STIX TC Extension Definition Policy

This document aims to provide guidance and defines policy for creating and approving extension definitions to the STIX specification.

1 Extension Definition Proposal Levels

This document defines three levels of extension proposals, known as Proposal Level, to allow organizations or individuals to define extensions to the STIX specification.

1.1 Private

Extension definitions can be created by any user of STIX. Such extension definitions, which are not shared with the STIX community, are called "Private".

There are many reasons to create a Private extension definitions:

- A short term need
- Sharing only within a trust group
- Experiments

Extension proposals at this level do not have to follow the normative text in this document. In other words, the guidance and procedures described here need not be strictly followed when creating an Private extension definition, as long as it is compliant with section 7.3 of the specification.

Private extension definitions will not be available in the Common Object Repository (see section 3 below).

1.2 Open

Extension definitions that will be available more globally are called "Open". The criteria for inclusion in the Common Object Repository (see section 3) is under discussion, but they will be minimally vetted by the repository maintainer. The expectation is that the extension definition is in a usable form, and is not a "work in progress". They SHOULD be as complete as possible, as per section 2.

Extension Definitions at this proposal level can be elevated to the Specification Candidate proposal level with approval of the TC (see next section).

1.3 Specification Candidate

Extension definition proposals at this level have the full support of the TC. Approval of a proposal is based on the expectation that the extension definition will become an official part of
a future version of the STIX specification. To manage different extension implementations of the same concepts, the TC SHOULD approve commencement of work on these proposals, or of elevating an Open extension definition to such a proposal. Depending on the proposal, a mini-working group could be formed to discuss and create or update the contents of the extension. Work on the extension definition must follow the specification candidate lifecycle workflow as documented in section 4. The advancement in the life cycle will include specific criteria under the control of the TC. At a certain point in the life cycle the extension definition MUST be complete, as per section 2.

1.4 Proposal Maturity

The three proposal levels should not be thought of as representing the level of maturity of the proposals. A Private extension might be fully mature and complete (but not shared), whereas a specification candidate extension might just be an idea for an extension that the TC is supporting.

2 Contents of an Extension Definition

Except where noted, the following MUST be followed for a Specification Candidate extension proposals, SHOULD be followed by an Open extension proposal, and MAY be followed for Private extension definitions.

2.1 JSON Schema

A JSON schema that can be used to validate any content that uses the extension definition MUST exist and be accessible. It must adhere to the following:

- It MUST use the property ‘additional_properties’ for all object definitions, and the value MUST be False. Deprecated custom properties are not allowed. Extensions to Extensions are not permitted by the specification.
- To adhere to the requirement for Lists as stated in the STIX specification, all array definitions MUST contain the property ‘minItems’ which MUST be set to 1.

Any JSON schemas developed by an external specification committee (i.e., an existing standard) which are then referenced by "$ref" properties in the JSON schema for the extension definition do not have to adhere to the previous normative statements.

2.2 Extension Definition Object

An extension definition object MUST be created. It should conform to the specification in section 7.3 of the specification. Additional constraints are as follows:
- The schema property **MUST** be a URL that links directly to the location of the JSON schema that defines the syntax and semantics of the extension definition.
- The external references property **MAY** contain a URL that links to the location of the human-readable documentation (see section 2.3) for the extension definition.
- The created_by_ref property **SHOULD** point to an identity object that is easily discoverable and include contact information about the individual or group that is responsible for the definition.

The object itself **SHOULD** be stored in the Common Object Repository (see section 3), unless it is a Private extension. Alternative repositories **MAY** be created.

### 2.3 Documentation

Documentation, using the style of the STIX 2.1 OASIS specification (see section 1.1 of the specification) **MUST** be created. The document can be created using MS Word, Google Docs, Markdown, or similar products. It **MUST** be available at the URL in the external references of the extension definition, if it is provided.

### 2.4 Examples

Examples **MUST** be provided. They **SHOULD** cover some of the common use cases for the extension. They **MUST** be validated by the JSON schema. They **MUST** be available at the URL in the external references of the extension definition, if it is provided.

### 3 Common Object Repository

The TC will support the maintenance of the Common Object Repository. The current candidate for this repository is [https://github.com/oasis-open/cti-stix-common-objects](https://github.com/oasis-open/cti-stix-common-objects). Inclusion in this repository will depend upon the level of the proposal as discussed in section 1.

#### 3.1 Maintaining The Repository

The TC will assign individuals that will maintain the repository. Policies for maintaining the repository will be set by the TC.

#### 3.2 Inclusion of a Proposal

The inclusion of an extension definition into the repository will be based on the Git workflow. Individuals or groups that are interested in making a proposal should start by creating a fork of the repository. For an Open proposal, once the definition is fairly complete, a pull request should be created so the repository maintainers can determine if it should be included in the main branch of the repository. The repository maintainer will send the TC an email asking if there are any objections to inclusion. Appeals to objections are handled by the TC.
A Specification Candidate proposal can start as an Open proposal, or simply a request to the TC to begin work on the extension. Because this extension will be supported by the TC and will probably be included in a future version of the STIX specification, an email should be sent asking if there are any objections to this change of status. A fork of the repository should be made for work to begin, if not already in existence. Because this will be a work-in-progress, from time to time it might make sense to entertain a pull request to update the main branch.

3.3 Status Within the Repository

Each extension proposal will have a status within the repository. The status enumeration is to be determined, but will include such terms as "open", "started", "complete", "archived", etc. The status will change based on the lifecycle workflow (see figure below). The status of a proposal informs users of the Extension Definition of the degree of vetting and refinement that has gone into the proposal. The status of each extension will be available to all users, preferably at the repository's home page.

3.4 Licenses, Intellectual Property, CLAs…

If an extension proposal is added to the common object repository, it must be done under the rules and directives set up by OASIS for making contributions to the OASIS Open repositories. This implies that the individual (and possibly the entity the individual is doing the work under) must sign a CLA (see https://cla-assistant.io/oasis-open/Open-Repo-admin). Contributing an extension proposal to the repository does not change the intellectual property rights of the contributor, but makes it available for use by others under the license agreement (https://github.com/oasis-open/cti-stix-common-objects/blob/main/LICENSE.md).

A contributed extension proposal can be used under any rules stated in the license, which allows for inclusion in future versions of the STIX specification. If an individual or entity does not want their extension proposal to be included in the specification, they should not contribute it to the common object repository, but make it available elsewhere. Contributing an extension proposal to the repository does not imply it will be included in a future version. The TC must approve the inclusion.

4 Specification Candidate Lifecycle

As described in section 1, the TC should create a formal process for shepherding extension definition proposals into future releases of the STIX specification. Using this process allows the TC to keep track of the various proposals, avoiding unnecessary duplication of work, and supporting the individual or group responsible for the proposal.
As stated above, not all extension definition proposals will become part of the STIX specification. Trust groups might have their own use cases for extensions that are not general enough for inclusion in the specification. Additionally, alternative extension definitions for the same STIX object type might be desirable. For instance, an extension definition for Incident has been created to support the conversion of STIX 1.x content to STIX 2.x by the STIX elevator. It is based on the STIX 1.x definition of an Incident, which is different than the extension definition for Incident that is under development under the auspices of the TC. The STIX 1.x related Incident extension definition will never be part of any future specification.

Here are some initial ground rules:

1. A member of the TC requests to work on an extension proposal. This may be a proposal that is already under development, and perhaps already available in the Common Object repository (and would be in the "open" status), or simply a request to start work on an extension concept (e.g., a new STIX object type, or additional properties to an existing one to support a particular use case). The proposal will then be in the "submitted" status. A proposal request might be submitted to the TC via the CTI email list, or be presented at the TC meetings. TC approval can be as simple as there being no objections made within a certain time frame (or on the TC call), or a formal vote. The proposal will then be in the "accepted" status.

2. The proposers MAY request a mini-working group to help develop the proposal. The mini-working group is open to anyone who is a member of the TC, without exception.

3. A leader(s) of the mini-working group SHOULD be chosen.

4. A fork of the Common Object Repository should be created for the mini-working groups for all contents of the Extension Definition. The proposal will then be in the "started" status.

5. A proposal SHOULD be developed using a markdown alternative, if possible, and kept in the fork. Suggested changes can be made via pull requests. The document should be viewable by anyone, however only TC members can contribute suggestions and changes.

6. The artifacts in the fork should adhere as close to the Extension Definition specification (as described in section 2) as possible.

7. From time to time, assuming the artifacts adhere to section 2, the current state of the fork can be merged into the main (master) branch, as long as there are no objections from the TC. This can be viewed as providing a beta version of the extension definition to the community. The proposal will then be in the "beta" status. Extension proposals in the beta status MUST adhere to the normative language in section 2.

8. Once the TC initiates development of a working document for the next version of the specification, the TC will determine which Specification Candidates should be included. Candidate proposals need not be complete to be included.

9. The editors of the specification will incorporate any approved extension definition's documentation into the working document. The proposal will be in the "approved" status.
The approved status only signifies that the proposal has been included in the working document, not that it will necessarily be part of the committee specification document. 10. At any time the proposers, with the approval of the mini-working group (if one exists), can decide to stop development of an extension definition. The proposal will then be in the "archived" status. An accepted extension definition's status can be changed to "archived" only by approval of the TC.

5 Implementation Issues

5.1 Producing and Consuming Content Specified Using an Extension Definition

Producing or consuming content expressed via an Extension Definition is optional, as noted in the Conformance section (see section 12.3.3) of the STIX 2.1 specification.

The rest of this section assumes that a producer or consumer wishes to support and validate content from at least some Extension Definitions. A producer/consumer does not need to support content from all Extension Definitions, and is free to ignore those which they have no interest in. It is also possible to process such content without validating it.

The TC supports the implementation of a STIX 2 validator, which is available in the OASIS-Open Github Repository. The following describes how this validator will process content using extension definitions. Users of STIX who do not use this validator, SHOULD implement something similar.

Being able to validate extension definition content requires the following:

1. The extension content uses the Extension Definition's id to identify which extension is being used. The object corresponding to this Extension Definition MUST be available. For Extension Definitions found in the Common Object Repository, this object MUST be accessible in the objects directory.
2. The extension definition's property schema **MUST** contain a url, which links to the location of the JSON schema for the extension definition.

The STIX 2 Validator will attempt to access the JSON schema. The validator already enables the option of providing additional JSON schemas via the command line. It is assumed that a JSON schema obtained over the internet could be handled in a similar way.

5.2 Producing and Consuming Content Specified Using an Approved Extension Definition

Once the TC has approved the inclusion of an extension definition into the specification's working document and the editors have published a new version of the working document, the content defined in the Extension Definition can now be expressed using the standard syntax.

Here is an example of the use of an extension definition to add some properties to an Indicator:

```json
{
    "confidence": 50,
    "created": "2021-12-13T22:25:58.000Z",
    "created_by_ref": "identity--6b4a79e7-6669-4177-91a9-f56d73f87f7a",
    "description": "ip watch list 1234-E-45",
    "extensions": {
        "extension-definition---d83fce45-ef58-4c6c-a3f4-1fbc32e98c6e": {
            "extension_type": "property-extension",
            "rank": 5,
            "toxicity": 8
        }
    },
    "id": "indicator--cc970c4d-83b3-48fd-a82a-f4f40a245f3f",
    "indicator_types": [
        "ip-watchlist"
    ],
    "modified": "2021-12-13T22:25:58.000Z",
    "pattern": "[ipv4-addr:value = '5.50.51.40'] AND [ipv4-addr:value = '5.50.51.39']",
    "pattern_type": "stix",
    "spec_version": "2.1",
    "type": "indicator",
    "valid_from": "2021-11-29T01:00:00.000000Z",
    "valid_until": "2021-11-29T01:00:00.000000Z"
}
```

Once the extension definition proposal is approved, the standard syntax is legal:

```json
{
    "confidence": 50,
    "created": "2021-12-13T22:25:58.000Z",
    "created_by_ref": "identity--6b4a79e7-6669-4177-91a9-f56d73f87f7a",
    "description": "ip watch list 1234-E-45",
    "id": "indicator--cc970c4d-83b3-48fd-a82a-f4f40a245f3f",
    "indicator_types": [
```
Because this extension definition might have at one time been in the "beta" status (or other statuses) there might exist content that uses the extension definition syntax. There are several ways to handle this situation:

1. Either syntax is valid, possibly forever, or just until the new specification has been released
2. Either syntax is valid, but the extension definition version is deprecated and a warning is issued by the validator. This deprecated syntax version will definitely not be valid once the new specification has been released
3. The extension definition version is no longer legal, and is marked as invalid. Producers/consumers should update to the new syntax.

The use of top-level-property-extensions for extension definitions that are specification candidates might help to make this less of an issue.

6 Guidance For Creating an Execution Definition

6.1 Open Vocabularies

STIX uses the concept of "open vocabularies" for various properties. An open vocabulary is one where there is a defined suggested list of values as part of the STIX specification, but other values are permitted to be used.

For instance, the implementation_languages property of the Malware object contains a list of the implementation languages used in the malware. Suggested values come from the implementation-language-ov open vocabulary. Because there are always new languages being defined, this list could quickly become "stale". However, because this property has values from an open vocabulary, values not listed in implementation-language-ov open vocabulary can be used and are still valid with respect to the specification.

The JSON schema rule for such properties simply enforces that the value(s) is a string. As a practice, the list of suggested values is included in the JSON schema for documentation purposes. However, reading the specification of such properties more closely, it states that
values "SHOULD" come from the open-vocabulary. The STIX validator needs to issue warnings whenever SHOULD normative text is not adhered to. In this case, it might be based on a typo of a suggested value.

Continuing with the implementation-language-ov open vocabulary example, at a certain point, newly defined languages will be commonly specified for this property, and the warnings would be undesirable and somewhat irrelevant. Including new values in an open vocabulary should be possible using an extension definition.

Because the STIX 2.1 specification defined extension types vocabulary as an enumeration, no additional type is permitted. This implies that we need to use one of the existing extension types to support the addition of a new value to an open vocabulary. For this example, property-extension was chosen.

```
{
  "id": "extension-definition--320740a0-26cd-4347-9020-a951d5d3ce29",
  "type": "extension-definition",
  "spec_version": "2.1",
  "name": "Additional values for implementation-language-ov",
  "description": "This extension adds the value 'rust' to the open vocabulary",
  "created": "2022-02-20T09:16:08.989000Z",
  "modified": "2022-02-20T09:16:08.989000Z",
  "created_by_ref": "identity--11b76a96-5d2b-45e0-8a5a-f6994f370731",
  "version": "1.1",
  "extension_types": [ "property-extension" ]
}
```

The json schema would redefine the implementation-language-ov to include "rust".

"implementation-language-ov": {
  "type": "string",
}
Here is an example of the extension definition in use:

```json
{
  "type": "malware",
  "spec_version": "2.1",
  "id": "malware--0c7b5b88-8ff7-4a4d-aa9d-feb398cd0061",
  "created": "2016-05-12T08:17:27.000Z",
  "extensions": {
    "extension-definition---320740a0-26cd-4347-9020-a951d5d3ce29": {
      "extension_type": "property-extension",
    }
  }
}
```

```
"implementation_languages": [
  "rust"
]
```

```
"modified": "2016-05-12T08:17:27.000Z",
"name": "rust ransomware",
"description": "ransomware implemented in rust",
"malware_types": ["ransomware"],
"is_family": false
```