STIX™ Version 2.0. Part 1: STIX Core Concepts

Committee Specification Draft 02 / Public Review Draft 02

03 May 2017

Specification URIs

This version:

http://docs.oasis-open.org/cti/stix/v2.0/csprd02/part1-stix-core/stix-v2.0-csprd02-part1-stix-core.docx (Authoritative)
http://docs.oasis-open.org/cti/stix/v2.0/csprd02/part1-stix-core/stix-v2.0-csprd02-part1-stix-core.html
http://docs.oasis-open.org/cti/stix/v2.0/csprd02/part1-stix-core/stix-v2.0-csprd02-part1-stix-core.pdf

Previous version:

http://docs.oasis-open.org/cti/stix/v2.0/csprd01/part1-stix-core/stix-v2.0-csprd01-part1-stix-core.docx (Authoritative)
http://docs.oasis-open.org/cti/stix/v2.0/csprd01/part1-stix-core/stix-v2.0-csprd01-part1-stix-core.html
http://docs.oasis-open.org/cti/stix/v2.0/csprd01/part1-stix-core/stix-v2.0-csprd01-part1-stix-core.pdf

Latest version:

http://docs.oasis-open.org/cti/stix/v2.0/stix-v2.0-part1-stix-core.docx (Authoritative)
http://docs.oasis-open.org/cti/stix/v2.0/stix-v2.0-part1-stix-core.html
http://docs.oasis-open.org/cti/stix/v2.0/stix-v2.0-part1-stix-core.pdf

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:

Rich Piazza (rpiazza@mitre.org), MITRE Corporation
John Wunder (jwunder@mitre.org), MITRE Corporation
Bret Jordan (bret_jordan@symantec.com), Symantec Corp.

Additional artifacts:

This prose specification is one component of a Work Product that also includes:

- (this document) *STIX™ Version 2.0. Part 1: STIX Core Concepts*. http://docs.oasis-open.org/cti/stix/v2.0/csprd02/part1-stix-core/stix-v2.0-csprd02-part1-stix-core.html.
- *STIX™ Version 2.0. Part 2: STIX Objects.* http://docs.oasis-open.org/cti/stix/v2.0/csprd02/part2-stix-objects/stix-v2.0-csprd02-part2-stix-objects.html.
- *STIX™ Version 2.0. Part 3: Cyber Observable Core Concepts.* http://docs.oasis-open.org/cti/stix/v2.0/csprd02/part3-cyber-observable-core/stix-v2.0-csprd02-part3-cyber-observable-core.html.

- *STIX™ Version 2.0. Part 4: Cyber Observable Objects.* http://docs.oasis-open.org/cti/stix/v2.0/csprd02/part4-cyber-observable-objects/stix-v2.0-csprd02-part4-cyber-observable-objects.html.
- *STIX™ Version 2.0. Part 5: STIX Patterning.* http://docs.oasis-open.org/cti/stix/v2.0/csprd02/part5-stix-patterning/stix-v2.0-csprd02-part5-stix-patterning.html.

Related work:

This specification replaces or supersedes:

- *STIX™ Version 1.2.1. Part 1: Overview.* Edited by Sean Barnum, Desiree Beck, Aharon Chernin, and Rich Piazza. Latest version: http://docs.oasis-open.org/cti/stix/v1.2.1/stix-v1.2.1-part1-overview.html.
- *CybOX™ Version 2.1.1. Part 01: Overview.* Edited by Trey Darley, Ivan Kirillov, Rich Piazza, and Desiree Beck. Latest version: http://docs.oasis-open.org/cti/cybox/v2.1.1/cybox-v2.1.1-part01-overview.html.

This specification is related to:

- *TAXII™ Version 2.0.* Edited by John Wunder, Mark Davidson, and Bret Jordan. Latest version: http://docs.oasis-open.org/cti/taxii/v2.0/taxii-v2.0.html.

Abstract:

Structured Threat Information Expression (STIX™) is a language for expressing cyber threat and observable information. This document defines concepts that apply across all of STIX and defines the overall structure of the STIX language.

Status:

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

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

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

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

Citation format:

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

[STIX-v2.0-Pt1-Core]

STIX™ Version 2.0. Part 1: STIX Core Concepts. Edited by Rich Piazza, John Wunder, and Bret Jordan. 03 May 2017. OASIS Committee Specification Draft 02 / Public Review Draft 02. http://docs.oasis-open.org/cti/stix/v2.0/csprd02/part1-stix-core/stix-v2.0-csprd02-part1-stix-core.html. Latest version: http://docs.oasis-open.org/cti/stix/v2.0/stix-v2.0-part1-stix-core.html.

Notices

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

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

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

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

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

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

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

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

STIX™, CYBOX™, AND TAXII™ (STANDARD OR STANDARDS) AND THEIR COMPONENT PARTS ARE PROVIDED "AS IS" WITHOUT ANY WARRANTY OF ANY KIND, EITHER EXPRESSED, IMPLIED, OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, ANY WARRANTY THAT THESE STANDARDS OR ANY OF THEIR COMPONENT PARTS WILL CONFORM TO SPECIFICATIONS, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR FREEDOM FROM INFRINGEMENT, ANY WARRANTY THAT THE STANDARDS OR THEIR COMPONENT PARTS

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 and, as a result, omits some of the objects and properties defined in STIX 1.2.1 (see STIX™ Version 1.2.1 Part 1: Overview). 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.

1.0.1 IPR Policy

This Committee Specification Public Review Draft is provided under the Non-Assertion Mode of the OASIS IPR Policy, the mode chosen when the Technical Committee was established.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the TC's web page (https://www.oasis-open.org/committees/cti/ipr.php).

1.1 Terminology

The key words "**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].

All text is normative except for examples, the overview (section 1.4), and any text marked non-normative.

1.2 Normative References

[IEEE 754-2008] "IEEE Standard for Floating-Point Arithmetic", IEEE 754-2008, August 2008. [Online]. Available: http://ieeexplore.ieee.org/document/4610935/

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

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

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

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

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

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014. http://www.rfc-editor.org/info/rfc7159.txt.

[TLP] Traffic Light Protocol, Version 1.0 (TLP). (2016, Aug. 25). FIRST. [Online]. Available: https://first.org/tlp

1.3 Non-Normative References

[CAPEC] Common Attack Pattern Enumeration and Classification (CAPEC). (2014, Nov. 7). The MITRE Corporation. [Online]. Available: http://capec.mitre.org.

[Casey 2007] Casey, T., Threat Agent Library Helps Identify Information Security Risks. September 2007. [Online]. Available: https://communities.intel.com/servlet/JiveServlet/downloadBody/1151-102-1-1111/Threat Agent Library_07-2202w.pdf.

[Casey 2015] Casey, T., “Understanding Cyberthreat Motivations to Improve Defense”, Intel, February 2015. [Online]. Available: https://communities.intel.com/servlet/JiveServlet/previewBody/23856-102-1-28290/understanding-cyberthreat-motivations-to-improve-defense-paper-l.pdf.

[Goessner 2007] Goessner, S., “JSONPath - XPath for JSON”, February 2007. [Online]. Available: http://goessner.net/articles/JsonPath/.

[JSON Schema] OASIS Cyber Threat Intelligence (CTI) TC, “cti-stix2-json-schemas”, OASIS. [Online]. Available: https://github.com/oasis-open/cti-stix2-json-schemas.

[Mell 2005] Mell, P., Kent, K. and Nusbaum, J., “Guide to Malware Incident Prevention and Handling”, NIST Special Publication 800-83, November 2005. [Online]. Available: http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-83.pdf.

[VERIS] VERIS Community Database. (n.d.). [Online]. Available: http://vcdb.org/

1.4 Overview

1.4.1 Graph-Based Model

STIX 2.0 is 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.

1.4.2 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 corresponds to a concept commonly used in CTI. Using the building blocks of SDOs alongside STIX relationships, entities can create and share broad and comprehensive CTI.

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.

STIX Domain Objects are defined in *STIX™ Version 2.0. Part 2: STIX Objects*.

1.4.3 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 be used to detect 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.

Embedded relationships (ID references) are described in section 3.2 and STIX Relationship Objects (SROs) are defined in section 3 of *STIX™ Version 2.0. Part 2: STIX Objects*.

1.4.4 Cyber Observables

Some parts of the STIX language require describing structured representation of observed objects and their properties in the cyber domain. These capabilities differ from the parts of STIX used to describe higher-level concepts in many ways and are therefore contained in a separate section of the specification. The Cyber Observable sections describe one or more observed data points, for example, information about a file that existed, a process that was observed running, or that network traffic occurred between two IPs. It describes the facts concerning **what** happened, but not necessarily the who or when, and never the why.

Cyber Observables are defined by two documents in this specification. *STIX™ Version 2.0. Part 3: Cyber Observable Core Concepts* describes and defines Cyber Observable Core Concepts, which are the parts of STIX that are specific to representation of cyber observables. *STIX™ Version 2.0. Part 4: Cyber Observable Objects* contains a library of Cyber Observable Objects: definitions for the types of things that can be observed.

1.4.5 STIX™ Patterning

In order to enhance detection of possibly malicious activity on networks and endpoints, a standard language is needed to describe what to look for in a cyber environment. The STIX Patterning language allows matching against timestamped Cyber Observable data (such as STIX Observed Data Objects) collected by a threat intelligence platform or other similar system so that other analytical tools and systems can be configured to react and handle incidents that might arise. STIX Patterning is a general concept that can be used anywhere, but in STIX it is currently used by the Indicator object.

STIX Patterning is defined in *STIX™ Version 2.0. Part 5: STIX Patterning*.

1.4.6 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 entities are permitted to use values outside of the suggested vocabulary.

STIX vocabularies are defined in section 6. Properties that are defined as open vocabularies identify a suggested vocabulary from that section. For example, the Indicator **`labels`** property, as defined in section 2.5 of *STIX™ Version 2.0. Part 2: STIX Objects*, uses the Indicator Label vocabulary as defined in section 6.5.

1.4.7 Serialization

STIX is defined independent of any specific storage or serialization. However, the mandatory-to-implement (MTI) serialization for STIX 2.0 is JSON [RFC7159]. In other words, all STIX-conformant tools have to implement support for JSON and can implement support for other serializations.

JSON schemas have been developed by members of the Cyber Threat Intelligence Technical Committee and are available in the cti-stix2-json-schemas OASIS Open Repository [JSON Schema]. The JSON schemas are informative and serve as a best effort attempt to validate that STIX 2.0 content meets the

structural requirements identified in this specification. This specification is the normative description of STIX 2.0.

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

1.4.8 Transporting STIX™

STIX 2.0 is transport-agnostic, i.e., the structures and serializations do not rely on any specific transport mechanism. A companion CTI specification, TAXII™, is designed specifically to transport STIX Objects. STIX provides a Bundle (see section 5) as a container for STIX Objects to allow for transportation of bulk STIX data, especially over non-TAXII communication mechanisms.

1.5 Naming Requirements

1.5.1 Property Names and String Literals

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.

1.5.2 Reserved 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.

1.6 Document Conventions

1.6.1 Naming Conventions

All type names, property names and literals are in lowercase, except when referencing canonical names defined in another standard (e.g. literal values from an IANA registry). Words in property names are separated with an underscore (_), while words in 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.

1.6.2 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 blue with a blue background – `malicious-activity`
  - All relationship types are string literals, therefore they will also appear in blue with a blue 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-point font, with straight quotes, black text and a `light grey background`, and 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`.
`external-reference`	A non-STIX identifier or reference to other related external content.
`float`	An IEEE 754 [IEEE 754-2008] double-precision number.
`hashes`	One or more cryptographic hashes.
`identifier`	An identifier (ID) for a STIX Domain Object, STIX Relationship Object, Bundle, or Marking Definition.
`integer`	A whole number.
`kill-chain-phase`	A name of a kill chain phase.
`list`	A sequence of values ordered based on how they appear in the list. The phrasing "`list` of type `<type>`" is used to indicate that all values within the list **MUST** conform to the specified type.
`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).

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 [RFC7159], which is a literal (unquoted) `true` or `false`.

Examples

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

2.2 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 [RFC7159] when representing `external-reference`.

2.2.1 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 [RFC3986].
`hashes` (optional)	`hashes`	Specifies a dictionary of hashes for the contents of the **`url`**. This **SHOULD** be provided when the **url** property is present.
`external_id` (optional)	`string`	An identifier for the external reference content.

2.2.2 Requirements

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

Examples

An `external-reference` to a VERIS Community Database (VCDB) [VERIS] 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",
      "hashes": {
        "SHA-256": "6db12788c37247f2316052e142f42f4b259d6561751e5f401a1ae2a6df9c674b"
      }
    }
  ],
  ...
}
```

An `external-reference` from the CAPEC™ [CAPEC] 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 an offline threat report (i.e., e-mailed, offline, etc.)

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

2.3 Float

Type Name: `float`

The float data type represents an IEEE 754 [IEEE 754-2008] double-precision number (e.g., a number with a fractional part). However, because the values ±Infinity and NaN are not representable in JSON, they are not valid values in STIX.

In the JSON MTI serialization, floating point values are represented by the JSON number type [RFC7159].

Examples

```
{
  ...
  "distance": 8.321,
  ...
}
```

2.4 Hashes

Type Name: `hashes`

The Hashes type represents 1 or more cryptographic hashes, as a special set of key/value pairs. Accordingly, the name of each hashing algorithm **MUST** be specified as a key in the dictionary and **MUST** identify the name of the hashing algorithm used to generate the corresponding value. This name **SHOULD** either be one of the values defined in the `hash-algorithm-ov` OR a custom value prepended with "x_" (e.g., "x_custom_hash").

Keys **MUST** be unique in each `hashes` property, **MUST** be in ASCII, and are limited to the characters a-z (lowercase ASCII), A-Z (uppercase ASCII), numerals 0-9, hyphen (-), and underscore (_). Keys **SHOULD** be no longer than 30 ASCII characters in length, **MUST** have a minimum length of 3 ASCII characters, **MUST** be no longer than 256 ASCII characters in length.

Examples

SHA-256 and Custom Hash

```
{
  "SHA-256": "6db12788c37247f2316052e142f42f4b259d6561751e5f401a1ae2a6df9c674b",
  "x_foo_hash": "aaaabbbbccccddddeeeeffff0123457890"
}
```

2.5 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 (all type names are lowercase strings, by definition) from the `type` property 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) [RFC4122].

The JSON MTI serialization uses the JSON string type [RFC7159] when representing `identifier`.

Examples

```
{
```

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

{
  ...
  "type": "threat-actor",
  "id": "threat-actor--5ee9db36-4a1e-4dd4-bb32-2551eda97f4a",
  ...
}
```

2.6 Integer

Type Name: `integer`

The integer data type represents a whole number. Unless otherwise specified, all integers **MUST** be capable of being represented as a signed 64-bit value ([-(2**53)+1, (2**53)-1]). Additional restrictions **MAY** be placed on the type as described where it is used.

In the JSON MTI serialization, integers are represented by the JSON number type [RFC7159].

Examples

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

2.7 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 [RFC7159] 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`**.**

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"
    }
  ],
  ...
}
```

2.8 List

Type Name: `list`

The `list` type defines a sequence of values ordered based on how they appear in the list. The phrasing "`list` of type `<type>`" is used to indicate that all values within the list **MUST** conform to the specified type. For instance, `list` of type `integer` means that all values of the list must be of the `integer` type. This specification does not specify the maximum number of allowed values in a `list`, however every instance of a `list` **MUST** have at least one value. Specific 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. If the property is required, the list **MUST** be present and **MUST** have at least one value.

The JSON MTI serialization uses the JSON array type [RFC7159], which is an ordered list of zero or more values.

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"
  ],
  ...
}
```

2.9 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, that is identified in the definition for that property. The suggested vocabularies are defined in section 6. 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 [RFC7159] when representing `open-vocab`.

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"],
  ...
}
```

2.10 String

Type Name: `string`

The `string` data type represents a finite-length string of valid characters from the Unicode coded character set [ISO10646]. Unicode incorporates ASCII and the characters of many other international character sets.

The JSON MTI serialization uses the JSON string type [RFC7159], which mandates the UTF-8 encoding for supporting Unicode.

Examples

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

2.11 Timestamp

Type Name: `timestamp`

The `timestamp` type defines how dates and times are represented in STIX.

The JSON MTI serialization uses the JSON string type [RFC7159] when representing `timestamp`.

2.11.1 Requirements

- The `timestamp` property **MUST** be a valid RFC 3339-formatted timestamp [RFC3339] using the format `YYYY-MM-DDTHH:mm:ss[.s+]Z` where the "s+" represents 1 or more sub-second values. The brackets denote that sub-second precision is optional, and that if no digits are provided, the decimal place **MUST NOT** be present.
- The timestamp **MUST** be represented in the UTC timezone and **MUST** use the "Z" designation to indicate this.

Examples

```
{
  ...
```

```
  "created": "2016-01-20T12:31:12.123Z",
  ...
}
```

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 [RFC7159] 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** property **MUST** be the name of one of the types of STIX Object defined in sections 2 and 3 of *STIX™ Version 2.0. Part 2: STIX Objects* (e.g., `indicator`) or the name of a custom object as defined by section 7.2.
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. Because the object type is part of the `identifier`, it is invalid 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. 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. The **created** property **MUST NOT** be changed when creating a new version of the object.

		The **created** timestamp **MUST** be precise to the nearest millisecond (exactly three digits after the decimal place in seconds). See section 3.4 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. Object creators **MUST** set the **modified** property when creating a new version of an object. The **modified** timestamp **MUST** be precise to the nearest millisecond (exactly three digits after the decimal place in seconds). See section 3.4 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. The default value of this property is `false`. See section 3.4 for further definition of versioning.
labels (optional)	`list` of type `string`	The **labels** property specifies a set of classifications. Each STIX Object can define a suggested vocabulary for the **labels** property. For example, the Indicator object, as defined in section 2.5 of *STIX™ Version 2.0. Part 2: STIX Objects*, uses the Indicator Label vocabulary as defined in section 6.5.

		If a vocabulary is defined, items in this list **SHOULD** come from the 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 section 4 for further definition of data markings.
granular_markings (optional)	`list` of type `granular-marking`	The **granular_markings** property specifies a list of granular markings applied to this object. See section 4 for further definition of data markings.

3.2 IDs and References

The **id** property universally and uniquely identifies an SDO, SRO, Bundle, or Marking Definition. It **MUST** meet the requirements of the `identifier` type (see section 2.5).

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 properties 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 property. 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 section 3.4. ID references can 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 **MAY** be captured in the **created_by_ref** property (or that property can be left blank, meaning the object creator is anonymous).

Entities that re-publish an object from another entity without making any changes to the object, and thus maintaining the original **id**, are not considered the object creator and **MUST 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

Versioning is the mechanism that object creators use to update and revoke the STIX Objects that they create. This section describes the versioning process and normative rules for performing versioning and revocation. STIX Objects are versioned using the **revoked**, **created**, and **modified** properties. See the properties table in section 3.1 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 the combination of its **id** and **modified** properties. The first version of the object **MUST** have the same timestamp for the **created** and **modified** properties. More recent values of the **modified** property indicate later versions of the object. Implementations **MUST** consider the version of the STIX Object with the most recent **modified** value to be the most recent state of the object. For every new version of an object, the **modified** property **MUST** be updated to represent the time that the new version was created. If a consumer receives two objects that are different, but have the same **id** and **modified** timestamp, it is not defined how the consumer handles the objects. 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. The object creator may (but not necessarily will) be identified in the **created_by_ref** property of the object. 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 **modified** properties) **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 **modified** property **MUST** be updated with the time of the change to indicate a new version.

Objects can also be revoked, which means that they are no longer considered valid by the object creator. As with issuing a new version, only the object creator is permitted to revoke a 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. Changing the **revoked** property to indicate that an object is revoked is an update to the object, and therefore its **modified** property **MUST** be updated at the same time. This specification does not address how implementations should handle revoked data.

3.4.1 Versioning Timestamps

There are two timestamp properties used to indicate when 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 creation and modification times for use in the **created** and **modified** properties (e.g. one system might use when the object is first added to

the local database as the creation time, while another might use the time when the object is first distributed as STIX).

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.

Examples

Example of a new version

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

Note: the IDs in the example below use a simplified format to help illustrate the changing IDs more clearly.

Step #	STIX Object	Object Creator Action

1	<pre>{
	  "type": "example",
	  "id": "example--1",
	  "created": "2016-05-01T06:13:14.000Z",
	  "modified": "2016-05-01T06:13:14.000Z",
	  "name": "attention",
	  "description": "this is the description"
	}</pre>
		Original version of an object is created.

2	N/A, STIX is not involved in this step	Object creator changes the name in their internal database.

3	<pre>{
	  "type": "example",
	  "id": "example--1",
	  "created": "2016-05-01T06:13:14.000Z",
	  "modified": "2016-05-08T03:43:44.000Z",
	  "name": "Attention!",</pre>
		Object creator updates the **`modified`** property.

	<pre>"description": "this is the description"
	}</pre>	

Example of derived object

One object creator has decided that the previous name 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	<pre>{
	"type": "example",
	"id": "example--1",
	"created": "2016-05-01T06:13:14.000Z",
	"modified": "2016-05-01T06:13:14.000Z",
	"name": "attention",
	"description": "this is the description"
	}</pre>
		Original object created (via new id and setting **`created`** and **`modified`** to the same value).

2	N/A, STIX is not involved in this step	Object creator changes the name in their internal database.

3	<pre>{
	"type": "example",
	"id": "example--1",
	"created": "2016-05-01T06:13:14.000Z",
	"modified": "2016-05-08T03:43:44.000Z",
	"name": "attention",
	"description": "this is the description",
	"revoked": true
	}</pre>
		Object creator revokes the existing object by setting **`revoked`** to `true`. The **`modified`** property is updated.

4	<pre>{
	"type": "example",
	"id": "example--2",
	"created": "2016-05-08T03:43:44.000Z",
	"modified": "2016-05-08T03:43:44.000Z",
	"name": "Something completely different",
	"description": "this is the description"
	}</pre>
		Object creator creates a new object (with a new **`id`** and setting **`created`** and **`modified`** to the same value).

5	<pre>{
	"type": "relationship",</pre>
		(Optional) Object creator creates a new Relationship indicating that

	<pre>{
	  "id": "relationship--3",
	  "created": "2016-05-08T03:43:44.000Z",
	  "modified": "2016-05-08T03:43:44.000Z",
	  "relationship_type": "derived-from",
	  "source_ref": "example--1",
	  "target_ref": "example--2"
	}</pre>
		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	<pre>{
	  "type": "example",
	  "id": "example--1",
	  "created": "2016-05-01T06:13:14.000Z",
	  "modified": "2016-05-01T06:13:14.000Z"
	}</pre>
		Consumer stores example object because this is the first time they have seen the object.

2	<pre>{
	  "type": "example",
	  "id": "example--1",
	  "created": "2016-05-01T06:13:14.000Z",
	  "modified": "2016-05-08T03:43:44.000Z"
	}</pre>
		Consumer updates example object because the received **`modified`** property is later than the object that is currently stored.

3	<pre>{
	  "type": "example",
	  "id": "example--1",
	  "created": "2016-05-01T06:13:14.000Z",
	  "modified": "2016-05-06T06:23:45.000Z"
	}</pre>
		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	<pre>{
	  "type": "example",
	  "id": "example--1",
	  "created": "2016-05-01T06:13:14.000Z",
	  "modified": "2016-05-11T06:41:21.000Z",</pre>
		Consumer decides to delete example object, but keeps some metadata regarding the object.

	<pre>"revoked": true
	}</pre>	

5	<pre>{
	"type": "example",
	"id": "example--1",
	"created": "2016-05-01T06:13:14.000Z",
	"modified": "2016-05-10T17:28:54.000Z"
	}</pre>
		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. (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	<pre>{
	"type": "example",
	"id": "example--2",
	"created": "2016-05-01T06:13:14.000Z",
	"modified": "2016-05-01T06:13:14.000Z"
	}</pre>
		The user clicks the "share" button, delivering the intelligence to sharing partners.

3	N/A – STIX is not involved in this scenario. (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	<pre>{
	"type": "example",
	"id": "example--2",
	"created": "2016-05-01T06:13:14.000Z",
	"modified": "2016-05-03T16:33:51.000Z"
	}</pre>
		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	<pre>{
	  "type": "example",
	  "id": "example--2",
	  "created": "2016-05-01T06:13:14.000Z",
	  "modified": "2016-05-08T13:35:12.000Z",
	  "revoked": true
	}</pre>
		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 relationship types that are specified in the definition of that SDO. The following common relationship types are defined for all SDOs. See section 1.4.3 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. `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 3.4.
`duplicate-of`	*<STIX Domain Object>*	*<STIX Domain Object of same type>*	The referenced source and target objects are semantically duplicates of each other. 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. 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. 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`
- `first_seen_precision`
- `last_seen_precision`
- `valid_from_precision`
- `valid_until_precision`

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.

Unlike STIX Objects, Marking Definition objects cannot be versioned because it would allow for indirect changes to the markings on a STIX Object. For example, if a Statement marking is changed from "Reuse Allowed" to "Reuse Prohibited", all STIX Objects marked with that Statement marking would effectively have an updated marking without being updated themselves. Instead, a new Statement marking with the new text should be created and the marked objects updated to point to the new marking.

The JSON MTI serialization uses the JSON object type [RFC7159] when representing `marking-definition`.

4.1.1 Properties

Property Name	Type	Description
`type` (required)	`string`	The **`type`** property identifies the type of object. The value of this property **MUST** be `marking-definition`.
`id` (required)	`identifier`	The **`id`** property universally and uniquely identifies this Marking Definition. 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. 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 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 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-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). 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). 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 (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 4.1.4, the Statement marking as defined in section 4.1.3, or some other marking definition defined elsewhere).

4.1.2 Relationships

Data Marking is not a STIX Object and MUST NOT have any SRO relationships to it or from it. This table lists the embedded relationships by property name along with their corresponding target.

Embedded Relationships	
created_by_ref	identity
object_marking_refs	marking-definition

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.

Examples

```
{
  "type": "marking-definition",
  "id": "marking-definition--34098fce-860f-48ae-8e50-ebd3cc5e41da",
  "created": "2016-08-01T00:00:00.000Z",
  "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 property. 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 [TLP] 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`	`{`
	`  "type": "marking-definition",`
	`  "id": "marking-definition--613f2e26-407d-48c7-9eca-b8e91df99dc9",`
	`  "created": "2017-01-20T00:00:00.000Z",`
	`  "definition_type": "tlp",`
	`  "definition": {`
	`  "tlp": "white"`
	`  }`
	`}`

`green`	`{`
	`  "type": "marking-definition",`
	`  "id": "marking-definition--34098fce-860f-48ae-8e50-ebd3cc5e41da",`
	`  "created": "2017-01-20T00:00:00.000Z",`
	`  "definition_type": "tlp",`
	`  "definition": {`
	`  "tlp": "green"`

	<pre> } }</pre>

amber	<pre>{
	  "type": "marking-definition",
	  "id": "marking-definition--f88d31f6-486f-44da-b317-01333bde0b82",
	  "created": "2017-01-20T00:00:00.000Z",
	  "definition_type": "tlp",
	  "definition": {
	    "tlp": "amber"
	  }
	}</pre>

red	<pre>{
	  "type": "marking-definition",
	  "id": "marking-definition--5e57c739-391a-4eb3-b6be-7d15ca92d5ed",
	  "created": "2017-01-20T00:00:00.000Z",
	  "definition_type": "tlp",
	  "definition": {
	    "tlp": "red"
	  }
	}</pre>

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.

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 section 4.3.1.1. The marking-definition referenced in the **marking_ref** property 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** property 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": "2016-02-14T00:00:00.000Z",
  "modified": "2016-02-14T00:00:00.000Z",
```

```
  "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 [Goessner 2007] 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.

Examples

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",
  "name": "Some name",
  "labels": ["first", "second"]
}
```

5 Bundle

Type Name: `bundle`

A Bundle is a collection of arbitrary STIX Objects and Marking Definitions grouped together in a single container. A Bundle does not have any semantic meaning and Objects are not 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`** properties. 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 [RFC7159] when representing `bundle`.

5.1 Properties

Property Name	Type	Description
`type` (required)	`string`	The **`type`** property identifies the type of object. The value of this property **MUST** be `bundle`.
`id` (required)	`identifier`	An identifier for this Bundle. The **`id`** property 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 or the ability to refer to bundles by ID.
`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. The value of this property **MUST** be `2.0` for bundles containing STIX Objects defined in this specification.
`objects` (optional)	`list` of type *`<STIX Object>`* or `marking-definition`	Specifies a set of one or more STIX Objects. Objects in this list **MUST** be a STIX Object (SDO, SRO or Custom Object) or a Marking Definition object.

5.2 Relationships

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

Examples

```
{
  "type": "bundle",
  "id": "bundle--5d0092c5-5f74-4287-9642-33f4c354e56d",
  "spec_version": "2.0",
  "objects": [
    {
      "type": "indicator",
      "id": "indicator--8e2e2d2b-17d4-4cbf-938f-98ee46b3cd3f",
      "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
      "created": "2016-04-29T14:09:00.000Z",
      "modified": "2016-04-29T14:09:00.000Z",
      "object_marking_refs": ["marking-definition--089a6ecb-cc15-43cc-9494-767639779123"],
      "name": "Poison Ivy Malware",
      "description": "This file is part of Poison Ivy",
      "pattern": "[file:hashes.'SHA-256' =
'aec070645fe53ee3b3763059376134f058cc337247c978add178b6ccdfb0019f']"
    },
    {
      "type": "marking-definition",
      "id": "marking-definition--34098fce-860f-48ae-8e50-ebd3cc5e41da",
      "created": "2016-08-01T00:00:00.000Z",
      "definition_type": "tlp",
      "definition": {
        "tlp": "green"
      }
    }
  ]
}
```

6 Vocabularies

The following sections provide object-specific listings for each of the vocabularies referenced in the object description sections defined in *STIX™ Version 2.0. Part 2: STIX Objects*. 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.

6.1 Attack Motivation

Vocabulary 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.

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

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. 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.

	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.
`dominance`	A desire to assert superiority over someone or something else. 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. Adversaries who act for ideological reasons (e.g., political, religious, human rights, environmental, desire to cause chaos/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. 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. 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. 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. 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.

	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. 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. 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.

6.2 Attack Resource Level

Vocabulary 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 [Casey 2007].

Vocabulary Summary	
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.

6.3 Hashing Algorithm Vocabulary

Vocabulary Name: hash-algorithm-ov

An open vocabulary of hashing algorithms.

When specifying a hashing algorithm not already defined within the hash-algorithm-ov, wherever an authoritative name for a hashing algorithm name is defined, it should be used as the value. In cases where no authoritative name exists and/or where there is variance in the naming of a particular hashing algorithm, producers should exercise their best judgement.

Vocabulary Summary

MD5, MD6, RIPEMD-160, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512, SSDEEP, WHIRLPOOL	
Vocabulary Value	**Description**
MD5	Specifies the MD5 message digest algorithm. The corresponding hash string for this value **MUST** be a valid MD5 message digest as defined in [RFC 1321].
MD6	Specifies the MD6 message digest algorithm. The corresponding hash string for this value **MUST** be a valid MD6 message digest as defined in the [MD6] proposal.
RIPEMD-160	Specifies the RIPEMD--160 (RACE Integrity Primitives Evaluation Message Digest) cryptographic hash function. The corresponding hash string for this value **MUST** be a valid RIPEMD-160 message digest as defined in the [RIPEMD-160] specification.
SHA-1	Specifies the SHA--1 (secure--hash algorithm 1) cryptographic hash function. The corresponding hash string for this value **MUST** be a valid SHA-1 message digest as defined in [RFC 3174].
SHA-224	Specifies the SHA--224 cryptographic hash function (part of the SHA-2 family). The corresponding hash string for this value **MUST** be a valid SHA-224 message digest as defined in [RFC 6234].
SHA-256	Specifies the SHA--256 cryptographic hash function (part of the SHA-2 family). The corresponding hash string for this value **MUST** be a valid SHA-256 message digest as defined in [RFC 6234].
SHA-384	Specifies the SHA--384 cryptographic hash function (part of the SHA-2 family). The corresponding hash string for this value **MUST** be a valid SHA-384 message digest as defined in [RFC 6234].
SHA-512	Specifies the SHA--512 cryptographic hash function (part of the SHA-2 family). The corresponding hash string for this value **MUST** be a valid SHA-512 message digest as defined in [RFC 6234].
SHA3-224	Specifies the SHA3-224 cryptographic hash function. The corresponding hash string for this value **MUST** be a valid SHA3-224 message digest as defined in [FIPS202].
SHA3-256	Specifies the SHA3-256 cryptographic hash function. The corresponding hash string for this value **MUST** be a valid SHA3-256 message digest as defined in [FIPS202].

SHA3-384	Specifies the SHA3-384 cryptographic hash function. The corresponding hash string for this value **MUST** be a valid SHA3-384 message digest as defined in [FIPS202].
SHA3-512	Specifies the SHA3-512 cryptographic hash function. The corresponding hash string for this value **MUST** be a valid SHA3-512 message digest as defined in [FIPS202].
ssdeep	Specifies the ssdeep fuzzy hashing algorithm. The corresponding hash string for this value **MUST** be a valid piecewise hash as defined in the [SSDEEP] specification.
WHIRLPOOL	Specifies the whirlpool cryptographic hash function. The corresponding hash string for this value **MUST** be a valid WHIRLPOOL message digest as defined in [ISO10118].

6.4 Identity Class

Vocabulary 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.

6.5 Indicator Label

Vocabulary 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.

6.6 Industry Sector

Vocabulary 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`	
`defence`	
`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	

6.7 Malware Label

Vocabulary 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-security-software, 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 [Mell2005].
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.

6.8 Report Label

Vocabulary 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`, `identity`, `indicator`, `malware`, `observed-data`, `threat-actor`, `tool`, `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.
`identity`	Report subject is a characterization of one or more identities 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.
`vulnerability`	Report subject is a characterization of one or more vulnerabilities and related information.

6.9 Threat Actor Label

Vocabulary 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. [Casey 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 attempts to disrupt an organization's business model or damage their image. This category includes actors sometimes referred to as anarchists, cyber vandals, extremists, and hacktivists.
`competitor`	An organization that competes in the same economic marketplace. 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. 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. 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. 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. “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. “Insider” in this context includes any person extended internal trust, such as regular employees, contractors, consultants, and temporary workers. 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. 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. 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. 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. 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.

6.10 Threat Actor Role

Vocabulary 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.
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.

6.11 Threat Actor Sophistication

Vocabulary Name: threat-actor-sophistication-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. Example Roles: Average User These actors: • 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. 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. Example Roles: Script-Kiddie These actors: ● attack known weaknesses; ● use well known scripts and tools; and ● 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. 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. Example Roles: Toolkit User These actors: ● attack known vulnerabilities; ● use attack frameworks and toolkits; and ● 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. 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. Example Roles: Toolkit Developer These actors: ● attack known vulnerabilities; ● can create their own tools; and

	• 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. Example Roles: Vulnerability Researcher, Reverse Engineer, Threat Researcher, Malware Creator These actors: • attack unknown and known vulnerabilities; • can create their own tools from scratch; and • have proficient knowledge of the tools.
innovator	Typically criminal or state actors who are organized, highly technical, proficient, well-funded professionals working in teams to discover new vulnerabilities and develop exploits. 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. Example Roles: Toolkit Innovator, 0-Day Exploit Author These actors: • attack unknown and known vulnerabilities; • create attacks against 0-Day exploits from scratch; and • create 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. These actors: • can create or use entire supply chains to launch an attack; • can create and design attacks for any systems, software package, or device; and • are responsible for APT-level attacks.

6.12 Tool Label

Vocabulary 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.

7 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.

7.1 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.

7.1.1 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.

Examples

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

7.2 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.

7.2.1 Requirements

- Producers **MAY** include any number of Custom Objects in STIX documents.
- Custom Objects **MUST** support the Common Properties as defined in section 3.1.
    - The definitions of these properties are the same as those defined in Common Properties and therefore those properties **MUST NOT** be used to represent the custom properties in the object.
- The **`type`** property in a Custom Object **MUST** be in ASCII and **MUST** only contain the characters a–z (lowercase ASCII), 0–9, and hyphen (-).
- The **`type`** property **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`** property 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, `[object-type]--[UUIDv4]`.
- Custom Objects **SHOULD** only be used when there is no existing STIX Object defined by the STIX specification that fulfils that need.

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": "2016-08-01T00:00:00.000Z",
      "modified": "2016-08-01T00:00:00.000Z",
      "some_custom_stuff": 14,
      "other_custom_stuff": "hello"
    }
  ]
}
```

8 Conformance

8.1 Producers and Consumers

A "STIX 2.0 Producer" is any software that creates STIX 2.0 content and conforms to the following normative requirements:

1. It **MUST** be able to create content encoded as JSON.
2. All properties marked required in the property table for the STIX Object or type **MUST** be present in the created content.
3. All properties **MUST** conform to the data type and normative requirements for that property.
4. It **MUST** support at least one STIX Object per the Conformance section in *STIX™ Version 2.0. Part 2: STIX Objects*.
5. It **MUST** support all features listed in section 8.2, Mandatory Features.
6. It **MAY** support any features listed in section 8.3, Optional Features. Software supporting an optional feature **MUST** comply with the normative requirements of that feature.
7. It **MUST** support JSON as a serialization format and **MAY** support serializations other than JSON.

A "STIX 2.0 Consumer" is any software that consumes STIX 2.0 content and conforms to the following normative requirements:

1. It **MUST** support parsing all required properties for the content that it consumes.
2. It **MUST** support all features listed in section 8.2, Mandatory Features.
3. It **MAY** support any features listed in section 8.3, Optional Features. Software supporting an optional feature **MUST** comply with the normative requirements of that feature.
4. It **MUST** support JSON as a serialization format and **MAY** support serializations other than JSON.

8.2 Mandatory Features

8.2.1 Versioning

A STIX 2.0 Producer or STIX 2.0 Consumer **MUST** support versioning by following the normative requirements listed in section 3.4.

8.3 Optional Features

8.3.1 Object-Level Data Markings

A STIX 2.0 Producer or STIX 2.0 Consumer **MAY** support "Object-Level Data Markings". Software claiming to support "Object-Level Data Markings" **MUST** follow the normative requirements listed in sections 4.1 and 4.2.

8.3.2 Granular Data Markings

A STIX 2.0 Producer or STIX 2.0 Consumer **MAY** support "Granular Data Markings". Software claiming to support "Granular Data Markings" **MUST** follow the normative requirements listed in sections 4.1 and 4.3.

8.3.3 Custom Properties

A STIX 2.0 Producer or STIX 2.0 Consumer **MAY** support "Custom Properties". Software claiming to support "Custom Properties" **MUST** follow the normative requirements listed in section 7.1.

8.3.4 Custom Objects

A STIX 2.0 Producer or STIX 2.0 Consumer **MAY** support "Custom Objects". Software claiming to support "Custom Objects" **MUST** follow the normative requirements listed in section 7.2.

Appendix A. Glossary

CAPEC - Common Attack Pattern Enumeration and Classification

Consumer - Any entity that receives STIX content

CTI - Cyber Threat Intelligence

Embedded Relationship - A link (an "edge" in a graph) between one STIX Object and another represented as a property on one object containing the ID of another object

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 or updated a STIX object (see section 3.3)

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 (a "node" in a graph)

SRO - STIX Relationship Object (one mechanism to represent an "edge" in a graph)

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)

STIX Relationship - A link (an "edge" in a graph) between two STIX Objects represented by either an SRO or an embedded relationship

TAXII - An application layer protocol for the communication of cyber threat information

TLP - Traffic Light Protocol

TTP - Tactic, technique, or procedure; behaviors and resources that attackers use to carry out their attacks

Appendix B. Acknowledgments

STIX Subcommittee Chairs:

Sarah Kelley, Center for Internet Security (CIS)
John Wunder, MITRE Corporation

Cyber Observable Subcommittee Chairs:

Trey Darley, Kingfisher Operations, sprl
Ivan Kirillov, MITRE Corporation

Special Thanks:

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

Sarah Kelley, Center for Internet Security (CIS)
Terry MacDonald, Cosive
Jane Ginn, Cyber Threat Intelligence Network, Inc. (CTIN)
Richard Struse, DHS Office of Cybersecurity and Communications
Iain Brown, GDS
Jason Keirstead, IBM
Tim Casey, Intel
Trey Darley, Kingfisher Operations, sprl
Allan Thomson, LookingGlass Cyber
Greg Back, MITRE Corporation
Ivan Kirillov, MITRE Corporation
Jon Baker, MITRE Corporation
John Wunder, MITRE Corporation
Sean Barnum, MITRE Corporation
Richard Piazza, MITRE Corporation
Christian Hunt, New Context Services, Inc.
John-Mark Gurney, New Context Services, Inc.
Aharon Chernin, Perch
Dave Cridland, Surevine
Bret Jordan, Symantec Corp.

Participants:

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
Russell Spitler, AlienVault

Ryan Clough, Anomali
Nicholas Hayden, Anomali
Wei Huang, Anomali
Angela Nichols, Anomali
Hugh Njemanze, Anomali
Katie Pelusi, Anomali
Dean Thompson, Australia and New Zealand Banking Group (ANZ Bank)
Alexander Foley, Bank of America
Sounil Yu, Bank of America
Vicky Laurens, Bank of Montreal
Humphrey Christian, Bay Dynamics
Ryan Stolte, Bay Dynamics
Alexandre Dulaunoy, CIRCL
Andras Iklody, CIRCL
Rapha'l Vinot, CIRCL
Sarah Kelley, CIS
Syam Appala, Cisco Systems
Ted Bedwell, Cisco Systems
David McGrew, Cisco Systems
Mark-David McLaughlin, Cisco Systems
Pavan Reddy, Cisco Systems
Omar Santos, Cisco Systems
Jyoti Verma, Cisco 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)
Jeff Odom, Dell
Sreejith Padmajadevi, Dell
Ravi Sharda, Dell
Will Urbanski, 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)
Jens Aabol, Difi-Agency for Public Management and eGovernment
Wouter Bolsterlee, EclecticIQ
Marko Dragoljevic, EclecticIQ
Oliver Gheorghe, EclecticIQ
Joep Gommers, EclecticIQ
Sergey Polzunov, EclecticIQ
Rutger Prins, EclecticIQ
Andrei S"rghi, EclecticIQ
Raymon van der Velde, EclecticIQ
Ben Sooter, Electric Power Research Institute (EPRI)
Chris Ricard, Financial Services Information Sharing and Analysis Center (FS-ISAC)
Phillip Boles, FireEye, Inc.
Prasad Gaikwad, FireEye, Inc.
Rajeev Jha, FireEye, Inc.
Anuj Kumar, FireEye, Inc.
Shyamal Pandya, FireEye, Inc.
Paul Patrick, FireEye, Inc.

Scott Shreve, FireEye, Inc.
Jon Warren, FireEye, Inc.
Remko Weterings, FireEye, Inc.
Gavin Chow, Fortinet Inc.
Steve Fossen, Fortinet Inc.
Kenichi Terashita, Fortinet Inc.
Ryusuke Masuoka, Fujitsu Limited
Daisuke Murabayashi, Fujitsu Limited
Derek Northrope, Fujitsu Limited
Jonathan Algar, GDS
Iain Brown, GDS
Adam Cooper, GDS
Mike McLellan, GDS
Tyrone Nembhard, GDS
Chris O'Brien, GDS
James Penman, GDS
Howard Staple, GDS
Chris Taylor, GDS
Laurie Thomson, GDS
Alastair Treharne, GDS
Julian White, GDS
Bethany Yates, GDS
Robert van Engelen, Genivia
Eric Burger, Georgetown University
Allison Miller, Google Inc.
Mark Risher, Google Inc.
Yoshihide Kawada, Hitachi, Ltd.
Jun Nakanishi, 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
Allen Hadden, 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
Stefan Hagen, Individual
Elysa Jones, Individual
Sanjiv Kalkar, Individual
Terry MacDonald, 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
Nathan Reller, Johns Hopkins University Applied Physics Laboratory
Pamela Smith, Johns Hopkins University Applied Physics Laboratory
David Laurance, JPMorgan Chase Bank, N.A.
Russell Culpepper, Kaiser Permanente
Beth Pumo, Kaiser Permanente
Michael Slavick, Kaiser Permanente
Trey Darley, Kingfisher Operations, sprl
Gus Creedon, Logistics Management Institute
Wesley Brown, LookingGlass
Jamison Day, LookingGlass
Kinshuk Pahare, LookingGlass
Allan Thomson, LookingGlass
Ian Truslove, LookingGlass
Chris Wood, LookingGlass
Greg Back, Mitre Corporation
Jonathan Baker, Mitre Corporation
Sean Barnum, Mitre Corporation
Desiree Beck, Mitre Corporation
Michael Chisholm, Mitre Corporation
Nicole Gong, Mitre Corporation
Ivan Kirillov, Mitre Corporation
Michael Kouremetis, Mitre Corporation
Chris Lenk, Mitre Corporation
Richard Piazza, Mitre Corporation
Larry Rodrigues, Mitre Corporation
Jon Salwen, Mitre Corporation
Charles Schmidt, Mitre Corporation
Alex Tweed, 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
Joe Brule, National Security Agency
Jessica Fitzgerald-McKay, National Security Agency
David Kemp, National Security Agency
Shaun McCullough, National Security Agency
John Anderson, NC4
Michael Butt, NC4
Mark Davidson, NC4
Daniel Dye, NC4
Angelo Mendonca, NC4
Michael Pepin, NC4

Natalie Suarez, NC4
Benjamin Yates, NC4
Daichi Hasumi, NEC Corporation
Takahiro Kakumaru, NEC Corporation
Lauri Korts-P_rn, NEC Corporation
John-Mark Gurney, New Context Services, Inc.
Christian Hunt, New Context Services, Inc.
Daniel Riedel, New Context Services, Inc.
Andrew Storms, New Context Services, Inc.
Stephen Banghart, NIST
David Darnell, North American Energy Standards Board
Cory Casanave, Object Management Group
Aharon Chernin, Perch
Dave Eilken, Perch
Sourabh Satish, Phantom
Josh Larkins, PhishMe Inc.
John Tolbert, Queralt Inc.
Ted Julian, Resilient Systems, Inc..
Igor Baikalov, Securonix
Joseph Brand, Semper Fortis Solutions
Duncan Sparrell, sFractal Consulting LLC
Thomas Schreck, Siemens AG
Rob Roel, Southern California Edison
Dave Cridland, Surevine Ltd.
Bret Jordan, Symantec Corp.
Curtis Kostrosky, Symantec Corp.
Juha Haaga, Synopsys
Masood Nasir, TELUS
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.
Patrick Coughlin, TruSTAR Technology
Chris Roblee, TruSTAR Technology
Mark Angel, U.S. Bank
Brian Fay, U.S. Bank
Joseph Frazier, U.S. Bank
Mark Heidrick, U.S. Bank
Mona Magathan, U.S. Bank
Yevgen Sautin, U.S. Bank
Richard Shok, U.S. Bank
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
Robert Coderre, VeriSign
Kyle Maxwell, VeriSign
Eric Osterweil, VeriSign
Patrick Maroney, Wapack Labs LLC
Anthony Rutkowski, Yanna Technologies LLC

Appendix C. Revision History

Revision	Date	Editor	Changes Made
01	2017-01-20	Bret Jordan, John Wunder, Rich Piazza, Ivan Kirillov, Trey Darley	Initial Version
02	2017-04-24	Bret Jordan, John Wunder, Rich Piazza, Ivan Kirillov, Trey Darley	Changes made from first public review