e.g., `related-to`), and SDO-specific relationships. Forward relationships (i.e., relationships *from* the SDO to other SDOs) are fully defined, while reverse relationships (i.e., relationships *to* the SDO from other SDOs) are duplicated for convenience.

Some SDOs are similar and can be grouped together into categories. Attack Pattern, Malicious, Malware, and Tool can all be considered types of tactics, techniques, and procedures (TTPs): they describe behaviors and resources that attackers use to carry out their attacks. Similarly, Campaign, Intrusion Set, and Threat Actor all describe information about why adversaries carry out attacks and how they organize themselves.

## 5.1. Attack Pattern

**Type Name:** `attack-pattern`

Attack Patterns are a type of TTP that describe ways that adversaries attempt to compromise targets. Attack Patterns are used to help categorize attacks, generalize specific attacks to the patterns that they follow, and provide detailed information about how attacks are performed. An example of an attack pattern is "spear phishing": a common type of attack where an attacker sends a carefully crafted e-mail message to a party with the intent of getting them to click a link or open an attachment to deliver malware. Attack Patterns can also be more specific: spear phishing as practiced by a particular threat actor (i.e., they might generally say that the target won a contest) can also be an Attack Pattern.

The Attack Pattern SDO contains textual descriptions of the pattern along with references to externally-defined taxonomies of attacks such as CAPEC <TODO: need reference>. Relationships from Attack Pattern can be used to relate it to what it targets (Vulnerabilities and Identities) and which tools and malware use it (Tool and Malware).

### 5.1.1. Properties

| **Common Properties** | | |
|---|---|---|
| **`type, id, created_by_ref, created, modified, version, revoked, labels, external_references, object_marking_refs, granular_markings`** | | |
| **Attack Pattern Specific Properties** | | |
| **`name, description, kill_chain_phases`** | | |
| **Property Name** | **Type** | **Description** |
| **`type`** (required) | `string` | The value of this field **MUST** be `attack-pattern` |
| **`external_references`** (optional) | `list` of type `external-reference` | A list of external references which refer to non-STIX information. This field **MAY** be used to provide one or more Attack Pattern identifiers, such as a CAPEC ID. When specifying a CAPEC ID, the **`source_name`** field of the external reference **MUST** be set to `capec` and the **`external_id`** field **MUST** be formatted as `CAPEC-[id]`. |
| **`name`** (required) | `string` | A name used to identify the Attack Pattern. |
| **`description`** (optional) | `string` | A description that provides more details and context about the Attack Pattern, potentially including its purpose and its key characteristics. |
| **`kill_chain_phases`** (optional) | `list` of type `kill-chain-phase` | The list of Kill Chain Phases for which this Attack Pattern is used. |

### 5.1.2. Relationships

These are the relationships explicitly defined between the Attack Pattern object and other SDOs. The first section lists the embedded relationships by property name along with their corresponding target. The rest of the table identifies the relationships that can be made from the Attack Pattern object by way of the Relationship object. The reverse relationships (relationships "to" the Attack Pattern object) are included as a convenience. For their definitions, please see the objects for which they represent a "from" relationship.

Relationships are not restricted to those listed below. Relationship objects can be created between any SDOs using the `related-to` relationship type or, as with open vocabularies, user-defined names.

| **Embedded Relationships** | | | |
|---|---|---|---|
| **`created_by_ref`** | `identity` | | |
| **`object_marking_refs`** | `marking-definition` | | |
| **Common Relationships** | | | |
| `duplicate-of`, `derived-from`, `related-to` | | | |
| **Source** | **Relationship Type** | **Target** | **Description** |
| `attack-pattern` | `targets` | `identity`, `vulnerability` | This Relationship describes that this Attack Pattern typically targets the type of victims or vulnerability represented by the related Identity or Vulnerability object.<br><br>For example, a `targets` Relationship linking an Attack Pattern for SQL injection to a Identity object representing domain administrators means that the form of SQL injection characterized by the Attack Pattern targets domain administrators in order to achieve its objectives.<br><br>Another example is a Relationship linking an Attack Pattern for SQL injection to a Vulnerability in blogging software means that the particular SQL injection attack exploits that vulnerability. |
| `attack-pattern` | `uses` | `malware`, `tool` | This Relationship describes that the |

|  |  |  |  |
|---|---|---|---|
|  |  |  | related Malware or Tool is used to perform the behavior identified in the Attack Pattern.<br><br>For example, a `uses` Relationship linking an Attack Pattern for a distributed denial of service (DDoS) to a Tool for Low Orbit Ion Cannon (LOIC) indicates that the tool can be used to perform those DDoS attacks. |
| **Reverse Relationships** |  |  |  |
| `indicator` | `indicates` | `attack-pattern` | See forward relationship for definition. |
| `course-of-action` | `mitigates` | `attack-pattern` | See forward relationship for definition. |
| `campaign`, `intrusion-set`, `threat-actor` | `uses` | `attack-pattern` | See forward relationship for definition. |

### 5.1.3. Examples

A generic attack pattern for spear phishing, referencing CAPEC

```
{
  "type": "attack-pattern",
  "id": "attack-pattern--0c7b5b88-8ff7-4a4d-aa9d-feb398cd0061",
  "created": "2016-05-12T08:17:27.000000Z",
  "modified": "2016-05-12T08:17:27.000000Z",
  "version": 1,
  "name": "Spear Phishing",
  "description": "...",
  "external_references": [
    {
      "source_name": "capec",
      "id": "CAPEC-163"
    }
  ]
}
```

A specific attack pattern for a particular form of spear phishing, referencing CAPEC

```
[
{
  "type": "attack-pattern",
  "id": "attack-pattern--7e33a43e-e34b-40ec-89da-36c9bb2cacd5",
  "created": "2016-05-12T08:17:27.000000Z",
  "modified": "2016-05-12T08:17:27.000000Z",
  "version": 1,
  "name": "Spear Phishing as Practiced by Adversary X",
  "description": "A particular form of spear phishing where the attacker claims that the
target had won a contest, including personal details, to get them to click on a link.",
  "external_references": [
    {
      "source_name": "capec",
      "id": "CAPEC-163"
    }
  ]
},
{
  "type": "relationship",
  "id": "relationship--57b56a43-b8b0-4cba-9deb-34e3e1faed9e",
  "created": "2016-05-12T08:17:27.000000Z",
  "modified": "2016-05-12T08:17:27.000000Z",
  "version": 1,
  "relationship_type": "uses",
  "source_ref": "intrusion-set--0c7e22ad-b099-4dc3-b0df-2ea3f49ae2e6",
  "target_ref": "attack-pattern--7e33a43e-e34b-40ec-89da-36c9bb2cacd5"
},
{
  "type": "intrusion-set",
  "id": "intrusion-set--0c7e22ad-b099-4dc3-b0df-2ea3f49ae2e6",
  "created": "2016-05-12T08:17:27.000000Z",
  "modified": "2016-05-12T08:17:27.000000Z",
  "version": 1,
  "name": "Adversary X"
}
]
```

## 5.2. Campaign

**Type Name:** `campaign`

A Campaign is a grouping of adversarial behaviors that describes a set of malicious activities or attacks (sometimes called waves) that occur over a period of time against a specific set of targets. Campaigns usually have well defined objectives and may be part of an Intrusion Set.

Campaigns are often attributed to an Intrusion Set and Threat Actors. The threat actors may reuse known infrastructure from the Intrusion Set or may set up new infrastructure specific for conducting that campaign.

Campaigns can be characterized by their objectives and the incidents they cause, people or resources they target, and the resources (infrastructure, intelligence, Malware, Tools, etc) they use.

For example, a Campaign could be used to describe a crime syndicate's attack using a specific variant of malware and new C2 servers against the executives of ACME Bank during the summer of 2016 in order to gain secret information about an upcoming merger with another bank.

### 5.2.1. Properties

| **Common Properties** | | |
|---|---|---|
| **`type, id, created_by_ref, created, modified, version, revoked, labels, external_references, object_marking_refs, granular_markings`** | | |
| **Campaign Specific Properties** | | |
| **`name, description, aliases, first_seen, first_seen_precision, objective`** | | |
| **Property Name** | **Type** | **Description** |
| **`type`** (required) | `string` | The value of this field **MUST** be `campaign` |
| **`name`** (required) | `string` | A name used to identify the Campaign. |
| **`description`** (optional) | `string` | A description that provides more details and context about the Campaign, potentially including its purpose and its key characteristics. |
| **`aliases`** (optional) | `list` of type `string` | Alternative names used to identify this Campaign |
| **`first_seen`** (optional) | `timestamp` | The time that his Campaign was first seen. |
| **`first_seen_precision`** (optional) | `timestamp-precision` | The precision of the **`first_seen`** timestamp. |
| **`objective`** (optional) | `string` | This field defines the Campaign’s primary goal, objective, desired outcome, or intended effect — what the Threat Actor hopes to accomplish with this Campaign. |

## 5.2.2. Relationships

These are the relationships explicitly defined between the Campaign object and other objects. The first section lists the embedded relationships by property name along with their corresponding target. The rest of the table identifies the relationships that can be made from the Campaign object by way of the Relationship object. The reverse relationships (relationships "to" the Campaign object) are included as a convenience. For their definitions, please see the objects for which they represent a "from" relationship.

Relationships are not restricted to those listed below. Relationships can be created between any objects using the `related-to` relationship type or, as with open vocabularies, user-defined names.

| | | | |
|---|---|---|---|
| **Embedded Relationships** | | | |
| **`created_by_ref`** | `identity` | | |
| **`object_marking_refs`** | `marking-definition` | | |
| **Common Relationships** | | | |
| `duplicate-of`, `derived-from`, `related-to` | | | |
| **Source** | **Relationship Type** | **Target** | **Description** |
| `campaign` | `attributed-to` | `intrusion-set`, `threat-actor` | This Relationship describes that the Intrusion Set or Threat Actor that is involved in carrying out the Campaign.<br><br>For example, an `attributed-to` Relationship from the Glass Gazelle Campaign to the Urban Fowl Threat Actor means that the actor carried out or was involved in some of the activity described by the Campaign. |
| `campaign` | `targets` | `identity`, `vulnerability` | This Relationship describes that the Campaign uses exploits of the related Vulnerability or targets the type of victims described by the related Identity.<br><br>For example, a `targets` Relationship from the Glass Gazelle Campaign to a Vulnerability in a blogging platform indicates that attacks performed as part of Glass Gazelle often exploit that Vulnerability.<br><br>Similarly, a `targets` Relationship from the Glass Gazelle Campaign to a |

| | | | |
|---|---|---|---|
| | | | Identity describing the energy sector in the United States means that the Campaign typically carries out attacks against targets in that sector. |
| `campaign` | `uses` | `attack-pattern`, `malware`, `tool` | This Relationship describes that attacks carried out as part of the Campaign typically use the related Attack Pattern, Malware, or Tool.<br><br>For example, a `uses` Relationship from the Glass Gazelle Campaign to the xInject Malware indicate that xInject is often used during attacks attributed to that Campaign. |
| **Reverse Relationships** | | | |
| `indicator` | `indicates` | `campaign` | See forward relationship for definition. |

### 5.2.3. Examples

```
{
  "type": "campaign",
  "id": "campaign--8e2e2d2b-17d4-4cbf-938f-98ee46b3cd3f",
  "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
  "created": "2016-04-06T20:03:48Z",
  "modified": "2016-04-06T20:03:48Z",
  "version": 1,
  "name": "Green Group Attacks Against Finance",
  "description": "Campaign by Green Group against a series of targets in the financial
services sector."
}
```

## 5.3. Course of Action

**Type Name:** `course-of-action`

**Note: The Course of Action object in STIX 2.0 is a stub. It is included to support basic use cases (such as sharing prose courses of action) but does not support the ability to represent automated courses of action or contain fields to represent metadata about courses of action. Future STIX 2 releases will expand it to include these capabilities.**

A Course of Action is an action taken either to prevent an attack or to respond to an attack that is in progress. It may describe technical, automatable responses (applying patches, reconfiguring firewalls) but can also describe higher level actions like employee training or policy changes. For example, a course of action to mitigate a vulnerability could describe applying the patch that fixes it.

The Course of Action SDO contains a textual description of the action; a reserved **`action`** field also serves as placeholder for future inclusion of machine automatable courses of action. Relationships from the Course of Action can be used to link it to the Vulnerabilities or behaviors (Tool, Malware, Attack Pattern) that it mitigates.

### 5.3.1. Properties

| **Common Properties** | | |
|---|---|---|
| **`type, id, created_by_ref, created, modified, version, revoked, labels, external_references, object_marking_refs, granular_markings`** | | |
| **Course of Action Specific Properties** | | |
| **`name, description, action`** | | |
| **Property Name** | **Type** | **Description** |
| **`type`** (required) | `string` | The value of this field **MUST** be `course-of-action` |
| **`name`** (required) | `string` | A name used to identify the Course of Action |
| **`description`** (optional) | `string` | A description that provides more details and context about the Course of Action, potentially including its purpose and its key characteristics. |
| **`action`** (reserved) | `RESERVED` | RESERVED – To capture structured/automated courses of action. |

### 5.3.2. Relationships

These are the relationships explicitly defined between the Course of Action object and other objects. The first section lists the embedded relationships by property name along with their corresponding target. The rest of the table identifies the relationships that can be made from the Course of Action object by way of the Relationship object. The reverse relationships (relationships "to" the Course of Action object) are included as a convenience. For their definitions, please see the objects for which they represent a "from" relationship.

Relationships are not restricted to those listed below. Relationships can be created between any objects using the `related-to` relationship type or, as with open vocabularies, user-defined names.

| **Embedded Relationships** | | | |
|---|---|---|---|
| **`created_by_ref`** | `identity` | | |
| **`object_marking_refs`** | `marking-definition` | | |
| **Common Relationships** | | | |
| `duplicate-of`, `derived-from`, `related-to` | | | |
| **Source** | **Relationship Type** | **Target** | **Description** |
| `course-of-action` | `mitigates` | `attack-pattern`, `malware`, `tool`, `vulnerability` | This Relationship describes that the Course of Action can mitigate the related Attack Pattern, Malware, Vulnerability, or Tool.<br><br>For example, a `mitigates` Relationship from a Course of Action to block an IP address to the xInject Malware indicates that the Course of Action mitigates the impact of the xInject. |
| **Reverse Relationships** | | | |
| — | — | — | — |

## 5.3.3. Examples

```
[
  {
    "type": "course-of-action",
    "id": "course-of-action--8e2e2d2b-17d4-4cbf-938f-98ee46b3cd3f",
    "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
    "created": "2016-04-06T20:03:48Z",
    "modified": "2016-04-06T20:03:48Z",
    "version": 1,
    "name": "Add TCP port 80 Filter Rule to the existing Block UDP 1434 Filter",
    "description": "This is how to add a filter rule to block inbound access to TCP port 80
the existing UDP 1434 filter ..."
  },
  {
    "type": "relationship",
    "id": "relationship--44298a74-ba52-4f0c-87a3-1824e67d7fad",
    "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
    "created": "2016-04-06T20:06:37Z",
    "modified": "2016-04-06T20:06:37Z",
    "version": 1,
    "source_ref": "course-of-action--8e2e2d2b-17d4-4cbf-938f-98ee46b3cd3f",
    "target_ref": "malware--31b940d4-6f7f-459a-80ea-9c1f17b5891b",
    "relationship_type": "mitigates"
  },
  {
    "type": "malware",
    "id": "malware--31b940d4-6f7f-459a-80ea-9c1f17b5891b",
    "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
    "created": "2016-04-06T20:07:09Z",
    "modified": "2016-04-06T20:07:09Z",
    "version": 1,
    "relationship_type": "Poison Ivy"
  }
]
```

## 5.4. Identity

**Type Name:** `identity`

Identities can represent actual individuals, organizations, or groups (e.g., ACME, Inc.) as well as classes of individuals, organizations, or groups (e.g., the finance sector in North America).

The Identity SDO can capture basic identifying information, contact information, and the sectors and regions that the Identity belongs to. Identity is used in STIX to represent, among other things, targets of attacks, information sources, object creators, and threat actor identities.

### 5.4.1. Properties

| Common Properties | | |
|---|---|---|
| **type, id, created_by_ref, created, modified, version, revoked, labels, external_references, object_marking_refs, granular_markings** | | |
| **Source Specific Properties** | | |
| **name, description, identity_class, sectors, regions, nationalities, contact_information** | | |
| **Property Name** | **Type** | **Description** |

| | | |
|---|---|---|
| `type` (required) | `string` | The value of this field **MUST** be `identity` |
| `labels` (optional) | `list` of type `string` | The list of roles that this Identity performs (e.g., CEO, Domain Administrators, Doctors, Hospital, or Retailer). No open vocabulary is yet defined for this property. |
| `name` (required) | `string` | The name of this Identity. When referring to a specific entity (e.g., an individual or organization), this field **SHOULD** contain the canonical name of the specific entity. |
| `description` (optional) | `string` | A description that provides more details and context about the Identity, potentially including its purpose and its key characteristics. |
| `identity_class` (required) | `open-vocab` | The type of entity that this Identity describes, e.g., an individual or organization.<br><br>This is an open vocabulary and the values **SHOULD** come from the `identity-class-ov` vocabulary. |
| `sectors` (optional) | `list` of type `open-vocab` | The list of industry sectors that this Identity belongs to.<br><br>This is an open vocabulary and values **SHOULD** come from the `industry-sector-ov` vocabulary. |
| `regions` (optional) | `list` of type `string` | The list of regions or geographic locations this Identity is located or operates in. |
| `nationalities` (optional) | `list` of type `string` | The nationalities of this Identity. The value **MUST** be from the ISO 3166-1 Alpha-2 codes and represented in lowercase <TODO ISO Ref>. |
| `contact_information` (optional) | `string` | The contact information (e-mail, phone number, etc.) for this Identity. No format for this information is currently defined by this specification. |

### 5.4.2. Relationships

There is an embedded relationship to Identity in all STIX Objects called **`created_by_ref`** that is inherited from the Common Properties. This property links each object with the Identity of the organization or individual that created the object.

These are the relationships explicitly defined between the Identity object and other objects. The first section lists the embedded relationships by property name along with their corresponding target. The rest of the table identifies the relationships that can be made from the Identity object by way of the Relationship object. None are defined for the Identity object. The reverse relationships (relationships "to" the Identity object) are included as a convenience. For their definitions, please see the objects for which they represent a "from" relationship.

Relationships are not restricted to those listed below. Relationships can be created between any objects using the `related-to` relationship type or, as with open vocabularies, user-defined names.

| **Embedded Relationships** | | | |
|---|---|---|---|
| **`created_by_ref`** | `identity` | | |
| **`object_marking_refs`** | `marking-definition` | | |
| **Common Relationships** | | | |
| `duplicate-of`, `derived-from`, `related-to` | | | |
| **Source** | **Relationship Type** | **Target** | **Description** |
| — | — | — | — |
| **Reverse Relationships** | | | |
| `attack-pattern`, `campaign`, `intrusion-set`, `malware`, `threat-actor`, `tool` | `targets` | `identity` | See forward relationship for definition. |
| `threat-actor` | `attributed-to`, `impersonates` | `identity` | See forward relationship for definition. |

### 5.4.3. Examples

An Identity for an individual named John Smith

```
{
  "type": "identity",
  ...,
  "name": "John Smith",
  "identity_class": "individual",
  ...
}
```

An Identity for a company named ACME Widget, Inc.

```
{
  "type": "identity",
  ...,
  "name": "ACME Widget, Inc.",
  "identity_class": "organization",
  ...
}
```

## 5.5. Indicator

**Type Name:** `indicator`

Indicators contain a pattern that can be used to detect suspicious or malicious cyber activity. For example, an Indicator may be used to represent a set of malicious domains and use the CybOX Patterning Language (TODO add reference) to specify these domains.

The Indicator SDO contains a simple textual description, the Kill Chain Phases that it detects behavior in, a time window for when the Indicator is valid or useful, and a required **`pattern`** property to capture a structured detection pattern. The **`pattern`** property can contain a detection pattern specified in either the CybOX Patterning Language (the default) or other patterning languages, such as Snort (REF:https://www.snort.org/) and YARA (REF:http://virustotal.github.io/yara/)). Conforming STIX implementations **MUST** support the CybOX Patterning Language <TODO: add reference> and **MAY** additionally support other pattern languages. While each structured pattern language has different syntax and potentially different  semantics, in general an Indicator is considered to have "matched" (or been "sighted") when the conditions specified in the structured pattern are satisfied in whatever context they are evaluated in.

Relationships from the Indicator can describe the malicious or suspicious behavior that it directly detects (Malware, Tool, and Attack Pattern) as well as the Campaigns, Intrusion Sets, and Threat Actors that it might indicate the presence of.

### 5.5.1. Properties

| **Common Properties** | | |
|---|---|---|
| `type`, `id`, `created_by_ref`, `created`, `modified`, `version`, `revoked`, `labels`, `external_references`, `object_marking_refs`, `granular_markings` | | |
| **Indicator Specific Properties** | | |
| `name`, `description`, `pattern`, `pattern_lang`, `pattern_lang_version`, `valid_from`, `valid_from_precision`, `valid_until`, `valid_until_precision`, `kill_chain_phases` | | |
| **Property Name** | **Type** | **Description** |
| **`type`** (required) | `string` | The value of this field **MUST** be `indicator` |
| **`labels`** (required) | `list` of type `open-vocab` | This field is an Open Vocabulary that specifies the type of indicator.<br><br>This is an open vocabulary and values **SHOULD** come from the `indicator-label-ov` vocabulary. |
| **`name`** (optional) | `string` | A name used to identify the Indicator. |
| **`description`** (optional) | `string` | A description that provides more details and context about the Indicator, potentially including its purpose and its key characteristics. |
| **`pattern_lang`** (optional) | `open-vocab` | The language used to define the pattern (in the **`pattern`** field). The default is `cybox` if the field is omitted.<br><br>This is an open vocabulary and values **SHOULD** come from the `pattern-lang-ov` vocabulary. |
| **`pattern_lang_version`** (optional) | `string` | The version of the language used to define the pattern (in the **`pattern`** field). There is no default.<br><br>The only CybOX patterning language version permitted is 1.0. Therefore, if the CybOX pattern |

| | | |
|---|---|---|
| | | language is used, the field **MUST** be omitted or **MUST** be `1.0`. |
| **`pattern`** (required) | `string` | The detection pattern for this Indicator. The default language is CybOX Patterning; implementations **MUST** support processing of CybOX patterns and **MAY** support others, such as Snort and YARA. |
| **`valid_from`** (required) | `timestamp` | The time from which this Indicator should be considered valuable intelligence. |
| **`valid_from_precision`** (optional) | `timestamp-precision` | The precision of the start timestamp. |
| **`valid_until`** (optional) | `timestamp` | The time at which this Indicator should no longer be considered valuable intelligence.<br><br>If the **`valid_until`** property is omitted, then there is no constraint on the latest time for which the Indicator should be used. |
| **`valid_until_precision`** (optional) | `timestamp-precision` | The precision of the valid until timestamp. |
| **`kill_chain_phases`** (optional) | `list` of type `kill-chain-phase` | The phases of the kill chain that this Indicator detects. <todo: Fix this definition.> |

### 5.5.2. Relationships

These are the relationships explicitly defined between the Indicator object and other objects. The first section lists the embedded relationships by property name along with their corresponding target. The rest of the table identifies the relationships that can be made from the Indicator object by way of the Relationship object. The reverse relationships (relationships "to" the Indicator object) are included as a convenience. For their definitions, please see the objects for which they represent a "from" relationship.

Relationships are not restricted to those listed below. Relationships can be created between any objects using the `related-to` relationship type or, as with open vocabularies, user-defined names.

| **Embedded Relationships** | | | |
|---|---|---|---|
| `created_by_ref` | `identity` | | |
| `object_marking_refs` | `marking-definition` | | |
| **Common Relationships** | | | |
| `duplicate-of`, `derived-from`, `related-to` | | | |
| **Source** | **Relationship Type** | **Target** | **Description** |
| `indicator` | `indicates` | `attack-pattern`, `campaign`, `intrusion-set`, `malware`, `threat-actor`, `tool` | This Relationship describes that the Indicator can detect evidence of the related Campaign, Intrusion, or Threat Actor. This evidence may not be direct: for example, the Indicator may detect secondary evidence of the Campaign, such as malware or behavior commonly used by that Campaign.<br><br>For example, an `indicates` Relationship from an Indicator to a Campaign object representing Glass Gazelle means that the Indicator is capable of detecting evidence of Glass Gazelle, such as command and control IPs commonly used by that Campaign. |
| **Reverse Relationships** | | | |
| — | — | — | — |

### 5.5.3. Examples

Indicator itself, with context

```
[
  {
    "type": "indicator",
    "id": "indicator--8e2e2d2b-17d4-4cbf-938f-98ee46b3cd3f",
    "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
    "created": "2016-04-06T20:03:48Z",
    "modified": "2016-04-06T20:03:48Z",
    "version": 1,
    "name": "Poison Ivy Malware",
    "description": "This file is part of Poison Ivy",
    "pattern": "file-object.hashes.md5 = '3773a88f65a5e780c8dff9cdc3a056f3'",
    "pattern_lang": "cybox",
    "valid_from": "2016-01-01T00:00:00Z"
  },
  {
    "type": "relationship",
    "id": "relationship--44298a74-ba52-4f0c-87a3-1824e67d7fad",
    "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
    "created": "2016-04-06T20:06:37Z",
    "modified": "2016-04-06T20:06:37Z",
    "version": 1,
    "source_ref": "indicator--8e2e2d2b-17d4-4cbf-938f-98ee46b3cd3f",
    "target_ref": "malware--31b940d4-6f7f-459a-80ea-9c1f17b5891b",
    "relationship_type": "indicates"
  },
  {
    "type": "malware",
    "id": "malware--31b940d4-6f7f-459a-80ea-9c1f17b5891b",
    "created": "2016-04-06T20:07:09Z",
    "modified": "2016-04-06T20:07:09Z",
    "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
    "version": 1,
    "name": "Poison Ivy"
  }
]
```

## 5.6. Intrusion Set

**Type Name:** `intrusion-set`

An Intrusion Set is a grouped set of adversarial behaviors and resources with common properties that is believed to be orchestrated by a single organization. An Intrusion Set may capture multiple Campaigns or other activities that are all tied together by shared attributes indicating a common known or unknown Threat Actor. New activity can be attributed to an Intrusion Set even if the Threat Actors behind the attack are not known. Threat Actors can move from supporting one Intrusion Set, to supporting another, or they may support multiple Intrusion Sets.

Where a Campaign is a set of attacks over a period of time against a specific set of targets to achieve some objective, an Intrusion Set is the entire attack package and may be used over a very long period of time in multiple Campaigns to achieve potentially multiple purposes.

While sometimes an Intrusion Set is not active, or changes focus, it is usually difficult to know if it has truly disappeared or ended. Analysts may have varying level of fidelity on attributing an Intrusion Set back to Threat Actors and may be able to only attribute it back to a nation state or perhaps back to an organization within that nation state.

## 5.6.1. Properties

| **Common Properties** | | |
|---|---|---|
| `type, id, created_by_ref, created, modified, version, revoked, labels, external_references, object_marking_refs, granular_markings` | | |
| **Campaign Specific Properties** | | |
| `name, description, aliases, first_seen, first_seen_precision, goals, resource_level, primary_motivation, secondary_motivations, region, country` | | |
| **Property Name** | **Type** | **Description** |
| **`type`** (required) | `string` | The value of this field **MUST** be `intrusion-set` |
| **`name`** (required) | `string` | A name used to identify this Intrusion Set. |
| **`description`** (optional) | `string` | A description that provides more details and context about the Intrusion Set, potentially including its purpose and its key characteristics. |
| **`aliases`** (optional) | `list` of type `string` | Alternative names used to identify this Intrusion Set. |
| **`first_seen`** (optional) | `timestamp` | The time that this Intrusion Set was first seen. |
| **`first_seen_precision`** (optional) | `timestamp-precision` | The precision value for the **`first_seen`** field. |
| **`goals`** (optional) | `list` of type `string` | The high level goals of this Intrusion Set, namely, *what* are they trying to do. For example, they may be motivated by personal gain, but their goal is to steal credit card numbers. To do this, they may execute specific Campaigns that have detailed objectives like compromising point of sale systems at a large retailer.<br><br>Another example: to gain information about latest merger and |

| | | |
|---|---|---|
| | | IPO information from ACME Bank. |
| **`resource_level`** (optional) | `open-vocab` | This defines the organizational level at which this Intrusion Set typically works, which in turn determines the resources available to this Intrusion Set for use in an attack.<br><br>This is an open vocabulary and values **SHOULD** come from the `attack-resource-level-ov` vocabulary. |
| **`primary_motivation`** (optional) | `open-vocab` | The primary reason, motivation, or purpose behind this Intrusion Set. The motivation is *why* the Intrusion Set wishes to achieve the goal (what they are trying to achieve).<br><br>For example, an Intrusion Set with a goal to disrupt the finance sector in a country might be motivated by ideological hatred of capitalism.<br><br>This is an open vocabulary and values **SHOULD** come from the `attack-motivation-ov` vocabulary. |
| **`secondary_motivations`** (optional) | `list` of type `open-vocab` | The secondary reasons, motivations, or purposes behind this Intrusion Set. These motivations can exist as an equal or near-equal cause to the primary motivation. However, it does not replace or necessarily magnify the primary motivation, but it might indicate additional context.<br><br>This is an open vocabulary and values **SHOULD** come from the `attack-motivation-ov` vocabulary. |
| **`region`** (optional) | `string` | The primary region of origin for this Intrusion Set, if the actual country is not yet known. |
| **`country`** (optional) | `string` | The primary country of origin for this Intrusion Set. The value **MUST** |

|  |  |  |
|---|---|---|
|  |  | be from the ISO 3166-1 Alpha-2 codes and represented in lowercase <TODO ISO Ref>. |

### 5.6.2. Relationships

These are the relationships explicitly defined between the Intrusion Set object and other objects. The first section lists the embedded relationships by property name along with their corresponding target. The rest of the table identifies the relationships that can be made from the Intrusion Set object by way of the Relationship object. The reverse relationships (relationships "to" the Intrusion Set object) are included as a convenience. For their definitions, please see the objects for which they represent a "from" relationship.

Relationships are not restricted to those listed below. Relationships can be created between any objects using the `related-to` relationship type or, as with open vocabularies, user-defined names.

| **Embedded Relationships** | | | |
|---|---|---|---|
| **`created_by_ref`** | `identity` | | |
| **`object_marking_refs`** | `marking-definition` | | |
| **Common Relationships** | | | |
| `duplicate-of`, `derived-from`, `related-to` | | | |
| **Source** | **Relationship Type** | **Target** | **Description** |
| `intrusion-set` | `attributed-to` | `threat-actor` | This Relationship describes that the related Threat Actor is involved in carrying out the Intrusion Set.<br><br>For example, an `attributed-to` Relationship from the Red Orca Intrusion Set to the Urban Fowl Threat Actor means that the actor carried out or was involved in some of the activity described by the Intrusion Set. |
| `intrusion-set` | `targets` | `identity`, `vulnerability` | This Relationship describes that the Intrusion Set uses exploits of the related Vulnerability or targets the type of victims described by the related Identity. |

| | | | |
|---|---|---|---|
| | | | For example, a `targets` Relationship from the Red Orca Intrusion Set to a Vulnerability in a blogging platform indicates that attacks performed as part of Red Orca often exploit that Vulnerability.<br><br>Similarly, a `targets` Relationship from the Red Orca Intrusion Set to an Identity describing the energy sector in the United States means that the Intrusion Set typically carries out attacks against targets in that sector. |
| `intrusion-set` | `uses` | `attack-pattern`, `malware`, `tool` | This Relationship describes that attacks carried out as part of the Intrusion Set typically use the related Attack Pattern, Malware, or Tool.<br><br>For example, a `uses` Relationship from the Red Orca Intrusion Set to the xInject Malware indicates that xInject is often used during attacks attributed to that Intrusion Set. |
| **Reverse Relationships** | | | |
| `campaign` | `attributed-to` | `intrusion-set` | See forward relationship for definition. |
| `indicator` | `indicates` | `intrusion-set` | See forward relationship for definition. |

### 5.6.3. Example

```
{
  "type": "intrusion-set",
  "id": "intrusion-set--4e78f46f-a023-4e5f-bc24-71b3ca22ec29",
  "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
  "created": "2016-04-06T20:03:48Z",
  "modified": "2016-04-06T20:03:48Z",
  "version": 1,
  "name": "Bobcat Breakin",
  "description": "Incidents usually feature a shared TTP of a bobcat being released within the
building containing network access, scaring users to leave their computers without locking
them first. Still determining where the threat actors are getting the bobcats.",
  "aliases": ["Zookeeper"],
  "goals": ["acquisition-theft", "harassment", "damage"]
}
```

## 5.7. Malware

**Type Name:** `malware`

**Note: The Malware object in STIX 2.0 is a stub. It is included to support basic use cases but is likely not useful for actual malware analysis or for including even simple malware instance data. Future versions of STIX 2 will expand it to include these capabilities.**

Malware is a type of TTP that is also known as malicious code and malicious software, refers to a program that is inserted into a system, usually covertly, with the intent of compromising the confidentiality, integrity, or availability of the victim's data, applications, or operating system (OS) or of otherwise annoying or disrupting the victim. Malware such as viruses and worms are usually designed to perform these nefarious functions in such a way that users are unaware of them, at least initially.[1]

The Malware SDO characterizes, identifies, and categorizes malware samples and families via a text description field. This provides detailed information about how the malware works and what it does. Relationships from Malware can capture what the malware targets (Vulnerability and Identity) and link it to another Malware SDO that is a variant of.

### 5.7.1. Properties

| **Common Properties** | | |
|---|---|---|
| **`type, id, created_by_ref, created, modified, version, revoked, labels, external_references, object_marking_refs, granular_markings`** | | |
| **Malware Specific Properties** | | |
| **`name, description, kill_chain_phases`** | | |
| **Property Name** | **Type** | **Description** |
| **`type`** (required) | `string` | The value of this field **MUST** be `malware` |
| **`labels`** (required) | `list` of type `open-vocab` | The type of malware being described.<br><br>This is an open vocabulary and values **SHOULD** come from the `malware-labels-ov` vocabulary. |

[1] NIST SP 800-83. http://csrc.nist.gov/publications/nistpubs/800-83/SP800-83.pdf.

| | | |
|---|---|---|
| **name** (required) | string | A name used to identify the Malware sample. |
| **description** (optional) | string | A description that provides more details and context about the Malware, potentially including its purpose and its key characteristics. |
| **kill_chain_phases** (optional) | list of type kill-chain-phase | The list of Kill Chain Phases for which this Malware can be used. |

### 5.7.2. Relationships

These are the relationships explicitly defined between the Malware object and other objects. The first section lists the embedded relationships by property name along with their corresponding target. The rest of the table identifies the relationships that can be made from the Malware object by way of the Relationship object. The reverse relationships (relationships "to" the Malware object) are included as a convenience. For their definitions, please see the objects for which they represent a "from" relationship.

Relationships are not restricted to those listed below. Relationships can be created between any objects using the related-to relationship type or, as with open vocabularies, user-defined names.

| **Embedded Relationships** | | | |
|---|---|---|---|
| **created_by_ref** | | identity | |
| **object_marking_refs** | | marking-definition | |
| **Common Relationships** | | | |
| duplicate-of, derived-from, related-to | | | |
| **Source** | **Relationship Type** | **Target** | **Description** |
| malware | targets | identity, vulnerability | This Relationship documents that this Malware is being used to target this Identity or exploit the Vulnerability.<br><br>For example, a targets Relationship linking a Malware representing a downloader to a |

|  |  |  |  |
|---|---|---|---|
|  |  |  | Vulnerability for CVE-2016-0001 means that the malware exploits that vulnerability.<br><br>Similarly, a `targets` Relationship linking a Malware representing a downloader to an Identity representing the energy sector means that downloader is typically used against targets in the energy sector. |
| `malware` | `uses` | `tool` | This Relationship documents that this Malware uses the related tool to perform its functions. |
| `malware` | `variant-of` | `malware` | This Relationship is used to document that one piece of Malware is a variant of another piece of Malware.<br><br>For example, TorrentLocker is a variant of CryptoLocker. |
| **Reverse Relationships** |  |  |  |
| `indicator` | `indicates` | `malware` | See forward relationship for definition. |
| `course-of-action` | `mitigates` | `malware` | See forward relationship for definition. |
| `attack-pattern`, `campaign`, `intrusion-set`, `threat-actor` | `uses` | `malware` | See forward relationship for definition. |

### 5.7.3. Examples

```
{
  "type": "malware",
  "id": "malware--0c7b5b88-8ff7-4a4d-aa9d-feb398cd0061",
  "created": "2016-05-12T08:17:27.000000Z",
  "modified": "2016-05-12T08:17:27.000000Z",
  "version": "1",
  "name": "Cryptolocker",
  "description": "...",
  "labels": ["ransomware"]
}
```

## 5.8. Observed Data

**Type Name:** `observed-data`

Observed Data conveys information that was observed on systems and networks, such as log data or network traffic, using the CybOX specification. For example, Observed Data can capture the observation of an IP address, a network connection, a file, or a registry key. Observed Data is not an intelligence assertion, it is simply information: this file was seen, without any context for what it means.

Observed Data captures both a single observation of a single entity (file, network connection) as well as the aggregation of multiple observations of an entity. When the **`number_observed`** property is `1` the Observed Data is of a single entity. When the **`number_observed`** property is greater than `1`, the observed data consists of several instances collected over the time window specified by the **first_observed** and **last_observed** properties. When used to collect aggregate data, it is likely that some fields in the CybOX object (e.g., timestamp fields) will be omitted because they would differ for each of the individual observations.

Observed Data may be used by itself (without relationships) to convey raw data collected from network and host-based detection tools. A firewall could emit a single Observed Data instance with a network connection for every connection it sees. The firewall could also aggregate data and instead send out an Observed Data instance every ten minutes with an appropriate **`number_observed`** value.

Observed Data may also be related to other SDOs to represent raw data that is relevant to those objects. The Sighting object, which captures the sighting of an Indicator, Malware, or other SDO, uses Observed Data to represent the raw information that led to the creation of the Sighting (i.e., what was actually seen to make you think you had that Malware instance).

### 5.8.1. Properties

| **Common Properties** | | |
|---|---|---|
| **`type, id, created_by_ref, created, modified, version, revoked, labels, external_references, object_marking_refs, granular_markings`** | | |
| **Observed Data Specific Properties** | | |
| **`first_observed, last_observed, number_observed, cybox`** | | |
| **Property Name** | **Type** | **Description** |

|  |  |  |
|---|---|---|
| `type` (required) | `string` | The value of this field **MUST** be `observed-data` |
| `first_observed` (required) | `timestamp` | The beginning of the time window that the data was observed during. |
| `last_observed` (required) | `timestamp` | The end of the time window that the data was observed during. |
| `number_observed` (required) | `number` | The number of times the data represented in the **cybox** property was observed. This **MUST** be an integer between 1 and 999,999,999 inclusive.<br><br>If the **number_observed** property is greater than `1`, the data contained in the **cybox** field was observed multiple times. In these cases, object creators **MAY** omit properties of the CybOX object (such as timestamps) that are specific to a single instance of that observed data. |
| `cybox` (required) | `cybox-container` | The CybOX content that describes a single "fact" that was observed.<br><br>The CybOX content may include multiple objects if those objects are part of a single observation. Multiple objects **MUST NOT** be used within the same Observed Data instance to describe multiple observations. |

### 5.8.2. Relationships

There are no relationships explicitly defined between the Observed Data object and other objects, other than those defined as common relationships. The first section lists the embedded relationships by property name along with their corresponding target.

In addition to the relationships created using the generic Relationship object, Observed Data is also a direct target of the Sighting SRO. Sightings represent a relationship between some intelligence entity that was seen (e.g., an Indicator or Malware instance), where it was seen, and what evidence was actually seen. The evidence (or raw data) in that relationship is captured as Observed Data.

Relationships are not restricted to those listed below. Relationships can be created between any objects using the `related-to` relationship type or, as with open vocabularies, user-defined names.

| **Embedded Relationships** | | | |
|---|---|---|---|
| **`created_by_ref`** | `identity` | | |
| **`object_marking_refs`** | `marking-definition` | | |
| **Common Relationships** | | | |
| `duplicate-of`, `derived-from`, `related-to` | | | |
| **Source** | **Name** | **Target** | **Description** |
| — | — | — | — |

## 5.8.3. Examples

Observed Data of a file object

```
{
  "type": "observed-data",
  "id": "observed-data--b67d30ff-02ac-498a-92f9-32f845f448cf",
  "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
  "created": "2016-04-06T19:58:16Z",
  "modified": "2016-04-06T19:58:16Z",
  "version": 1,
  "first_observed": "2015-12-21T19:00:00Z",
  "last_observed": "2015-12-21T19:00:00Z",
  "number_observed": 50,
  "cybox": {
    "spec_version": "3.0",
    "objects": {
      "0": {
        "type": "file-object",
        ...
        }
      }
    }
  }
}
```

## 5.9. Report

**Type Name:** `report`

Reports are collections of threat intelligence focused on one or more topics, such as a description of a threat actor, malware, or attack technique, including context and related details. They are used to group similar threat intelligence together so that it can be published as a comprehensive cyber threat story.

The Report SDO contains a list of references to SDOs and SROs (the CTI objects included in the report) along with a textual description and the name of the report.

For example, a threat report produced by ACME Defense Corp. discussing the Glass Gazelle campaign could be represented using Report. The Report itself would contain the narrative of the report while the Campaign SDO and any related SDOs (e.g., Indicators for the Campaign, Malware it uses, and the associated Relationships) would be referenced in the report contents.

### 5.9.1. Properties

| **Common Properties** |
|---|
| **`type, id, created_by_ref, created, modified, version, revoked, labels, external_references, object_marking_refs, granular_markings`** |
| **Report Specific Properties** |
| **`name, description, published, object_refs`** |

| **Property Name** | **Type** | **Description** |
|---|---|---|
| **`type`** (required) | `string` | The value of this field **MUST** be `report` |
| **`labels`** (required) | `list` of type `open-vocab` | This field is an Open Vocabulary that specifies the primary subject of this report.<br><br>This is an open vocabulary and values **SHOULD** come from the `report-label-ov` vocabulary. |
| **`name`** (required) | `string` | A name used to identify the Report. |
| **`description`** (optional) | `string` | A description that provides more details and context about the Report, potentially including its purpose and its key characteristics. |

| | | |
|---|---|---|
| **published** (required) | timestamp | The date that this Report object was officially published by the creator of this report.<br><br>The publication date (public release, legal release, etc.) may be different than the date the report was created or shared internally (the date in the **created** property). |
| **object_refs** (required) | list of type identifier | Specifies the STIX Objects that are referred to by this Report. |

## 5.9.2. Relationships

There are no relationships explicitly defined between the Report object and other objects, other than those defined as common relationships. The first section lists the embedded relationships by property name along with their corresponding target.

Relationships are not restricted to those listed below. Relationships can be created between any objects using the related-to relationship name or, as with open vocabularies, user-defined names.

| **Embedded Relationships** | |
|---|---|
| **created_by_ref** | identity |
| **object_marking_refs** | marking-definition |
| **object_refs** | list of type identifier |
| **Common Relationships** | |
| duplicate-of, derived-from, related-to | |

## 5.9.3. Examples

A standalone Report; the consumer may or may not already have access to the referenced STIX Objects.

```
{
  "type": "report",
  "id": "report--84e4d88f-44ea-4bcd-bbf3-b2c1c320bcb3",
  "created_by_ref": "identity--a463ffb3-1bd9-4d94-b02d-74e4f1658283",
  "created": "2015-12-21T19:59:11Z",
  "modified": "2016-05-21T19:59:11Z",
  "version": 1,
  "name": "The Black Vine Cyberespionage Group",
  "description": "A simple report with an indicator and campaign",
  "labels": ["campaign"],
  "object_refs": [
    "indicator--26ffb872-1dd9-446e-b6f5-d58527e5b5d2",
    "campaign--83422c77-904c-4dc1-aff5-5c38f3a2c55c",
    "relationship--f82356ae-fe6c-437c-9c24-6b64314ae68a"
  ]
}
```

A Bundle with a Report and the STIX Objects that are referred to by  the Report

```
{
  "type": "bundle",
  "id": "bundle--44af6c39-c09b-49c5-9de2-394224b04982",
  "identities": [
    {
      "type": "identity",
      "id": "identity--a463ffb3-1bd9-4d94-b02d-74e4f1658283",
      ...,
      "name": "Acme Cybersecurity Solutions"
    }
  ],
   "reports": [
    {
      "type": "report",
      "id": "report--84e4d88f-44ea-4bcd-bbf3-b2c1c320bcbd",
      "created_by_ref": "identity--a463ffb3-1bd9-4d94-b02d-74e4f1658283",
      "created": "2015-12-21T19:59:11Z",
      "modified": "2016-05-21T19:59:11Z",
      "version": 1,
      "name": "The Black Vine Cyberespionage Group",
      "description": "A simple report with an indicator and campaign",
      "labels": ["campaign"],
      "report_contains_refs": [
        "indicator--26ffb872-1dd9-446e-b6f5-d58527e5b5d2",
        "campaign--83422c77-904c-4dc1-aff5-5c38f3a2c55c",
        "relationship--f82356ae-fe6c-437c-9c24-6b64314ae68a"
      ]
    }
  ],
  "indicators": [
    {
      "type": "indicator",
      "id": "indicator--26ffb872-1dd9-446e-b6f5-d58527e5b5d2",
      "created": "2015-12-21T19:59:17Z",
      "modified": "2016-05-21T19:59:17Z",
      "version": 1,
      "name": "Some indicator",
      "indicator_types": ["anonymization"],
      "created_by_ref": "identity--a463ffb3-1bd9-4d94-b02d-74e4f1658283"
    }
  ],
  "campaigns": [
    {
      "type": "campaign",
      "id": "campaign--83422c77-904c-4dc1-aff5-5c38f3a2c55c",
      "created_by_ref": "identity--a463ffb3-1bd9-4d94-b02d-74e4f1658283",
      "created": "2015-12-21T19:59:17Z",
      "modified": "2016-05-21T19:59:17Z",
      "version": 1,
      "name": "Some Campaign"
    }
  ],
  “relationships”: [
    {
      "id": "relationship--f82356ae-fe6c-437c-9c24-6b64314ae68a",
      "type": "relationship",
      "created_by_ref": "identity--a463ffb3-1bd9-4d94-b02d-74e4f1658283",
      "created": "2015-12-21T19:59:17.000000+00:00",
      "source_ref": "indicator--26ffb872-1dd9-446e-b6f5-d58527e5b5d2",
      "target_ref": "campaign--26ffb872-1dd9-446e-b6f5-d58527e5b5d2",
      "name": "indicates"
    },
  ]
}
```

## 5.10. Threat Actor

**Type Name:** `threat-actor`

Threat Actors are actual individuals, groups, or organizations believed to be operating with malicious intent. A Threat Actor is not an Intrusion Set but may support or be affiliated with various Intrusion Sets, groups, or organizations over time.

Threat Actors leverage their resources and the resources of an Intrusion Set to conduct attacks and run Campaigns against targets.

Threat Actors can be characterized by their motives, capabilities, intentions/goals, sophistication level, past activities, resources they have access to, and their role in the organization.

### 5.10.1. Properties

| Common Properties |
|---|

| type, id, created_by_ref, created, modified, version, revoked, labels, external_references, object_marking_refs, granular_markings | | |
|---|---|---|
| **Threat Actor Specific Properties** | | |
| **name, description, aliases, roles, goals, sophistication, resource_level, primary_motivation, secondary_motivations, personal_motivations** | | |
| **Property Name** | **Type** | **Description** |
| **type** (required) | string | The value of this field **MUST** be threat-actor |
| **labels** (required) | list of type open-vocab | This field specifies the type of threat actor, if known or suspected.<br><br>This is an open vocabulary and values **SHOULD** come from the threat-actor-label-ov vocabulary. |
| **name** (required) | string | A name used to identify this Threat Actor or Threat Actor group. |
| **description** (optional) | string | A description that provides more details and context about the Threat Actor, potentially including its purpose and its key characteristics. |
| **aliases** (optional) | list of type string | A list of other names that this Threat Actor is believed to use. |
| **roles** (optional) | list of type open-vocab | A list of roles the Threat Actor plays.<br><br>This is an open vocabulary and the values **SHOULD** come from the threat-actor-roles-ov vocabulary. |
| **goals** (optional) | list of type string | The high level goals of this Threat Actor, namely, *what* are they trying to do. For example, they may be motivated by personal gain, but their goal is to steal credit card numbers. To do this, they may execute specific Campaigns that have detailed objectives like compromising point of sale systems at a large retailer. |
| **sophistication** (optional) | open-vocab | The skill, specific knowledge, special training, or expertise a Threat Actor must have to perform the attack. |

| | | |
|---|---|---|
| | | This is an open vocabulary and values **SHOULD** come from the `threat-actor-sophistication-ov` vocabulary. |
| **`resource_level`** (optional) | `open-vocab` | This defines the organizational level at which this Threat Actor typically works, which in turn determines the resources available to this Threat Actor for use in an attack. This attribute is linked to the Sophistication Level attribute — a specific resource level implies that the Threat Actor has access to at least a specific sophistication level.<br><br>This is an open vocabulary and values **SHOULD** come from the `attack-resource-level-ov` vocabulary. |
| **`primary_motivation`** (optional) | `open-vocab` | The primary reason, motivation, or purpose behind this Threat Actor. The motivation is *why* the Threat Actor wishes to achieve the goal (what they are trying to achieve).<br><br>For example, a Threat Actor with a goal to disrupt the finance sector in a country might be motivated by ideological hatred of capitalism.<br><br>This is an open vocabulary and values **SHOULD** come from the `attack-motivation-ov` vocabulary. |
| **`secondary_motivations`** (optional) | `list` of type `open-vocab` | The secondary reasons, motivations, or purposes behind this Threat Actor.<br><br>These motivations can exist as an equal or near-equal cause to the primary motivation. However, it does not replace or necessarily magnify the primary motivation, but it might indicate additional context.<br><br>This is an open vocabulary and values **SHOULD** come from the `attack-motivation-ov` vocabulary. |
| **`personal_motivations`** (optional) | `list` of type `open-vocab` | The personal reasons, motivations, or purposes of the Threat Actor regardless of organizational goals.<br><br>Personal motivation, which is independent of the organization's goals, describes what |

| | | |
|---|---|---|
| | | impels an individual to carry out an attack. Personal motivation may align with the organization’s motivation—as is common with activists—but more often it supports personal goals. For example, an individual analyst may join a Data Miner corporation because his or her skills may align with the corporation’s objectives. But the analyst most likely performs his or her daily work toward those objectives for personal reward in the form of a paycheck. The motivation of personal reward may be even stronger for Threat Actors who commit illegal acts, as it is more difficult for someone to cross that line purely for altruistic reasons.<br><br>This is an open vocabulary and values **SHOULD** come from the `attack-motivation-ov` vocabulary. |

### 5.10.2. Relationships

These are the relationships explicitly defined between the Threat Actor object and other objects. The first section lists the embedded relationships by property name along with their corresponding target. The rest of the table identifies the relationships that can be made from the Threat Actor object by way of the Relationship object. The reverse relationships (relationships "to" the Threat Actor object) are included as a convenience. For their definitions, please see the objects for which they represent a "from" relationship.

Relationships are not restricted to those listed below. Relationships can be created between any objects using the `related-to` relationship type or, as with open vocabularies, user-defined names.

| **Embedded Relationships** | | | |
|---|---|---|---|
| **`created_by_ref`** | `identity` | | |
| **`object_marking_refs`** | `marking-definition` | | |
| **Common Relationships** | | | |
| `duplicate-of`, `derived-from`, `related-to` | | | |
| **Source** | **Relationship Type** | **Target** | **Description** |

| | | | |
|---|---|---|---|
| `threat-actor` | `attributed-to`, `impersonates` | `identity` | This Relationship describes that the actor is the real identity represented in the related Identity.<br><br>For example, an `attributed-to` Relationship from the jay-sm17h Threat Actor to the John Smith Identity means that the actor known as jay-sm17h is John Smith. |
| `threat-actor` | `targets` | `identity`, `vulnerability` | This Relationship describes that the Threat Actor uses exploits of the related Vulnerability or targets the type of victims described by the related Identity.<br><br>For example, a `targets` Relationship from the John Smith Threat Actor to a Vulnerability in a blogging platform indicates that attacks performed by John Smith often exploit that Vulnerability.<br><br>Similarly, a `targets` Relationship from the John Smith Threat Actor to an Identity describing the energy sector in the United States means that John Smith often carries out attacks against targets in that sector. |
| `threat-actor` | `uses` | `attack-pattern`, `malware`, `tool` | This Relationship describes that attacks carried out as part of the Threat Actor typically use the related Attack Pattern, Malware, or Tool.<br><br>For example, a `uses` Relationship from the John Smith Threat Actor to the xInject Malware indicate that xInject is often used by John Smith. |
| **Reverse Relationships** | | | |
| `campaign, intrusion-set,` | `attributed-to` | `threat-actor` | See forward relationship for definition. |

|  |  |  |  |
|---|---|---|---|
| `indicator` | `indicates` | `threat-actor` | See forward relationship for definition. |

### 5.10.3. Examples

```
{
  "type": "threat-actor",
  "id": "threat-actor--8e2e2d2b-17d4-4cbf-938f-98ee46b3cd3f",
  "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
  "created": "2016-04-06T20:03:48Z",
  "modified": "2016-04-06T20:03:48Z",
  "version": 1,
  "name": "Evil Org",
  "description": "The Evil Org threat actor group"
}
```

## 5.11. Tool

**Type Name:** `tool`

Tools are legitimate software that can be used by threat actors to perform attacks. Knowing how and when threat actors use such tools can be important for understanding how campaigns are executed. Unlike malware, these tools or software packages are often found on a system and have legitimate purposes for power users, system administrators, network administrators, or even normal users. Remote access tools (e.g., RDP) and network scanning tools (e.g., NMAP) are examples of Tools that may be used by a Threat Actor during an attack.

The Tool SDO characterizes the properties of these software tools and can be used as a basis for making an assertion about how a Threat Actor uses them during an attack. It contains properties to name and describe the tool, a list of Kill Chain Phases the tool can be used to carry out, and the version of the tool.

This SDO **MUST NOT** be used to document malware. Further, Tool **MUST NOT** be used to document tools used as part of a course of action in response to an attack. Tools used during response activities can be included directly as part of a Course of Action SDO.

### 5.11.1. Properties

| **Common Properties** |
|---|
| **`type, id, created_by_ref, created, modified, version, revoked, labels,`** |

| | | |
|---|---|---|
| `external_references, object_marking_refs, granular_markings` | | |
| **Tool Specific Properties** | | |
| `name, description, kill_chain_phases, tool_version` | | |
| **Property Name** | **Type** | **Description** |
| `type` (required) | `string` | The value of this field **MUST** be `tool` |
| `labels` (required) | `list` of type `open-vocab` | The kind(s) of tool(s) being described.<br><br>This is an open vocabulary and values **SHOULD** come from the `tool-label-ov` vocabulary. |
| `name` (required) | `string` | The name used to identify the Tool. |
| `description` (optional) | `string` | A description that provides more details and context about the Tool, potentially including its purpose and its key characteristics. |
| `kill_chain_phases` (optional) | `list` of type `kill-chain-phase` | The list of Kill Chain Phases for which this Tool can be used. |
| `tool_version` (optional) | `string` | The version identifier associated with the Tool. |

### 5.11.2. Relationships

These are the relationships explicitly defined between the Tool object and other objects. The first section lists the embedded relationships by property name along with their corresponding target. The rest of the table identifies the relationships that can be made from the Tool object by way of the Relationship object. The reverse relationships (relationships "to" the Tool object) are included as a convenience. For their definitions, please see the objects for which they represent a "from" relationship.

Relationships are not restricted to those listed below. Relationships can be created between any objects using the `related-to` relationship type or, as with open vocabularies, user-defined names.

| **Embedded Relationships** |
|---|

| | |
|---|---|
| **`created_by_ref`** | `identity` |
| **`object_marking_refs`** | `marking-definition` |
| **Common Relationships** | |
| `duplicate-of`, `derived-from`, `related-to` | |

| **Source** | **Relationship Type** | **Target** | **Description** |
|---|---|---|---|
| `tool` | `targets` | `identity`, `vulnerability` | This Relationship documents that this Tool is being used to target this Identity or exploit the Vulnerability.<br><br>For example, a `targets` Relationship linking an exploit Tool to a Vulnerability for CVE-2016-0001 means that the tool exploits that vulnerability.<br><br>Similarly, a `targets` Relationship linking a DDoS Tool to an Identity representing the energy sector means that Tool is typically used against targets in the energy sector. |
| **Reverse Relationships** | | | |
| `indicator` | `indicates` | `tool` | See forward relationship for definition |
| `course-of-action` | `mitigates` | `tool` | See forward relationship for definition |
| `attack-pattern`, `campaign`, `intrusion-set`, `malware`, `threat-actor` | `uses` | `tool` | See forward relationship for definition |

### 5.11.3. Examples

```
{
  "type": "tool",
  "id": "tool--8e2e2d2b-17d4-4cbf-938f-98ee46b3cd3f",
  "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
  "created": "2016-04-06T20:03:48Z",
  "modified": "2016-04-06T20:03:48Z",
  "version": 1,
  "name": "VNC"
}
```

## 5.12. Vulnerability

**Type Name:** `vulnerability`

A Vulnerability is a mistake in software that can be directly used by a hacker to gain access to a system or network [TODO add NIST ref]. For example, if a piece of malware exploits CVE-2015-12345, a Malware object could be linked to a Vulnerability object that references CVE-2015-12345.

The Vulnerability SDO is primarily used to link to external definitions of vulnerabilities or to describe 0-day vulnerabilities that do not yet have an external definition. Typically, other SDOs assert relationships to Vulnerability objects when a specific vulnerability is targeted and exploited as part of malicious cyber activity. As such, Vulnerability objects can be used as a linkage to the asset management and compliance process.

### 5.12.1. Properties

| **Common Properties** | | |
|---|---|---|
| **type, id, created_by_ref, created, modified, version, revoked, labels, external_references, object_marking_refs, granular_markings** | | |
| **Vulnerability Specific Properties** | | |
| **name, description** | | |
| **Property Name** | **Type** | **Description** |
| **type** (required) | `string` | The value of this field MUST be `vulnerability` |
| **external_references** (optional) | `list` of type `external-reference` | A list of external references which refer to non-STIX information. This field **MAY** be used to provide one or more Vulnerability identifiers, such as a CVE ID [TODO: add reference]. When specifying a CVE ID, the **source_name** field of the external reference **MUST** be set to `cve` and the **external_id** field |

|  |  |  |
|---|---|---|
|  |  | **MUST** be the exact CVE identifier. |
| **name** (required) | string | A name used to identify the Vulnerability. |
| **description** (optional) | string | A description that provides more details and context about the Vulnerability, potentially including its purpose and its key characteristics. |

### 5.12.2. Relationships

These are the relationships explicitly defined between the Vulnerability object and other objects. The first section lists the embedded relationships by property name along with their corresponding target. The rest of the table identifies the relationships that can be made from the Vulnerability object by way of the Relationship object. None are defined for the Vulnerability object. The reverse relationships (relationships "to" the Vulnerability object) are included as a convenience. For their definitions, please see the objects for which they represent a "from" relationship.

Relationships are not restricted to those listed below. Relationships can be created between any objects using the related-to relationship type or, as with open vocabularies, user-defined names.

| **Embedded Relationships** |  |  |  |
|---|---|---|---|
| **created_by_ref** | identity |  |  |
| **object_marking_refs** | marking-definition |  |  |
| **Common Relationships** |  |  |  |
| duplicate-of, derived-from, related-to |  |  |  |
| **Source** | **Relationship Type** | **Target** | **Description** |
| — | — | — | — |
| **Reverse Relationships** |  |  |  |
| attack-pattern, campaign, intrusion-set, | targets | vulnerability | See forward relationship for definition. |

| | | | |
|---|---|---|---|
| `malware`, `threat-actor`, `tool` | | | |
| `course-of-action` | `mitigates` | `vulnerability` | See forward relationship for definition. |

### 5.12.3. Examples

```
{
  "type": "vulnerability",
  "id": "vulnerability--0c7b5b88-8ff7-4a4d-aa9d-feb398cd0061",
  "created": "2016-05-12T08:17:27.000000Z",
  "modified": "2016-05-12T08:17:27.000000Z",
  "version": 1,
  "name": "CVE-2016-1234"
  "external_references": [
    {
      "source_name": "cve",
      "id": "CVE-2016-1234"
    }
  ]
}
```

# 6. STIX Relationship Objects

STIX Relationship Objects (SROs) represent types of relationships used to describe CTI. The generic Relationship SRO is used to describe many varied types of relationships, while the specific Sighting SRO contains additional properties to represent Sighting relationships.

Property information, relationship information, and examples are provided for each SRO defined below. Property information includes common properties as well as properties that are specific to each SRO. Because SROs cannot be the source or target of other SROs, relationship information is included but only to describe embedded relationships (e.g., **`created_by_ref`**).

## 6.1. Relationship

**Type Name:** `relationship`

The Relationship object is used to link together two SDOs in order to describe how they are related to each other. If SDOs are considered "nodes" or "vertices" in the graph, the Relationship objects (SROs) represent "edges".

STIX defines many relationship types to link together SDOs. These relationships are contained in the "Relationships" table under each SDO definition. Relationship types defined in the specification **SHOULD** be used to ensure consistency. An example of a specification-defined relationship is that an `indicator` `indicates` a `campaign`. That relationship type is listed in the Relationships section of the Indicator SDO definition.

STIX also allows relationships from any SDO to any SDO that have not been defined in this specification. These relationships **MAY** use the `related-to` relationship type or **MAY** use a custom relationship type. As an example, a user might want to link `malware` directly to a `tool`. They can do so using `related-to` to say that the Malware is related to the Tool but not describe how, or they could use `delivered-by` (a custom name they determined) to indicate more detail.

Note that some relationships in STIX may seem like "shortcuts". For example, an Indicator doesn't really detect a Campaign: it detects activity (Attack Patterns, Malware, etc.) that are often used by that campaign. While some analysts might want all of the source data and think that shortcuts are misleading, in many cases it's helpful to provide just the key points (shortcuts) and leave out the low-level details. In other cases, the low-level analysis may not be known or sharable, while the high-level analysis is. For these reasons, relationships that might appear to be "shortcuts" are not excluded from STIX.

### 6.1.1. Specification-Defined Relationships Summary

This relationship summary table is provided as a convenience. If there is a discrepancy between this table and the relationships defined with each of the SDOs, then the relationships defined with the SDOs **MUST** be viewed as authoritative.

| Source | Type | Target | Source | Type | Target |
|---|---|---|---|---|---|
| `attack-pattern` | `targets` | `vulnerability` | `intrusion-set` | `attributed-to` | `threat-actor` |
| `attack-pattern` | `targets` | `identity` | `intrusion-set` | `targets` | `identity` |
| `attack-pattern` | `uses` | `malware` | `intrusion-set` | `targets` | `vulnerability` |
| `attack-pattern` | `uses` | `tool` | `intrusion-set` | `uses` | `attack-pattern` |
| `campaign` | `attributed-to` | `intrusion-set` | `intrusion-set` | `uses` | `malware` |
| `campaign` | `attributed-to` | `threat-actor` | `intrusion-set` | `uses` | `tool` |

| | | | | | |
|---|---|---|---|---|---|
| `campaign` | `targets` | `identity` | `malware` | `targets` | `identity` |
| `campaign` | `targets` | `vulnerability` | `malware` | `targets` | `vulnerability` |
| `campaign` | `uses` | `attack-pattern` | `malware` | `uses` | `tool` |
| `campaign` | `uses` | `malware` | `malware` | `variant-of` | `malware` |
| `campaign` | `uses` | `tool` | `threat-actor` | `attributed-to` | `identity` |
| `course-of-action` | `mitigates` | `attack-pattern` | `threat-actor` | `impersonates` | `identity` |
| `course-of-action` | `mitigates` | `malware` | `threat-actor` | `targets` | `identity` |
| `course-of-action` | `mitigates` | `tool` | `threat-actor` | `targets` | `vulnerability` |
| `course-of-action` | `mitigates` | `vulnerability` | `threat-actor` | `uses` | `attack-pattern` |
| `indicator` | `indicates` | `attack-pattern` | `threat-actor` | `uses` | `malware` |
| `indicator` | `indicates` | `campaign` | `threat-actor` | `uses` | `tool` |
| `indicator` | `indicates` | `intrusion-set` | `tool` | `targets` | `identity` |
| `indicator` | `indicates` | `malware` | `tool` | `targets` | `vulnerability` |
| `indicator` | `indicates` | `threat-actor` | | | |
| `indicator` | `indicates` | `tool` | | | |

### 6.1.2. Properties

| **Common Properties** | | |
|---|---|---|
| **`type, id, created_by_ref, created, modified, version, revoked, labels, external_references, object_marking_refs, granular_markings`** | | |
| **Relationship Specific Properties** | | |
| **`relationship_type, description, source_ref, target_ref`** | | |
| **Property Name** | **Type** | **Description** |
| **`type`** (required) | `string` | The value of this field **MUST** be `relationship` |
| **`relationship_type`** (required) | `string` | The name used to identify the type of Relationship. This value **SHOULD** be an exact value listed in the relationships for the source and target SDO, but **MAY** be |

| | | |
|---|---|---|
| | | any string. The value of this field **MUST** be in ASCII and is limited to characters a–z (lowercase ASCII), 0–9, and dash (-). |
| **`description`** (optional) | `string` | A description that provides more details and context about the Relationship, potentially including its purpose and its key characteristics. |
| **`source_ref`** (required) | `identifier` | The **`id`** of the source (from) object. The value **MUST** be an ID reference to an SDO (i.e., it cannot point to an SRO, Bundle, or Marking Definition). |
| **`target_ref`** (required) | `identifier` | The **`id`** of the target (to) object. The value **MUST** be an ID reference to an SDO (i.e., it cannot point to an SRO, Bundle, or Marking Definition). |

### 6.1.3. Relationships

There are no relationships between the Relationship object and other objects, other than the embedded relationships. Those relationships are listed below by property name along with their corresponding target.

| **Embedded Relationships** | |
|---|---|
| **`created_by_ref`** | `identity` |
| **`object_marking_refs`** | `marking-definition` |

## 6.2. Sighting

**Type Name:** `sighting`

A Sighting denotes the belief that something in CTI (e.g., an indicator, malware, tool, threat actor, etc.) was seen. Sightings are used to track who and what are being targeted, how attacks are carried out, and to track trends in attack behavior.

The Sighting relationship object is a special type of SRO: it is a relationship that contains extra fields not present on the generic Relationship object. These extra fields are included to represent data specific to sighting relationships (e.g., **`count`**, representing how many times something was seen), but for other purposes the Sighting can be thought of as a Relationship with a name of "sighting-of". Sighting is captured as a relationship because you cannot have a sighting unless you have something that has been sighted. Sighting does not make sense without the relationship to what was sighted.

Sighting relationships relate three aspects of the sighting:

- What was sighted, such as the Indicator, Malware, Campaign, or other SDO (**`sighting_of_ref`**);
- Who sighted it and/or where it was sighted, represented as an Identity (**`where_sighted_refs`**); and
- What was actually seen on systems and networks, represented as Observed Data (**`observed_data_refs`**).

What was sighted is required: a sighting does not make sense unless you say what you saw. Who sighted it, where it was sighted, and what was actually seen are optional: in many cases it is not necessary to provide that level of detail in order to provide value.

Sightings are used whenever any SDO has been "seen". In some cases, the object creator wishes to convey very little information about the sighting: the details might be sensitive, but the fact that they saw a malware instance or threat actor could still be very useful. In other cases, providing the details may be helpful or even necessary: saying exactly which of the 1000 IP addresses in an indicator were sighted is helpful when tracking which of those IPs is still malicious.

Sighting is distinct from Observed Data in that Sighting is an intelligence assertion ("I saw this threat actor") while Observed Data is simply information ("I saw this file"). When you combine them by including the linked Observed Data (**`observed_data_refs`**) from a Sighting, you can say "I saw this file, and that makes me think I saw this threat actor". Although **confidence** is currently reserved, notionally confidence would be added to Sighting (the intelligence relationship) but not to Observed Data (the raw information).

### 6.2.1. Properties

| **Common Properties** | | |
|---|---|---|
| **`type, id, created_by_ref, created, modified, version, revoked, labels, external_references, object_marking_refs, granular_markings`** | | |
| **Sighting Specific Properties** | | |
| **`first_seen, first_seen_precision, last_seen, last_seen_precision, count, sighting_of_ref, observed_data_refs, where_sighted_refs, summary`** | | |
| **Property Name** | **Type** | **Description** |

| | | |
|---|---|---|
| **`type`** (required) | `string` | The value of this field **MUST** be `sighting` |
| **`first_seen`** (optional) | `timestamp` | The beginning of the time window during which the SDO referenced by the `sighting_of_ref` property was sighted. |
| **`first_seen_precision`** (optional) | `timestamp-precision` | The precision of the **`first_seen`** timestamp. |
| **`last_seen`** (optional) | `timestamp` | The end of the time window during which the SDO referenced by the `sighting_of_ref` property was sighted. |
| **`last_seen_precision`** (optional) | `timestamp-precision` | The precision of the **`last_seen`** timestamp. |
| **`count`** (optional) | `number` | This **MUST** be an integer between 0 and 999,999,999 inclusive and represents the number of times the SDO referenced by the `sighting_of_ref` property was sighted.<br><br>Observed Data has a similar property called **`number_observed`**, which refers to the number of times the data was observed. These counts refer to different concepts and are distinct.<br><br>For example, a single sighting of a DDoS bot might have many millions of observations of the network traffic that it generates. Thus, the Sighting **`count`** would be 1 (the bot was observed once) but the Observed Data **`number_observed`** would be much higher. |
| **`sighting_of_ref`** (required) | `identifier` | An ID reference to the SDO that was sighted (e.g., Indicator or Malware).<br><br>For example, if this is a sighting of an Indicator, that indicator's ID would be the value of this property. |
| **`observed_data_refs`** (optional) | `list` of type `identifier` | A list of ID references to the Observed Data objects that contain |

| | | |
|---|---|---|
| | | the raw cyber data for this Sighting.<br><br>For example, a Sighting of an Indicator with an IP address could include the Observed Data for the network connection that the Indicator was used to detect. |
| **`where_sighted_refs`** (optional) | `list` of type `identifier` | A list of ID references to the Identity (victim) objects of the entities that saw the sighting.<br><br>Omitting the **`where_sighted_refs`** field does not imply that the sighting was seen by the object creator. To indicate that the sighting was seen by the object creator, an Identity representing the object creator should be listed in **`where_sighted_refs`**. |
| **`summary`** (optional) | `boolean` | The **`summary`** property indicates whether the Sighting should be considered summary data. Summary data is an aggregation of previous Sightings reports and should not be considered primary source data. Default value is `false`. |

## 6.2.2. Relationships

There are no relationships between the Sighting object and other objects, other than the embedded relationships. Those relationships are listed below by property name along with their corresponding target.

| **Embedded Relationships** | |
|---|---|
| **`created_by_ref`** | `identity` |
| **`object_marking_refs`** | `marking-definition` |
| **`sighting_of_ref`** | `identifier` |
| **`observed_data_refs`** | `list` of type `identifier` |

|  |  |
|---|---|
| **where_sighted_refs** | `list` of type `identifier` |

## 6.2.3. Examples

Sighting of Indicator, without Observed Data

```
{
  "type": "sighting",
  "id": "sighting--ee20065d-2555-424f-ad9e-0f8428623c75",
  "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
  "created": "2016-04-06T20:08:31Z",
  "modified": "2016-04-06T20:08:31Z",
  "version": 1,
  "sighting_of_ref": "indicator--8e2e2d2b-17d4-4cbf-938f-98ee46b3cd3f"
}
```

Sighting of Indicator, with Observed Data (what exactly was seen) and where it was seen

```
[
  {
    "type": "sighting",
    "id": "sighting--ee20065d-2555-424f-ad9e-0f8428623c75",
    "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
    "created": "2016-04-06T20:08:31Z",
    "modified": "2016-04-06T20:08:31Z",
    "version": 1,
    "first_seen": "2015-12-21T19:00:00Z",
    "last_seen": "2015-12-21T19:00:00Z",
    "count": 50,
    "sighting_of_ref": "indicator--8e2e2d2b-17d4-4cbf-938f-98ee46b3cd3f",
    "observed_data_refs": ["observed-data--b67d30ff-02ac-498a-92f9-32f845f448cf"],
    "where_sighted_refs": ["identity--b67d30ff-02ac-498a-92f9-32f845f448ff"]
  },
  {
    "type": "observed-data",
    "id": "observed-data--b67d30ff-02ac-498a-92f9-32f845f448cf",
    "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
    "created": "2016-04-06T19:58:16Z",
    "modified": "2016-04-06T19:58:16Z",
    "version": 1,
    "start": "2015-12-21T19:00:00Z",
    "stop": "2016-04-06T19:58:16Z",
    "count": 50,
    "cybox": {
      "objects": {
       "1": {
          "type": "file-object",
          ...
        }
      }
    ]
  }
]
```

# 7. Bundle

**Type Name:** `bundle`

A Bundle is a collection of arbitrary STIX Objects grouped together in a single container. A Bundle does not have any semantic meaning and objects in the same Bundle are not necessarily related. Objects **MUST NOT** be considered related by virtue of being in the same Bundle.

Bundle is not STIX Object, so it does not have any of the Common Properties other than the **`type`** and **`id`** fields. Bundle is transient and implementations should not assume that other implementations will treat it as a persistent object.

The JSON MTI serialization uses the JSON object type <TODO: add reference> when representing `bundle`.

## 7.1. Properties

| Property Name | Type | Description |
|---|---|---|
| **`type`** (required) | `string` | The value of this field **MUST** be `bundle` |
| **`id`** (required) | `identifier` | An identifier for this Bundle. The **`id`** field for the Bundle is designed to help tools that may need it for processing, but tools are not required to store or track it. Consuming tools should not rely on the presence of this property. |
| **`spec_version`** (required) | `string` | The version of the STIX specification used to represent the content in this Bundle. This enables non-TAXII transports or other transports without their own content identification mechanisms to know the version of STIX content.<br><br>The value of this property **MUST** be `2.0` for bundles containing STIX Objects defined in this specification. |

| | | |
|---|---|---|
| **attack_patterns** (optional) | list of type attack-pattern | Specifies a set of one or more Attack Patterns. |
| **campaigns** (optional) | list of type campaign | Specifies a set of one or more Campaigns. |
| **courses_of_action** (optional) | list of type course-of-action | Specifies a set of one or more Courses of Action. |
| **identities** (optional) | list of type identity | Specifies a set of one or more Identities. |
| **indicators** (optional) | list of type indicator | Specifies a set of one or more cyber threat Indicators. |
| **intrusion_sets** (optional) | list of type intrusion-set | Specifies a set of one or more cyber threat Intrusion Sets. |
| **malware** (optional) | list of type malware | Specifies a set of one or more Malware objects. |
| **marking_definitions** (optional) | list of type marking-definition | Specifies a set of one or more Marking Definitions. |
| **observed_data** (optional) | list of type observed-data | Specifies a set of one or more piece of Observed Data. |
| **relationships** (optional) | list of type relationship | Specifies a set of one or more relationships between SDOs. |
| **reports** (optional) | list of type report | Specifies a set of one or more Reports. |
| **sightings** (optional) | list of type sighting | Specifies a set of one or more Sightings. |
| **threat_actors** (optional) | list of type threat-actor | Specifies a set of one or more Threat Actors. |
| **tools** (optional) | list of type tool | Specifies a set of one or more Tools. |
| **vulnerabilities** (optional) | list of type vulnerability | Specifies a set of one or more Vulnerabilities. |
| **custom_objects** (optional) | list of type custom-object | Specifies a list of one or more custom objects. |

## 7.2. Relationships

Bundle is not a STIX Object and **MUST NOT** have any relationships to it or from it.

## 7.3. Examples

```
{
  "type": "bundle",
  "id": "bundle--5d0092c5-5f74-4287-9642-33f4c354e56d",
  "spec_version": "2.0”,
  "indicators": [
    {
      "type": "indicator",
      "id": "indicator--8e2e2d2b-17d4-4cbf-938f-98ee46b3cd3f",
      "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
      "created": "2016-04-29T14:09:00.123456Z",
      "modified": "2016-04-29T14:09:00.123456Z",
      "version": 1,
      "object_marking_refs": ["marking-definition--089a6ecb-cc15-43cc-9494-767639779123"],
      "name": "Poison Ivy Malware",
      "description": "This file is part of Poison Ivy",
      "pattern": "file-object.hashes.md5 = '3773a88f65a5e780c8dff9cdc3a056f3'"
    }
  ],
  "marking_definitions": [
    {
      "type": "marking-definition",
      "id": "marking-definition--34098fce-860f-48ae-8e50-ebd3cc5e41da",
      "created": "2016-08-01T00:00:00Z",
      "definition_type": "tlp",
      "definition": {
        "tlp": "green"
      }
    }
  ]
}
```

# 8. Vocabularies

The following sections provide object-specific listings for each of the vocabularies referenced in the object description sections. STIX vocabularies, which all have type names ending in '-ov', are "open": they provide a listing of common and industry accepted terms as a guide to the user but do not limit the user to that defined list.

## 8.1. Attack Motivation

**Type Name:** `attack-motivation-ov`

The attack motivation vocabulary is currently used in the following SDOs:

- Intrusion Set
- Threat Actor

Knowing a Threat Actor or Intrusion Set's motivation may allow an analyst or defender to better understand likely targets and behaviors.

Motivation shapes the intensity and the persistence of an attack. Threat Actors and Intrusion Sets usually act in a manner that reflects their underlying emotion or situation, and this informs defenders of the manner of attack. For example, a spy motivated by nationalism (ideology) likely has the patience to achieve long-term goals and work quietly for years, whereas a cyber-vandal out for notoriety can create an intense and attention-grabbing attack but may quickly lose interest and move on. Understanding these differences allows defenders to implement controls tailored to each type of attack for greatest efficiency.

(TODO REF) This section including vocabulary items and their descriptions is based on the *Threat Agent Motivations* publication from Intel Corp in February 2015[2].

| **Vocabulary Summary** | |
|---|---|
| `accidental`, `coercion`, `dominance`, `ideology`, `notoriety`, `organizational-gain`, `personal-gain`, `personal-satisfaction`, `revenge`, `unpredictable` | |
| **Vocabulary Value** | **Description** |
| `accidental` | A non-hostile actor whose benevolent or harmless intent inadvertently causes harm.<br><br>For example, a well-meaning and dedicated employee who through distraction or poor training unintentionally causes harm to his or her organization. |
| `coercion` | Being forced to act on someone else's behalf.<br><br>Adversaries who are motivated by coercion are often forced through intimidation or blackmail to act illegally for someone else's benefit. Unlike the other motivations, a coerced person does not act for personal gain, but out of fear of incurring a loss. |

[2] Intel Corp Threat Agent Motivations Feb 2015

| | |
|---|---|
| `dominance` | A desire to assert superiority over someone or something else.<br><br>Adversaries who are seeking dominance over a target are focused on using their power to force their target into submission or irrelevance. Dominance may be found with ideology in some state-sponsored attacks and with notoriety in some cyber vandalism based attacks. |
| `ideology` | A passion to express a set of ideas, beliefs, and values that may shape and drive harmful and illegal acts.<br><br>Adversaries who act for ideological reasons (e.g., political, religious, human rights, environmental, desire to cause cause/anarchy, etc.) are not usually motivated primarily by the desire for profit; they are acting on their own sense of morality, justice, or political loyalty.<br><br>For example, an activist group may sabotage a company's equipment because they believe the company is harming the environment. |
| `notoriety` | Seeking prestige or to become well known through some activity.<br><br>Adversaries motivated by notoriety are often seeking either personal validation or respect within a community and staying covert is not a priority. In fact one of the main goals is to garner the respect of their target audience. |
| `organizational-gain` | Seeking advantage over a competing organization, including a military organization.<br><br>Adversaries motivated by increased profit or other gains through an unfairly obtained competitive advantage are often seeking theft of intellectual property, business processes, or supply chain agreements and thus accelerating their position in a market or capability. |
| `personal-gain` | The desire to improve one's own financial status.<br><br>Adversaries motivated by a selfish desire for personal gain are often out for gains that come from financial fraud, hacking for hire, or intellectual property theft.<br><br>While a Threat Actor or Intrusion Set may be seeking personal gain this does not mean they are acting alone. Individuals can band together solely to maximize their own personal profits. |
| `personal-satisfaction` | A desire to satisfy a strictly personal goal, including curiosity, thrill-seeking, amusement, etc. |

| | |
|---|---|
| | Threat Actors or Intrusion Set driven by personal satisfaction may incidentally receive some other gain from their actions, such as a profit, but their primary motivation is to gratify a personal, emotional need. Individuals can band together with others toward a mutual, but not necessarily organizational, objective. |
| `revenge` | A desire to avenge perceived wrongs through harmful actions such as sabotage, violence, theft, fraud, or embarrassing certain individuals or the organization.<br><br>A disgruntled Threat Actor or Intrusion Set seeking revenge can include current or former employees, who may have extensive knowledge to leverage when conducting attacks. Individuals can band together with others if the individual believes that doing so will enable them to cause more harm. |
| `unpredictable` | Acting without identifiable reason or purpose and creating unpredictable events.<br><br>Unpredictable is not a miscellaneous or default category. Unpredictable means a truly random and likely bizarre event, which seems to have no logical purpose to the victims. |

## 8.2. Attack Resource Level

**Type Name:** `attack-resource-level-ov`

The attack resource level vocabulary is currently used in the following SDO(s):

- Intrusion Set
- Threat Actor

Attack Resource Level is an open vocabulary that captures the general level of resources that a threat actor, intrusion set, or campaign might have access to. It ranges from individual, a person acting alone, to government, the resources of a national government.

This section including vocabulary items and their descriptions is based on the *Threat Agent Library* publication from Intel Corp in September 2007[3]

| **Vocabulary Summary** | |
|---|---|

[3] Intel Corp Threat Agent Library Sept 2007

|  |  |
|---|---|
| `individual`, `club`, `contest`, `team`, `organization`, `government` |  |

| **Vocabulary Value** | **Description** |
|---|---|
| `individual` | Resources limited to the average individual; Threat Actor acts independently. |
| `club` | Members interact on a social and volunteer basis, often with little personal interest in the specific target. An example might be a core group of unrelated activists who regularly exchange tips on a particular blog. Group persists long term. |
| `contest` | A short-lived and perhaps anonymous interaction that concludes when the participants have achieved a single goal. For example, people who break into systems just for thrills or prestige may hold a contest to see who can break into a specific target first. It also includes announced "operations" to achieve a specific goal, such as the original "OpIsrael" call for volunteers to disrupt all of Israel's Internet functions for a day. |
| `team` | A formally organized group with a leader, typically motivated by a specific goal and organized around that goal. Group persists long term and typically operates within a single geography. |
| `organization` | Larger and better resourced than a team; typically a company or crime syndicate. Usually operates in multiple geographic areas and persists long term. |
| `government` | Controls public assets and functions within a jurisdiction; very well resourced and persists long term. |

## 8.3. Identity Class

**Type Name:** `identity-class-ov`

The identity class vocabulary is currently used in the following SDO(s):

- Identity

This vocabulary describes the type of entity that the Identity represents: whether it describes an organization, group, individual, or class.

| **Vocabulary Summary** |  |
|---|---|
| `individual`, `group`, `organization`, `class`, `unknown` |  |

| Vocabulary Value | Description |
|---|---|
| `individual` | A single person. |
| `group` | An informal collection of people, without formal governance, such as a distributed hacker group. |
| `organization` | A formal organization of people, with governance, such as a company or country. |
| `class` | A class of entities, such as all hospitals, all Europeans, or the Domain Administrators in a system. |
| `unknown` | It is unknown whether the classification is individual, group, organization, or class. |

## 8.4. Indicator Label

**Type Name:** `indicator-label-ov`

The indicator label vocabulary is currently used in the following SDO(s):

- Indicator

Indicator labels is an open vocabulary used to categorize Indicators. It is intended to be high-level to promote consistent practices. Indicator labels should not be used to capture information that can be better captured via related Malware or Attack Pattern objects. It is better to link an Indicator to a Malware object describing Poison Ivy rather than simply labeling it with "poison-ivy".

| Vocabulary Summary | |
|---|---|
| `anomalous-activity`, `anonymization`, `benign`, `compromised`, `malicious-activity`, `attribution` | |
| **Vocabulary Value** | **Description** |
| `anomalous-activity` | Unexpected, or unusual activity that may not necessarily be malicious or indicate compromise. This type of activity may include reconnaissance-like behavior such as port scans or version identification, network behavior anomalies, and asset and/or user behavioral anomalies. |
| `anonymization` | Suspected anonymization tools or infrastructure (proxy, TOR, VPN, etc.). |

|  |  |
|---|---|
| `benign` | Activity that is not suspicious or malicious in and of itself, but when combined with other activity may indicate suspicious or malicious behavior. |
| `compromised` | Assets that are suspected to be compromised. |
| `malicious-activity` | Patterns of suspected malicious objects and/or activity. |
| `attribution` | Patterns of behavior that indicate attribution to a particular Threat Actor or Campaign. |

## 8.5. Industry Sector

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

The industry sector vocabulary is currently used in the following SDO(s):

- Identity

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

| **Vocabulary Summary** |  |
|---|---|
| `agriculture`, `aerospace`, `automotive`, `communications`, `construction`, `defence`, `education`, `energy`, `entertainment`, `financial-services`, `government-national`, `government-regional`, `government-local`, `government-public-services`, `healthcare`, `hospitality-leisure`, `infrastructure`, `insurance`, `manufacturing`, `mining`, `non-profit`, `pharmaceuticals`, `retail`, `technology`, `telecommunications`, `transportation`, `utilities` |  |
| **Vocabulary Value** | **Description** |
| `agriculture` |  |
| `aerospace` |  |
| `automotive` |  |
| `communications` |  |
| `construction` |  |
| `defense` |  |

| | |
|---|---|
| `education` | |
| `energy` | |
| `entertainment` | |
| `financial-services` | |
| `government-national` | |
| `government-regional` | |
| `government-local` | |
| `government-public-services` | emergency services, sanitation |
| `healthcare` | |
| `hospitality-leisure` | |
| `infrastructure` | |
| `insurance` | |
| `manufacturing` | |
| `mining` | |
| `non-profit` | |
| `pharmaceuticals` | |
| `retail` | |
| `technology` | |
| `telecommunications` | |
| `transportation` | |
| `utilities` | |

## 8.6. Malware Label

**Type Name:** `malware-label-ov`

The malware label vocabulary is currently used in the following SDO(s):

- Malware

Malware label is an open vocabulary that represents different types and functions of malware. Malware labels are not mutually exclusive: a malware instance can be both spyware and a screen capture tool.

| **Vocabulary Summary** | |
|---|---|
| `adware`, `backdoor`, `bot`, `ddos`, `dropper`, `exploit-kit`, `keylogger`, `ransomware`, `remote-access-trojan`, `resource-exploitation`, `rogue-antivirus`, `rootkit`, `screen-capture`, `spyware`, `trojan`, `virus`, `worm` | |
| **Vocabulary Value** | **Description** |
| `adware` | Any software that is funded by advertising. Adware may also gather sensitive user information from a system. |
| `backdoor` | A malicious program that allows an attacker to perform actions on a remote system, such as transferring files, acquiring passwords, or executing arbitrary commands [TODO: Ref NIST). |
| `bot` | A program that resides on an infected system, communicating with and forming part of a botnet. The bot may be implanted by a worm or Trojan, which opens a backdoor. The bot then monitors the backdoor for further instructions. |
| `ddos` | A tool used to perform a distributed denial of service attack. |
| `dropper` | A type of trojan that deposits an enclosed payload (generally, other malware) onto the target computer. |
| `exploit-kit` | A software toolkit to target common vulnerabilities. |
| `keylogger` | A type of malware that surreptitiously monitors keystrokes and either records them for later retrieval or sends them back to a central collection point. |
| `ransomware` | A type of malware that encrypts files on a victim's system, demanding payment of ransom in return for the access codes required to unlock files. |
| `remote-access-trojan` | A remote access trojan program (or RAT), is a trojan horse capable of controlling a machine through commands issued by a remote attacker. |
| `resource-exploitation` | A type of malware that steals a system's resources (e.g., CPU cycles), such as a bitcoin miner. |

| | |
|---|---|
| `rogue-security-software` | A fake security product that demands money to clean phony infections. |
| `rootkit` | A type of malware that hides its files or processes from normal methods of monitoring in order to conceal its presence and activities. Rootkits can operate at a number of levels, from the application level — simply replacing or adjusting the settings of system software to prevent the display of certain information — through hooking certain functions or inserting modules or drivers into the operating system kernel, to the deeper level of firmware or virtualization rootkits, which are activated before the operating system and thus even harder to detect while the system is running. |
| `screen-capture` | A type of malware used to capture images from the target systems screen, used for exfiltration and command and control. |
| `spyware` | Software that gathers information on a user's system without their knowledge and sends it to another party. Spyware is generally used to track activities for the purpose of delivering advertising. |
| `trojan` | Any malicious computer program which is used to hack into a computer by misleading users of its true intent. |
| `virus` | A malicious computer program that replicates by reproducing itself or infecting other programs by modifying them. |
| `worm` | A self-replicating, self-contained program that usually executes itself without user intervention. |

## 8.7. Pattern Language

**Type Name:** `pattern-lang-ov`

The pattern language vocabulary is currently used in the following SDO(s):

- Indicator

Pattern language is an open vocabulary that describes the different types of pattern languages that can be used in a STIX Indicator.

| **Vocabulary Summary** | |
|---|---|

| cybox, openioc, snort, yara | |
|---|---|
| **Vocabulary Value** | **Description** |
| cybox | A CybOX Patterning v1.0 [TODO Ref] pattern. cybox is the default value. |
| openioc | An OpenIOC pattern (REF). |
| snort | A Snort pattern (REF). |
| yara | A YARA pattern (REF). |

## 8.8. Report Label

**Type Name:** report-label-ov

The report label vocabulary is currently used in the following SDO(s):

- Report

Report label is an open vocabulary to describe the primary purpose or subject of a report. For example, a report that contains malware and indicators for that malware should have a report label of malware to capture that the malware is the primary purpose. Report labels are not mutually exclusive: a Report can be both a malware report and a tool report. Just because a report contains objects of a type does not mean that the report should include that label. If the objects are there to simply provide evidence or context for other objects, it is not necessary to include them in the label.

| **Vocabulary Summary** | |
|---|---|
| threat-report, attack-pattern, campaign, indicator, malware, observed-data, threat-actor, tool, victim-target, vulnerability | |
| **Vocabulary Value** | **Description** |
| threat-report | Report subject is a broad characterization of a threat across multiple facets. |
| attack-pattern | Report subject is a characterization of one or more attack patterns and related information. |
| campaign | Report subject is a characterization of one or more campaigns and related information. |

| | |
|---|---|
| `indicator` | Report subject is a characterization of one or more indicators and related information. |
| `intrusion-set` | Report subject is a characterization of one or more intrusion sets and related information. |
| `malware` | Report subject is a characterization of one or more malware instances and related information. |
| `observed-data` | Report subject is a characterization of observed data and related information. |
| `threat-actor` | Report subject is a characterization of one or more threat actors and related information. |
| `tool` | Report subject is a characterization of one or more tools and related information. |
| `victim-target` | Report subject is a characterization of one or more victim targets and related information. |
| `vulnerability` | Report subject is a characterization of one or more vulnerabilities and related information. |

## 8.9. Threat Actor Label

**Type Name:** `threat-actor-label-ov`

The threat actor label vocabulary is currently used in the following SDO(s):

- Threat Actor

Threat actor label is an open vocabulary used to describe what type of threat actor the individual or group is. For example, some threat actors are competitors who try to steal information, while others are activists who act in support of a social or political cause. Actor labels are not mutually exclusive: a threat actor can be both a disgruntled insider and a spy.

[REF: Threat Agent Library, Intel Corporation, September 2007]

| Vocabulary Summary | |
|---|---|
| `activist`, `competitor`, `crime-syndicate`, `criminal`, `hacker`, `insider-accidental`, `insider-disgruntled`, `nation-state`, `sensationalist`, `spy`, `terrorist` | |

| Vocabulary Value | Description |
|---|---|
| `activist` | Highly motivated, potentially destructive supporter of a social or political cause (e.g., trade, labor, environment, etc) that attempt to disrupt an organization's business model or damage their image.<br><br>This category includes actors sometimes referred to as anarchists, cyber vandals, extremists, and hacktivists. |
| `competitor` | An organization that competes in the same economic marketplace.<br><br>The goal of a competitor is to gain an advantage in business with respect to the rival organization it targets. It usually does this by copying intellectual property, trade secrets, acquisition strategies, or other technical or business data from a rival organization with the intention of using the data to bolster its own assets and market position. |
| `crime-syndicate` | An enterprise organized to conduct significant, large-scale criminal activity for profit.<br><br>Crime syndicates, also known as organized crime, are generally large, well-resourced groups that operate to create profit from all types of crime. |
| `criminal` | Individual who commits computer crimes, often for personal financial gain and often involves the theft of something valuable.<br><br>Intellectual property theft, extortion via ransomware, and physical destruction are common examples. A criminal as defined here refers to those acting individually or in very small or informal groups. For sophisticated organized criminal activity, see the crime syndicate descriptor. |
| `hacker` | An individual that tends to break into networks for the thrill or the challenge of doing so.<br><br>Hackers may use advanced skills or simple attack scripts they have downloaded. |
| `insider-accidental` | A non-hostile insider who unintentionally exposes the organization to harm.<br><br>“Insider” in this context includes any person extended internal trust, such as regular employees, contractors, consultants, and temporary workers. |

| | |
|---|---|
| `insider-disgruntled` | Current or former insiders who seek revengeful and harmful retaliation for perceived wrongs.<br><br>“Insider” in this context includes any person extended internal trust, such as regular employees, contractors, consultants, and temporary workers.<br><br>Disgruntled threat actors may have extensive knowledge that can be leveraged when conducting attacks and can take any number of actions including sabotage, violence, theft, fraud, espionage, or embarrassing individuals or the organization. |
| `nation-state` | Entities who work for the government or military of a nation state or who work at their direction.<br><br>These actors typically have access to significant support, resources, training, and tools and are capable of designing and executing very sophisticated and effective Intrusion Sets and Campaigns. |
| `sensationalist` | Seeks to cause embarrassment and brand damage by exposing sensitive information in a manner designed to cause a public relations crisis.<br><br>A Sensationalist may be an individual or small group of people motivated primarily by a need for notoriety. Unlike the Activist, the Sensationalist generally has no political goal, and is not using bad PR to influence the target to change its behavior or business practices. |
| `spy` | Secretly collects sensitive information for use, dissemination, or sale.<br><br>Traditional spies (governmental and industrial) are part of a well-resourced intelligence organization and are capable of very sophisticated clandestine operations. However, insiders such as employees or consultants acting as spies can be just as effective and damaging, even when their activities are largely opportunistic and not part of an overall campaign. |
| `terrorist` | Uses extreme violence to advance a social or political agenda as well as monetary crimes to support its activities.<br><br>In this context a terrorist refers to individuals who target noncombatants with violence to send a message of fear far beyond the actual events. They may act independently or as part of a terrorist organization. |

| | |
|---|---|
| | Terrorist organizations must typically raise much of their operating budget through criminal activity, which often occurs online. Terrorists are also often adept at using and covertly manipulating social media for both recruitment and impact. |

## 8.10. Threat Actor Role

**Type Name:** `threat-actor-role-ov`

The threat actor role vocabulary is currently used in the following SDO(s):

- Threat Actor

Threat actor role is an open vocabulary that is used to describe the different roles that a threat actor can play. For example, some threat actors author malware or operate botnets while other actors actually carry out attacks directly.

Threat actor roles are not mutually exclusive. For example, an actor can be both a financial backer for attacks and also direct attacks.

| **Vocabulary Summary** | |
|---|---|
| `agent`, `director`, `independent`, `infrastructure-architect`, `infrastructure-operator`, `malware-author`, `sponsor` | |
| **Vocabulary Value** | **Description** |
| `agent` | Threat actor executes attacks either on behalf of themselves or at the direction of someone else. |
| `director` | The threat actor who directs the activities, goals, and objectives of the malicious activities. |
| `independent` | A threat actor acting by themselves. |
| `infrastructure-architect` | Someone who designs the battle space <TODO> |
| `infrastructure-operator` | The threat actor who provides and supports the attack infrastructure that is used to deliver the attack (botnet providers, cloud services, etc.). |
| `malware-author` | The threat actor who authors malware or other malicious tools. |

|  |  |
|---|---|
| sponsor | The threat actor who funds the malicious activities. |

## 8.11. Threat Actor Sophistication

**Type Name:** `attack-sophistication-level-ov`

Threat actor sophistication vocabulary is currently used in the following SDO(s):

- Threat Actor

Threat actor sophistication vocabulary captures the skill level of a threat actor. It ranges from "none", which describes a complete novice, to "strategic", which describes an attacker who is able to influence supply chains to introduce vulnerabilities. This vocabulary is separate from resource level because an innovative, highly-skilled threat actor may have access to very few resources while a minimal-level actor might have the resources of an organized crime ring.

| **Vocabulary Summary** |  |
|---|---|
| `none`, `minimal`, `intermediate`, `advanced`, `expert`, `innovator`, `strategic` |  |
| **Vocabulary Value** | **Description** |
| `none` | Can carry out random acts of disruption or destruction by running tools they do not understand. Actors in this category have average computer skills.<br><br>Example Roles: Average User<br><br>These actors:<br>● can not launch targeted attacks |
| `minimal` | Can minimally use existing and frequently well known and easy-to-find techniques and programs or scripts to search for and exploit weaknesses in other computers. Commonly referred to as a script-kiddie.<br><br>These actors rely on others to develop the malicious tools, delivery mechanisms, and execution strategy and often do not fully understand the tool they are using or how they work. They also lack the ability to conduct their own reconnaissance and targeting research.<br><br>Example Roles: Script-Kiddie<br><br>These actors: |

| | |
|---|---|
| | ● attack known weaknesses;<br>● use well known scripts and tools; and<br>● have minimal knowledge of the tools. |
| `intermediate` | Can proficiently use existing attack frameworks and toolkits to search for and exploit vulnerabilities in computers or systems. Actors in this category have computer skills equivalent to an IT professional and typically have a working knowledge of networks, operating systems, and possibly even defensive techniques and will typically exhibit some operational security.<br><br>These actors rely others to develop the malicious tools and delivery mechanisms, but are able to plan their own execution strategy. They are proficient in the tools they are using and how they work and can even make minimal modifications as needed.<br><br>Example Roles: Toolkit User<br><br>These actors:<br>● attack known vulnerabilities;<br>● use attack frameworks and toolkits; and<br>● have proficient knowledge of the tools. |
| `advanced` | Can develop their own tools or scripts from publicly known vulnerabilities to target systems and users. Actors in this category are very adept at IT systems and have a background in software development along with a solid understanding of defensive techniques and operational security.<br><br>These actors rely on others to find and identify weaknesses and vulnerabilities in systems, but are able to create their own tools, delivery mechanisms, and execution strategies.<br><br>Example Roles: Toolkit Developer<br><br>These actors:<br>● attack known vulnerabilities;<br>● can create their own tools; and<br>● have proficient knowledge of the tools. |
| `expert` | Can focus on the discovery and use of unknown malicious code, are is adept at installing user and kernel mode rootkits, frequently use data mining tools, target corporate executives and key users (government and industry) for the purpose of stealing personal and corporate data. Actors in this category are very adept at IT systems and software development and are experts with security systems, defensive techniques, attack methods, and operational security.<br><br>Example Roles: Vulnerability Researcher, Reverse Engineer, Threat Researcher, Malware Creator |

| | |
|---|---|
| | These actors:<br>● attack unknown and known vulnerabilities;<br>● can create their own tools from scratch; and<br>● have proficient knowledge of the tools. |
| `innovator` | Typically a criminal or state actors who are organized, highly technical, proficient, well funded professionals working in teams to discover new vulnerabilities and develop exploits.<br><br>Demonstrates sophisticated capability. An innovator has the ability to create and script unique programs and codes targeting virtually any form of technology. At this level, this actor has a deep knowledge of networks, operating systems, programming languages, firmware, and infrastructure topologies and will demonstrate operational security when conducting his activities. Innovators are largely responsible for the discovery of 0-day vulnerabilities and the development of new attack techniques.<br><br>Example Roles: Toolkit Innovator, 0-Day Exploit Author<br><br>These actors:<br>● attack unknown and known vulnerabilities;<br>● creates attacks against 0-Day exploits from scratch; and<br>● creates new and innovative attacks and toolkits. |
| `strategic` | State actors who create vulnerabilities through an active program to “influence” commercial products and services during design, development or manufacturing, or with the ability to impact products while in the supply chain to enable exploitation of networks and systems of interest.<br><br>These actors:<br>● can create or use entire supply chains to launch an attack;<br>● can create and design attacks for any systems, software package, or device; and<br>● are responsible for APT level attacks. |

## 8.12. Tool Label

**Type Name:** `tool-label-ov`

The tool label vocabulary is currently used in the following SDO(s):

- Tool

Tool labels describe the categories of tools that can be used to perform attacks.

| Vocabulary Summary | |
|---|---|
| `denial-of-service`, `exploitation`, `information-gathering`, `network-capture`, `credential-exploitation`, `remote-access`, `vulnerability-scanning` | |
| **Vocabulary Value** | **Description** |
| `denial-of-service` | Tools used to perform denial of service attacks or DDoS attacks, such as Low Orbit Ion Cannon (LOIC) and DHCPig. |
| `exploitation` | Tools used to exploit software and systems, such as sqlmap and Metasploit. |
| `information-gathering` | Tools used to enumerate system and network information, e.g., NMAP. |
| `network-capture` | Tools used to capture network traffic, such as Wireshark and Kismet. |
| `credential-exploitation` | Tools used to crack password databases or otherwise exploit/discover credentials, either locally or remotely, such as John the Ripper and NCrack. |
| `remote-access` | Tools used to access machines remotely, such as VNC and Remote Desktop. |
| `vulnerability-scanning` | Tools used to scan systems and networks for vulnerabilities, e.g., Nessus. |

# 9. Customizing STIX

There are two primary means to customize STIX: Custom Properties, and Custom Objects. Custom Properties provides a mechanism and requirements for adding properties not defined by this specification to existing STIX Objects. Custom Objects, on the other hand, provides a mechanism and requirements to create custom STIX Objects (objects not defined by this specification).

A consumer that receives a STIX document containing Custom Properties or Objects it does not understand **MAY** refuse to process the document or **MAY** ignore those properties or objects and continue processing the document.

Producers of STIX documents that contain Custom Properties or Objects should recognize that consumers may not understand them and may ignore them. Producers should define any

Custom Properties and Objects they use, along with any rules for processing them, and make these definitions and rules accessible to any potential consumers. This specification does not specify a process for doing this.

## 9.1. Custom Properties

There will be cases where certain information exchanges can be improved by adding properties that are neither specified nor reserved in this document; these properties are called **Custom Properties**. This section provides guidance and requirements for how producers can use Custom Properties and how consumers should interpret them in order to extend STIX in an interoperable manner.

### 9.1.1. Requirements

- A STIX Object **MAY** have any number of Custom Properties.
- Custom Property names **MUST** be in ASCII and **MUST** only contain the characters a–z (lowercase ASCII), 0–9, and underscore (_).
- Custom Property names **SHOULD** start with "x_" followed by a source unique identifier (such as a domain name with dots replaced by underscores), an underscore and then the name. For example, **`x_example_com_customfield`**.
- Custom Property names **MUST** have a minimum length of 3 ASCII characters.
- Custom Property names **MUST** be no longer than 250 ASCII characters in length.
- Custom Property names that do not start with "x_" may be used in a future version of the specification for a different meaning. If compatibility with future versions of this specification is required, the "x_" prefix **MUST** be used.
- Custom Properties **SHOULD** only be used when there is no existing properties defined by the STIX specification that fulfils that need.

### 9.1.2. Examples

```
{
  ...,
  "x_acme_org_confidence: 10,
  "x_acme_org_scoring": {
    "impact": "high",
    "probability": "low"
  },
  ...
}
```

## 9.2. Custom Objects

There will be cases where certain information exchanges can be improved by adding objects that are not specified nor reserved in this document; these objects are called **Custom Objects**.

This section provides guidance and requirements for how producers can use Custom Objects and how consumers should interpret them in order to extend STIX in an interoperable manner.

### 9.2.1. Requirements

- Producers **MAY** include any number of Custom Objects in STIX documents.
- Custom Objects **MUST** contain the required Common Properties (**`id`**, **`type`**, **`version`**, **`modified`**, **`created`**, **`created_by_ref`**) and **MAY** contain any optional Common Property (defined in Section TODO).
    - The definitions of these properties are the same as those defined in Common Properties and therefore those fields **MUST NOT** be used to represent the custom properties in the object.
- The **type** field in a Custom Object **MUST** be in ASCII and **MUST** only contain the characters a–z (lowercase ASCII), 0–9, and hyphen (**-**).
- The **type** field **MUST NOT** contain a hyphen (-) character immediately following another hyphen (-) character.
- Custom Object names **MUST** have a minimum length of 3 ASCII characters.
- Custom Object names **MUST** be no longer than 250 ASCII characters in length.
- The value of the **`type`** field in a Custom Object **SHOULD** start with “x-” followed by a source unique identifier (like a domain name with dots replaced by dashes), a dash and then the name. For example, `x-example-com-customobject`.
- A Custom Object whose name is not prefixed with “x-” may be used in a future version of the specification with a different meaning. Therefore, if compatibility with future versions of this specification is required, the “x-” prefix **MUST** be used.
- The value of the **`id`** property in a Custom Object **MUST** use the same format as the `identifier` type, namely, name--uuid.
- Custom Objects **SHOULD** only be used when there is no existing STIX Object defined by the STIX specification that fulfils that need.

### 9.2.2. Examples

```
{
  "type": "bundle",
  "id": "bundle--f37aa79d-f5f5-4af7-874b-734d32c08c10",
  "custom_objects": [
    {
      "type": "x-example-com-customobject",
      "id": "x-example-com-customobject--4527e5de-8572-446a-a57a-706f15467461",
      "created": "2016-08-01T00:00:00Z",
      "modified": "2016-08-01T00:00:00Z",
      "version": 1,
      "some_custom_stuff": 14,
      "other_custom_stuff": "hello"
    }
  ]
}
```

# 10. Conformance

# 11. Appendix A. Acknowledgments

**STIX Subcommittee Chairs**
John Wunder (jwunder@mitre.org), MITRE Corporation
Aharon Chernin (achernin@soltra.com), Soltra

**Special Thanks**
The following individuals made substantial contributions to this specification in the form of normative text and proofing and their contributions are gratefully acknowledged:

- Bret Jordan, Blue Coat Systems, Inc.
- Terry MacDonald, Cosive
- Jane Ginn, Cyber Threat Intelligence Network, Inc. (CTIN)
- Richard Struse, DHS Office of Cybersecurity and Communications
- Jason Keirstead, IBM
- Tim Casey, Intel
- Allan Thomson, LookingGlass Cyber
- Jon Baker, MITRE Corporation
- John Wunder, MITRE Corporation
- Richard Piazza, MITRE Corporation
- John-Mark Gurney, New Context Services, Inc.
- Iain Brown, United Kingdom Cabinet Office

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

- David Crawford, Aetna
- Marcos Orallo, Airbus Group SAS
- Roman Fiedler, AIT Austrian Institute of Technology
- Florian Skopik, AIT Austrian Institute of Technology
- Ryan Clough, Anomali
- Wei Huang, Anomali
- Hugh Njemanze, Anomali
- Katie Pelusi, Anomali
- Aaron Shelmire, Anomali

- Jason Trost, Anomali
- Dean Thompson, Australia and New Zealand Banking Group (ANZ Bank)
- Alexander Foley, Bank of America
- Tony Pham, Bank of America
- Gautam Aggarwal, Bay Dynamics
- Humphrey Christian, Bay Dynamics
- Anil Nandigam, Bay Dynamics
- Ryan Stolte, Bay Dynamics
- Owen Johnson, Blue Coat Systems, Inc.
- Bret Jordan, Blue Coat Systems, Inc.
- Sarah Kelley, Center for Internet Security (CIS)
- Cory Kennedy, CenturyLink
- Alexandre Dulaunoy, CIRCL
- Andras Iklody, CIRCL
- Raphaël Vinot, CIRCL
- Syam Appala, Cisco Systems
- Ted Bedwell, Cisco Systems
- David McGrew, Cisco Systems
- Pavan Reddy, Cisco Systems
- Omar Santos, Cisco Systems
- Jyoti Verma, Cisco Systems
- Joey Peloquin, Citrix Systems
- Doug DePeppe, Cyber Threat Intelligence Network, Inc. (CTIN)
- Jane Ginn, Cyber Threat Intelligence Network, Inc. (CTIN)
- Ben Othman, Cyber Threat Intelligence Network, Inc. (CTIN)
- Will Urbanski, Dell
- Jeff Williams, Dell
- Sean Sobieraj, DHS Office of Cybersecurity and Communications (CS&C)
- Richard Struse, DHS Office of Cybersecurity and Communications (CS&C)
- Marlon Taylor, DHS Office of Cybersecurity and Communications (CS&C)
- Wouter Bolsterlee, EclecticIQ
- Marko Dragoljevic, EclecticIQ
- Joep Gommers, EclecticIQ
- Sergey Polzunov, EclecticIQ
- Rutger Prins, EclecticIQ
- Andrei Sîrghi, EclecticIQ
- Raymon van der Velde, EclecticIQ
- Robert Griffin, EMC
- Jeff Odom, EMC
- Sreejith Padmajadevi, EMC
- Ravi Sharda, EMC
- David Eilken, Financial Services Information Sharing and Analysis Center (FS-ISAC)
- Chris Ricard, Financial Services Information Sharing and Analysis Center (FS-ISAC)
- Phillip Boles, FireEye, Inc.

- Prasad Gaikwad, FireEye, Inc.
- Pavan Gorakav, FireEye, Inc.
- Pavan Gorakavi, FireEye, Inc.
- Rajeev Jha, FireEye, Inc.
- Anuj Kumar, FireEye, Inc.
- Shyamal Pandya, FireEye, Inc.
- Paul Patrick, FireEye, Inc.
- Scott Shreve, FireEye, Inc.
- Gavin Chow, Fortinet Inc.
- Steve Fossen, Fortinet Inc.
- Kenichi Terashita, Fortinet Inc.
- Neil Edwards, Fujitsu Limited
- Ryusuke Masuoka, Fujitsu Limited
- Daisuke Murabayashi, Fujitsu Limited
- Derek Northrope, Fujitsu Limited
- Robert van Engelen, Genivia
- Eric Burger, Georgetown University
- Mark Risher, Google Inc.
- Richard Austin, Hewlett Packard Enterprise (HPE)
- Tomas Sander, Hewlett Packard Enterprise (HPE)
- Jun Nakanishi, Hitachi, Ltd.
- Yukari Nishikawa, Hitachi, Ltd.
- Kazuo Noguchi, Hitachi, Ltd.
- Akihito Sawada, Hitachi, Ltd.
- Yutaka Takami, Hitachi, Ltd.
- Masato Terada, Hitachi, Ltd.
- Peter Allor, IBM
- Eldan Ben-Haim, IBM
- Sandra Hernandez, IBM
- Jason Keirstead, IBM
- John Morris, IBM
- Laura Rusu, IBM
- Ron Williams, IBM
- Paul Martini, iboss, Inc.
- Jerome Athias, Individual
- Peter Brown, Individual
- Joerg Eschweiler, Individual
- Elysa Jones, Individual
- Sanjiv Kalkar, Individual
- Terry MacDonald, Individual
- Patrick Maroney, Individual
- Alex Pinto, Individual
- Tim Casey, Intel Corporation
- Kent Landfield, Intel Corporation

- Karin Marr, Johns Hopkins University Applied Physics Laboratory
- Julie Modlin, Johns Hopkins University Applied Physics Laboratory
- Mark Moss, Johns Hopkins University Applied Physics Laboratory
- Mark Munoz, Johns Hopkins University Applied Physics Laboratory
- Pamela Smith, Johns Hopkins University Applied Physics Laboratory
- Terrence Driscoll, JPMorgan Chase Bank, N.A.
- David Laurance, JPMorgan Chase Bank, N.A.
- Russell Culpepper, Kaiser Permanente
- Beth Pumo, Kaiser Permanente
- Michael Slavick, Kaiser Permanente
- Trey Darley, Kingfisher Operations, sprl
- Jacob Hinkle, LexisNexis, a Division of Reed Elsevier
- Kinshuk Pahare, LookingGlass
- Allan Thomson, LookingGlass
- Ian Truslove, LookingGlass
- Lee Vorthman, LookingGlass
- Chris Wood, LookingGlass
- Greg Back, MITRE Corporation
- Jonathan Baker, MITRE Corporation
- Sean Barnum, MITRE Corporation
- Desiree Beck, MITRE Corporation
- Nicole Gong, MITRE Corporation
- Jasen Jacobsen, MITRE Corporation
- Ivan Kirillov, MITRE Corporation
- Richard Piazza, MITRE Corporation
- Jon Salwen, MITRE Corporation
- Charles Schmidt, MITRE Corporation
- Emmanuelle Vargas-Gonzalez, MITRE Corporation
- John Wunder, MITRE Corporation
- James Cabral, MTG Management Consultants, LLC.
- Scott Algeier, National Council of ISACs (NCI)
- Denise Anderson, National Council of ISACs (NCI)
- Josh Poster, National Council of ISACs (NCI)
- Mike Boyle, National Security Agency
- Jessica Fitzgerald-McKay, National Security Agency
- David Kemp, National Security Agency
- Takahiro Kakumaru, NEC Corporation
- John-Mark Gurney, New Context Services, Inc.
- Christian Hunt, New Context Services, Inc.
- James Moler, New Context Services, Inc.
- Daniel Riedel, New Context Services, Inc.
- Andrew Storms, New Context Services, Inc.
- David Darnell, North American Energy Standards Board
- Cory Casanave, Object Management Group

- Don Thibeau, Open Identity Exchange
- Johnny Gau, Oracle
- Sunil Ravipati, Oracle
- Josh Larkins, PhishMe Inc.
- John Tolbert, Queralt, Inc.
- Daniel Wyschogrod, Raytheon Company-SAS
- Ted Julian, Resilient Systems, Inc..
- Igor Baikalov, Securonix
- Joseph Brand, Semper Fortis Solutions
- Bernd Grobauer, Siemens AG
- John Anderson, Soltra
- Aishwarya Asok Kumar, Soltra
- Peter Ayasse, Soltra
- Jeff Beekman, Soltra
- Michael Butt, Soltra
- Cynthia Camacho, Soltra
- Aharon Chernin, Soltra
- Mark Clancy, Soltra
- Mark Davidson, Soltra
- Paul Dion, Soltra
- Daniel Dye, Soltra
- Robert Hutto, Soltra
- Raymond Keckler, Soltra
- Ali Khan, Soltra
- Chris Kiehl, Soltra
- Clayton Long, Soltra
- Michael Pepin, Soltra
- Natalie Suarez, Soltra
- David Waters, Soltra
- Benjamin Yates, Soltra
- Dave Cridland, Surevine Ltd.
- Tom Blauvelt, Symantec Corp.
- Robert Keith, Symantec Corp.
- Curtis Kostrosky, Symantec Corp.
- Juha Haaga, Synopsys
- Greg Reaume, TELUS
- Alan Steer, TELUS
- Crystal Hayes, The Boeing Company
- Wade Baker, ThreatConnect, Inc.
- Cole Iliff, ThreatConnect, Inc.
- Andrew Pendergast, ThreatConnect, Inc.
- Ben Schmoker, ThreatConnect, Inc.
- Jason Spies, ThreatConnect, Inc.
- Ryan Trost, ThreatQuotient, Inc.

- Chris Roblee, TruSTAR Technology
- Mark Angel, U.S. Bank
- Brian Fay, U.S. Bank
- Mark Heidrick, U.S. Bank
- Mona Magathan, U.S. Bank
- Yevgen Sautin, U.S. Bank
- Jonathan Algar, United Kingdom Cabinet Office
- Iain Brown, United Kingdom Cabinet Office
- Adam Cooper, United Kingdom Cabinet Office
- Mike McLellan, United Kingdom Cabinet Office
- Tyrone Nembhard, United Kingdom Cabinet Office
- Chris O'Brien, United Kingdom Cabinet Office
- James Penman, United Kingdom Cabinet Office
- Howard Staple, United Kingdom Cabinet Office
- Chris Taylor, United Kingdom Cabinet Office
- Laurie Thomson, United Kingdom Cabinet Office
- Alastair Treharne, United Kingdom Cabinet Office
- Julian White, United Kingdom Cabinet Office
- Bethany Yates, United Kingdom Cabinet Office
- James Bohling, US Department of Defense (DoD)
- Eoghan Casey, US Department of Defense (DoD)
- Gary Katz, US Department of Defense (DoD)
- Jeffrey Mates, US Department of Defense (DoD)
- Evette Maynard-Noel, US Department of Homeland Security
- Justin Stekervetz, US Department of Homeland Security
- Robert Coderre, VeriSign
- Kyle Maxwell, VeriSign
- Eric Osterweil, VeriSign

# 12. Appendix B. Revision History