stix-ramrod Documentation

Release 1.2.0

The MITRE Corporation

March 06, 2020

Contents

Version: 1.2.0

The **stix-ramrod** library provides scripts and APIs for updating STIX and CybOX XML content.

For more information about STIX, please visit the STIX website. For more information about CybOX, please visit the CybOX website.

CHAPTER 1

Contents

Version: 1.2.0

1.1 Installation

The installation of stix-ramrod can be accomplished through a few different workflows.

1.1.1 Recommended Installation

Use PyPI and pip:

```
$ pip install stix-ramrod
```

You might also want to consider using a virtualenv. Please refer to the pip installation instructions for details regarding the installation of pip.

1.1.2 Dependencies

The stix-ramrod package relies on some non-standard Python libraries for the processing of XML content. Revisions of stix-ramrod may depend on particular versions of dependencies to function correctly. These versions are detailed within the distutils setup.py installation script.

The following libraries are required to use stix-ramrod:

- lxml - A Pythonic binding for the C libraries **libxml2** and **libxslt**.

Each of these can be installed with `pip` or by manually downloading packages from PyPI. On Windows, you will probably have the most luck using pre-compiled binaries for `lxml`. On Ubuntu (12.04 or 14.04), you should make sure the following packages are installed before attempting to compile `lxml` from source:

- libxml2-dev
- libxslt1-dev
- zlib1g-dev

Warning: Users have encountered errors with versions of libxml2 (a dependency of lxml) prior to version 2.9.1. The default version of libxml2 provided on Ubuntu 12.04 is currently 2.7.8. Users are encouraged to upgrade libxml2 manually if they have any issues. Ubuntu 14.04 provides libxml2 version 2.9.1.

1.1.3 Manual Installation

If you are unable to use pip, you can also install python-stix with setuptools. If you don't already have setuptools installed, please install it before continuing.

1. Download and install the *dependencies* above. Although setuptools will generally install dependencies automatically, installing the dependencies manually beforehand helps distinguish errors in dependency installation from errors in stix installation. Make sure you check to ensure the versions you install are compatible with the version of stix you plan to install.
2. Download the desired version of stix-ramrod from PyPI or the GitHub releases page. The steps below assume you are using the 1.2.0 release.
3. Extract the downloaded file. This will leave you with a directory named stix-ramrod-1.2.0.

```
$ tar -zxf stix-ramrod-1.2.0.tar.gz
$ ls
stix-ramrod-1.2.0 stix-ramrod-1.2.0.tar.gz
```

OR

```
$ unzip stix-ramrod-1.2.0.zip
$ ls
stix-ramrod-1.2.0 stix-ramrod-1.2.0.zip
```

4. Run the installation script.

```
$ cd stix-ramrod-1.2.0
$ python setup.py install
```

5. Test the installation.

```
$ python
Python 2.7.8 (default, Mar 22 2014, 22:59:56)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import ramrod
>>> print ramrod.__version__
1.0a1
```

If you don't see an `ImportError`, the installation was successful.

1.1.4 Further Information

If you're new to installing Python packages, you can learn more at the Python Packaging User Guide, specifically the Installing Python Packages section.

Version: 1.2.0

1.2 Getting Started

This page gives an introduction to **stix-ramrod** and how to use it. Please note that this page is being actively worked on and feedback is welcome! If you have a suggestion or something doesn't look right, let us know: (stix@mitre.org).

Note that the GitHub repository is named `stix-ramrod`, but once installed, the library is imported using the `import ramrod` statement.

1.2.1 Installation

To install **stix-ramrod** just run `pip install stix-ramrod`. If you have any issues, please refer to the instructions found on the Installation page.

1.2.2 Scripts

These instructions tell you how to upgrade STIX or CybOX content using the scripts bundled with **stix-ramrod**.

Ramrod Update

Currently, the only script bundled with **stix-ramrod** is the `ramrod_update.py` script, which can be found on your `PATH` after installing **stix-ramrod**.

Options

Running `ramrod_update.py -h` displays the following:

```
$ ramrod_update.py -h
usage: ramrod_update.py [-h] --infile INFILE [--outfile OUTFILE]
                        [--from VERSION IN] [--to VERSION OUT]
                        [--disable-vocab-update] [--disable-remove-optionals]
                        [-f]

Ramrod Updater v1.0a1: Updates STIX and CybOX documents.

optional arguments:
  -h, --help            show this help message and exit
  --infile INFILE       Input STIX/CybOX document filename.
  --outfile OUTFILE     Output XML document filename. Prints to stdout if no
                        filename is provided.
  --from VERSION IN     The version of the input document. If not supplied,
                        RAMROD will try to determine the version of the input
                        document.
  --to VERSION OUT      Update document to this version. If no version is
                        supplied, the document will be updated to the latest
                        version.
  --disable-vocab-update
                        Controlled vocabulary strings will not be updated.
  --disable-remove-optionals
                        Do not remove empty elements and attributes which were
                        required in previous language versions but became
                        optional in later releases.
  -f, --force           Removes untranslatable fields, remaps non-unique IDs,
                        and attempts to force the update process.
```

Updating STIX And CybOX Content

The `ramrod_update.py` script can accept either STIX `STIX_Package` or CybOX `Observables` documents as input. You don't need to tell it that you're updaing STIX or CybOX content–it'll figure it out for you!

Basics To update content, just provide `--infile` and `--outfile` arguments which specify the input filename and output filename. If `--outfile` is not specified, the updated document will be printed to `stdout`.

```
$ ramrod_update.py --infile stix_doc.xml --outfile update_stix_doc.xml
```

Specifying Input And Output Versions By default, `ramrod_update.py` will inspect the input document for a version number and assume that it wants to be updated to the latest version. However, you can pass in `--from` and/or `--to` arguments to override this behavior.

To specify the output version, use the `--to` argument. The following example shows how to translate a STIX v1.0 document to STIX v1.0.1:

```
$ ramrod_update.py --infile stix_1.0_doc.xml --to 1.0.1
```

If the input document does not have a version number specified (it should!), you can use the `--from` argument to declare the version of the document. The following example shows how you might declare the input document to be version STIX v1.1:

```
$ ramrod_update.py --infile stix_unversioned_doc.xml --from 1.1
```

Handling Untranslatable Elements And Attributes Some STIX and CybOX constructs have changed a lot between revisions because of growing requirements from community members, or bugfixes where the incorrect data type was assigned to a field initially and needed to be corrected. Because of this, sometimes a lossless update isn't possible.

By default, `ramrod_update.py` will inspect the input document for untranslatable fields or ID collisions, and alert the user of their presence:

```
$ ramrod_update.py --infile samples/stix_1.0_forcible.xml
[!] Update Error: Found untranslatable fields in source document.
[!] Found the following untranslatable items:
    Line 88: {http://stix.mitre.org/TTP-1}Attack_Pattern
    Line 71: {http://stix.mitre.org/TTP-1}Malware_Instance
```

At this point, users can decide to force the update process by using the `--force` or `-f` argument. This will remove the untranslatable items from the document during the update process and attempt to render a schema-valid document in the process.

Note: STIX v1.1 and CybOX v2.1 introduced schema-enforced ID uniqueness constraints. If updating content that is older than STIX v1.1 or CybOX 2.1, non-unique IDs will halt an update process. Using `--force` will cause new, unique IDs to be generated and assigned to colliding nodes.

Version: 1.2.0

1.3 What Gets Updated?

The STIX and CybOX languages have grown over the years and seen several revisions, each introducing more capabilities for structuring and characterizing cyber threat intelligence data while also taking care of bug fixes.

The following sections detail what *actually* happens to your STIX and CybOX content during the **stix-ramrod** update process.

Note: These sections don't detail *every* change made between STIX and CybOX language revisions. Some language changes are completely transparent to end-users and don't require any updates or translations at all.

The sections below cover the transformations that **stix-ramrod** performs to render a schema-valid, updated XML document.

Version: 1.2.0

1.3.1 Updates to STIX Content

The following sections detail the changes that **stix-ramrod** makes when upgrading STIX content.

Note: The sections below do not detail the full breadth and depth of changes made to the STIX language between releases. Some updates, such as the addition of new structures or controlled vocabulary terms do not require any changes to be made to the source content during an update.

To see a complete list of changes made to STIX, check out the *Release Notes* section of a STIX Language Release page.

Contents

STIX v1.1.1 to v1.2

STIX v1.2 was a minor release of the STIX language that introduced new schemas, expanded vocabularies and introduced new capabilities for existing data types.

STIX 1.2 is **completely** backwards compatible with STIX 1.1.1, so **stix-ramrod** makes minimal changes to STIX v1.1.1 content when upgrading to STIX v1.2.

The sections below describe the changes **stix-ramrod** performs during an upgrade from STIX v1.1.1 to STIX v1.2.

General Updates

The following general changes are made to STIX v1.1 content when updating to STIX v1.1.1:

- The `xsi:schemaLocation` attribute updated to refer to STIX v1.2 schemas, hosted at http://stix.mitre.org/.
- The `version` attribute on `STIXType` instances set to `1.2`.
- The `version` attribute on `IncidentType` instances set to `1.2`.
- The `version` attribute on `TTPType` instances set to `1.2`.
- The `version` attribute on `CourseOfActionType` instances set to `1.2`.
- The `version` attribute on `ThreatActorType` instances set to `1.2`.
- The `version` attribute on `CampaignType` instances set to `1.2`.
- The `version` attribute on `ExploitTargetType` instances set to `1.2`.
- The `version` attribute on `IndicatorType` instances set to `2.2`.

Untranslatable Fields

No field translations are performed when upgrading from STIX v1.1.1 to STIX v1.2.

Translated Fields

There are no required translations when upgrading from STIX v1.1.1 to STIX v1.2.

Controlled Vocabulary Updates

At a minimum, controlled vocabulary updates include updates to the `vocab_name`, `vocab_reference`, and `xsi:type` attributes to refer to new data type names and versions. Instance values may be updated if typos were fixed in new versions.

The following updates were made to default STIX controlled vocabularies, defined by the `stix_default_vocabularies.xsd` schema.

- `DiscoveryMethodVocab-1.0` updated to `DiscoveryMethodVocab-2.0`.
  - Term `'Fraud Detection'` corrected to `'External - Fraud Detection'`.

Note: Controlled Vocabulary updates can be disabled in **stix-ramrod** via the *`ramrod.UpdateOptions`* class or the `--disable-vocab-update` option if using `ramrod_update.py`.

Empty Optional Fields Removed

No fields were changed from required to optional between STIX v1.1.1 and STIX v1.2.

STIX v1.1 to v1.1.1

STIX v1.1.1 was a bugfix release of the STIX language that fixed incorrect data types, typos, and requirements.

The sections below describe the changes **stix-ramrod** performs during an upgrade from STIX v1.1. to STIX v1.1.1

General Updates

The following general changes are made to STIX v1.1 content when updating to STIX v1.1.1:

- The `xsi:schemaLocation` attribute updated to refer to STIX v1.1 schemas, hosted at http://stix.mitre.org/.
- The `version` attribute on `STIXType` instances set to `1.1.1`.
- The `version` attribute on `IncidentType` instances set to `1.1.1`.
- The `version` attribute on `TTPType` instances set to `1.1.1`.
- The `version` attribute on `CourseOfActionType` instances set to `1.1.1`.
- The `version` attribute on `ThreatActorType` instances set to `1.1.1`.
- The `version` attribute on `CampaignType` instances set to `1.1.1`.
- The `version` attribute on `ExploitTargetType` instances set to `1.1.1`.
- The `version` attribute on `IndicatorType` instances set to `2.1.1`.

Note: STIX v1.1 and STIX v1.1.1 are both tightly integrated with CybOX v2.1. Updating STIX v1.1 content to STIX v1.1.1 will result in CybOX schema locations in the `xsi:schemaLocation` attribute to be updated to point to the schemas hosted at http://cybox.mitre.org/. No other updates to CybOX content are performed.

Untranslatable Fields

All fields can be translated from STIX v1.1 to STIX v1.1.1.

Translated Fields

The following fields and data types are were changed in STIX v1.1 in a manner that requires translation in order to maintain a schema-valid status.

stixCommon:ConfidenceType and stixCommon:StatementType When updating from STIX v1.0.1 to STIX v1.1, instances of `stixCommon:ConfidenceType` and `stixCommon:StatementType` must have their `Source` child elements updated to be instances of `stixCommon:InformationSourceType`.

In STIX v1.1, the `Source` field was of type `stixCommon:ControlledVocabularyStringType`.

In STIX v1.1.1, the `Source` field was updated to be of type `stixCommon:InformationSourceType`, a much richer data type with many more fields.

The value of the STIX v1.1 `Source` field is translated into an instance of `stixCommon:IdentityType`, where the `Source` value becomes the value of the `Name` field under `stixCommon:IdentityType`. The new `stixCommon:IdentityType` instance is assigned to the `Identity` field of the `stixCommon:InformationSourceType Source` field.

Example: A STIX v1.1 `stixCommon:ConfidenceType` instance.

```xml
<stixCommon:Confidence>
    <stixCommon:Source>Example</stixCommon:Source>
</stixCommon:Confidence>
```

Example: A STIX v1.1.1 `stixCommon:ConfidenceType` instance.

```
<stixCommon:Confidence>
    <stixCommon:Source>
        <stixCommon:Identity>
            <stixCommon:Name>Example</stixCommon:Name>
        </stixCommon:Identity>
    </stixCommon:Source>
</stixCommon:Confidence>
```

indicator:SightingType When updating from STIX v1.1 to STIX v1.1.1, instances of `indicator:SightingType` must have their `Source` child element updated to be instances of `stixCommon:InformationSourceType`.

In STIX v1.1, the `Source` field was of type `stixCommon:StructuredTextType`.

In STIX v1.1.1, the `Source` field was updated to be of type `stixCommon:InformationSourceType`, a much richer data type with many more fields.

The value of the STIX v1.1 `Source` field is translated into an instance of `stixCommon:IdentityType`, where the `Source` value becomes the value of the `Name` field under `stixCommon:IdentityType`. The new `stixCommon:IdentityType` instance is assigned to the `Identity` field of the `stixCommon:InformationSourceType` `Source` field.

Example: A STIX v1.1 `indicator:SightingType` instance.

```
<indicator:Sighting>
    <indicator:Source>Example</indicator:Source>
</indicator:Sighting>
```

Example: A STIX v1.1.1 `indicator:SightingType` instance.

```
<indicator:Sighting>
    <indicator:Source>
        <stixCommon:Identity>
            <stixCommon:Name>Foobar</stixCommon:Name>
        </stixCommon:Identity>
    </indicator:Source>
</indicator:Sighting>
```

stixCommon:CampaignReferenceType When updating from STIX v1.1 to STIX v1.1.1, instances of `stixCommon:CampaignReferenceType` must be updated.

In STIX v1.1, the `stixCommon:CampaignReferenceType` contained a child `Names` element, which was of type `stixCommon:NamesType`.

In STIX v1.1.1, the `stixCommon:CampaignReferenceType` was updated to extend the `stixCommon:GenericRelationshipType` and introduced a new `Campaign` element layer as a result.

Example: A STIX v1.1 `stixCommon:CampaignReferenceType` instance.

```
<indicator:Related_Campaigns>
    <indicator:Related_Campaign>
        <stixCommon:Names>
            <stixCommon:Name>Example</stixCommon:Name>
        </stixCommon:Names>
    </indicator:Related_Campaign>
    <indicator:Related_Campaign idref='campaign-foo-1'/>
</indicator:Related_Campaigns>
```

Example: A STIX v1.1.1 `stixCommon:CampaignReferenceType` instance.

```
<indicator:Related_Campaigns>
    <indicator:Related_Campaign>
        <stixCommon:Campaign>
            <stixCommon:Names>
                <stixCommon:Name>Example</stixCommon:Name>
            </stixCommon:Names>
        </stixCommon:Campaign>
    </indicator:Related_Campaign>
    <indicator:Related_Campaign>
        <stixCommon:Campaign idref="campaign-foo-1>
    </indicator:Related_Campaign>
</indicator:Related_Campaigns>
```

Controlled Vocabulary Updates

At a minimum, controlled vocabulary updates include updates to the `vocab_name`, `vocab_reference`, and `xsi:type` attributes to refer to new data type names and versions. Instance values may be updated if typos were fixed in new versions.

The following updates were made to default STIX controlled vocabularies, defined by the `stix_default_vocabularies.xsd` schema.

- `AvailabilityLossVocab-1.0` updated to ``AvailabilityLossVocab-1.1.1`.
    - Term `'Degredation'` corrected to `'Degradation'`.

Note: Controlled Vocabulary updates can be disabled in **stix-ramrod** via the *`ramrod.UpdateOptions`* class or the `--disable-vocab-update` option if using `ramrod_update.py`.

Empty Optional Fields Removed

The following elements were required in STIX v1.1 but became optional in STIX v1.1.1. Empty instances of these fields will be stripped during the update process.

- All child nodes of the Generic Test Mechanism extension instance, `GenericTestMechanismType`.

Note: By default, **stix-ramrod** will remove empty instances of optional fields which are discovered in input content. This behavior can be disabled via the *`ramrod.UpdateOptions`* class, or the `--disable-remove-optionals` flag if using the bundled `ramrod_update.py`

STIX v1.0.1 to v1.1

STIX v1.1 was a minor release which came after STIX v1.0.1. STIX v1.1 introduced a number of new fields, data types, and extensions for capturing and characterizing cyber threat intelligence data.

The sections below describe the changes **stix-ramrod** performs during an upgrade from STIX v1.0.1 to v1.1

General Updates

The following general changes are made to STIX v1.0.1 content when updating to STIX v1.1

- The `xsi:schemaLocation` attribute updated to refer to STIX v1.1 schemas, hosted at http://stix.mitre.org/.
- The `version` attribute on `STIXType` instances set to `1.1`.
- The `version` attribute on `IncidentType` instances set to `1.1`.
- The `version` attribute on `TTPType` instances set to `1.1`.
- The `version` attribute on `CourseOfActionType` instances set to `1.1`.
- The `version` attribute on `ThreatActorType` instances set to `1.1`.
- The `version` attribute on `CampaignType` instances set to `1.1`.
- The `version` attribute on `ExploitTargetType` instances set to `1.1`.
- The `version` attribute on `IndicatorType` instances set to `2.1`.
- Namespace definitions for MAEC 4.0.1 Malware extension removed during translation: `http://stix.mitre.org/extensions/Malware#MAEC4.0-1`
- Namespace definitions for CAPEC 2.6.1 Attack Pattern extension removed during translation: `http://stix.mitre.org/extensions/AP#CAPEC2.6-1`

Note: CybOX v2.0.1 is tightly integrated into STIX v1.0.1. As such, any CybOX 2.0.1 content found within a STIX v1.0.1 document will be updated to CybOX 2.1. See the Updates to CybOX Content page for more details about CybOX content updates with **stix-ramrod**.

Untranslatable Fields

The following fields, data types, attributes or other structures cannot be translated to STIX v1.1. Updating content which includes these fields will require a **forced** update.

- Instances of MAEC 4.0.1 Malware extension `MAEC4.0InstanceType`.
- Instances of CAPEC 2.6.1 Attack Pattern extension `CAPEC2.6InstanceType`.
- Instances of `ttp:Malware` where all children are instances of MAEC 4.0.1 Malware extension.
- Instances of `ttp:Attack_Patterns` where all children are instances of CAPEC 2.6.1 Attack Pattern extension.
- Instances of `stixCommon:Date_Time` that do not have valid `xs:dateTime` values.

Translated Fields

The following fields and data types are were changed in STIX v1.1 in a manner that requires translation in order to maintain a schema-valid status.

stixCommon:Contributors When updating from STIX v1.0.1 to STIX v1.1, instances of `stixCommon:ContributorsType` must be translated to instances of `stixCommon:ContributingSourceType`.

The STIX v1.0.1 ContributorsType contains a list of `Contributor` elements under it which were IdentityType instances.

The STIX v1.1 ContributingSourcesType contains a list of `Source` elements under it which are instances of InformationSourceType.

Because InformationSourceType has an `Identity` child element which is an instance of `IdentityType`, we can perform the following transformation:

Example: A STIX v1.0.1 `ContributorsType` instance.

```
<stixCommon:Contributors>
    <stixCommon:Contributor>
        <stixCommon:Name>Example</stixCommon:Name>
    </stixCommon:Contributor>
    <stixCommon:Contributor>
        <stixCommon:Name>Another</stixCommon:Name>
    </stixCommon:Contributor>
</stixCommon:Contributors>
```

Example: A STIX v1.1 `ContributingSourceType` instance.

```
<stixCommon:Contributing_Sources>
    <stixCommon:Source>
        <stixCommon:Identity>
            <stixCommon:Name>Example</stixCommon:Name>
        </stixCommon:Identity>
    </stixCommon:Source>
    <stixCommon:Source>
        <stixCommon:Identity>
            <stixCommon:Name>Another</stixCommon:Name>
        </stixCommon:Identity>
    </stixCommon:Source>
</stixCommon:Contributing_Sources>
```

ttp:Exploit_Targets When updating from STIX v1.0.1 to STIX v1.1, instances of `stixCommon:ExploitTargetsType` change from a flat list of `stixCommon:ExploitTargetBaseType` instances to an extension of `stixCommon:GenericRelationshipListType`.

Example: A STIX v1.0.1 `ttp:Exploit_Targets` instance.

```
<ttp:Exploit_Targets>
   <stixCommon:Exploit_Target idref='example:et-1'/>
   <stixCommon:Exploit_Target idref='example:et-2'/>
</ttp:Exploit_Targets>
```

Example: A STIX v1.1 `ttp:Exploit_Targets` instance.

```
<ttp:Exploit_Targets>
    <ttp:Exploit_Target>
        <stixCommon:Exploit_Target idref='example:et-1'/>
    </ttp:Exploit_Target>
    <ttp:Exploit_Target>
        <stixCommon:Exploit_Target idref='example:et-2'/>
    </ttp:Exploit_Target>
</ttp:Exploit_Targets>
```

Controlled Vocabulary Updates

At a minimum, controlled vocabulary updates include updates to the `vocab_name`, `vocab_reference`, and `xsi:type` attributes to refer to new data type names and versions. Instance values may be updated if typos were fixed in new versions.

The following updates were made to default STIX controlled vocabularies, defined by the `stix_default_vocabularies.xsd` schema.

- `MotivationVocab-1.0.1` updated to `MotivationVocab-1.1`.
  - Term `'Policital'` corrected to `'Political'`.
- `IndicatorTypeVocab-1.0` updated to `IndicatorTypeVocab-1.1`.

Note: Controlled Vocabulary updates can be disabled in **stix-ramrod** via the *`ramrod.UpdateOptions`* class or the `--disable-vocab-update` option if using `ramrod_update.py`.

Empty Optional Fields Removed

The following elements were required in STIX v1.0.1 but became optional in STIX v1.1. Empty instances of these fields will be stripped during the update process.

- `marking:Controlled_Structure`
- `marking:Marking_Structure`

Note: By default, **stix-ramrod** will remove empty instances of optional fields which are discovered in input content. This behavior can be disabled via the *`ramrod.UpdateOptions`* class, or the `--disable-remove-optionals` flag if using the bundled `ramrod_update.py`

STIX v1.0 to v1.0.1

STIX v1.0.1 was a bugfix release which came after STIX v1.0. Because it is an bugfix release the number of changes is small.

The sections below describe the changes **stix-ramrod** performs during an upgrade from STIX v1.0 to v1.0.1

General Updates

The following general changes are made to STIX v1.0 content when updating to STIX v1.0.1.

- The `xsi:schemaLocation` attribute updated to refer to STIX v1.0.1 schemas, hosted at http://stix.mitre.org/.
- The `version` attribute on `STIXType` instances set to `1.0.1`.
- The `version` attribute on `IncidentType` instances set to `1.0.1`.
- The `version` attribute on `TTPType` instances set to `1.0.1`.
- The `version` attribute on `CourseOfActionType` instances set to `1.0.1`.
- The `version` attribute on `ThreatActorType` instances set to `1.0.1`.
- The `version` attribute on `CampaignType` instances set to `1.0.1`.
- The `version` attribute on `ExploitTargetType` instances set to `1.0.1`.
- The `version` attribute on `IndicatorType` instances set to `2.0.1`.
- Namespace definitions for MAEC 4.0 Malware extension removed during translation: `http://stix.mitre.org/extensions/Malware#MAEC4.0-1`

- Namespace definitions for CAPEC 2.5 Attack Pattern extension removed during translation: `http://stix.mitre.org/extensions/AP#CAPEC2.5-1`

Note: CybOX v2.0 is tightly integrated into STIX v1.0. As such, any CybOX 2.0 content found within a STIX v1.0 document will be updated to CybOX 2.0.1. See the Updates to CybOX Content page for more details about CybOX content updates with **stix-ramrod**.

Untranslatable Fields

The following fields, data types, attributes or other structures cannot be translated to STIX v1.0.1. Updating content which includes these fields will require a **forced** update.

- Instances of MAEC 4.0 Malware extension `MAEC4.0InstanceType`.
- Instances of `ttp:Malware` where all children are instances of MAEC 4.0 Malware extension.
- Instances of CAPEC 2.5 Attack Pattern extension `CAPEC2.5InstanceType`.
- Instances of `ttp:Attack_Patterns` where all children are instances of CAPEC 2.5 Attack Pattern extension.

Controlled Vocabulary Updates

At a minimum, controlled vocabulary updates include updates to the `vocab_name`, `vocab_reference`, and `xsi:type` attributes to refer to new data type names and versions. Instance values may be updated if typos were fixed in new versions.

The following updates were made to default STIX controlled vocabularies, defined by the `stix_default_vocabularies.xsd` schema.

- `MotivationVocab-1.0` updated to `MotivationVocab-1.0.1`.
  - Term `'Ideological - Anti-Establisment'` corrected to `'Ideological - Anti-Establishment'`.
- `PlanningAndOperationalSupportVocab-1.0` updated to `PlanningAndOperationalSupportVocab-1.0.1`.
  - Term `'Planning - Open-Source Intelligence (OSINT) Gethering'` corrected to `'Planning - Open-Source Intelligence (OSINT) Gathering'`
  - Term `'Planning '` corrected to `'Planning'` (trailing space removed)

Note: Controlled Vocabulary updates can be disabled in **stix-ramrod** via the *`ramrod.UpdateOptions`* class or the `--disable-vocab-update` option if using `ramrod_update.py`.

Empty Optional Fields Removed

There are no optional fields that are removed when updating from STIX v1.0 to STIX v1.0.1.

Version: 1.2.0

1.3.2 Updates to CybOX Content

The following sections detail the changes that **stix-ramrod** makes when upgrading CybOX content.

Note: The sections below do not detail the full breadth and depth of changes made to the CybOX language between releases. Some updates, such as the addition of new CybOX Objects or controlled vocabulary terms do not require any changes to be made to the source content during an update.

To see a complete list of changes made to CybOX, check out the *Release Notes* section of a CybOX Language Release page.

Contents

CybOX v2.0.1 to 2.1

CybOX v2.1 was a minor release made to CybOX v2.0.1 and included many bug fixes, some of which resulted in backwards incompatibilities with previous versions of CybOX. On top of bug fixes, CybOX 2.1 introduced new controlled vocabularies and terms, CybOX Objects, data types and backwards-compatible structural enhancements.

The sections below describe the changes **stix-ramrod** performs during an upgrade from CybOX 2.0.1 to CybOX 2.1.

General Updates

The following general changes are made to CybOX 2.0.1 content when updating to CybOX 2.1.

- The `xsi:schemaLocation` attribute updated to refer to CybOX 2.1 schemas, hosted at http://cybox.mitre.org/.
- The `cybox_major_version` attribute on `ObservableType` instances set to `2`.
- The `cybox_minor_version` attribute on `ObservableType` instances set to `1`.
- The `cybox_update_version` attribute removed from `ObservablesType` instances.

Untranslatable Fields

The following fields, data types, attributes or other structures cannot be translated to CybOX v2.1. Updating content which includes these fields will require a **forced** update.

- `HTTPSessionObj:X_Forwarded_Proto` element instances.
- `Type` elements instances found in `WinExecutableFileObj:PESectionType`.
- `WinMailslotObj:Handle` element instances when it contains more than one child `Handle` element.

- `WinTaskObj:Trigger_Type` element instances.

Object Updates

The following changes are made to CybOX Objects.

HTTP Session Object

- `HTTPSessionObj:DNT` element data type changed from `URIObj:URIObjectType` to `cyboxCommon:StringObjectPropertyType`.
- `HTTPSessionObj:Vary` element data type changed from `URIObj:URIObjectType` to `cyboxCommon:StringObjectPropertyType`.
- `HTTPSessionObj:Refresh` updated from `cyboxCommon:IntegerObjectPropertyType` to `cyboxCommon:StringObjectPropertyType`

Network Packet Object

- `PacketObj:Protol_Addr_Size` renamed to `PacketObj:Proto_Addr_Size`
- `PacketObj:Excapsulating_Security_Payload` renamed to `PacketObj:Encapsulating_Security_Paylo`
- `PacketObj:Authenication_Data` renamed to `PacketObj:Authentication_Data`

Windows Driver Object

- The version of the `Win_Driver_Object.xsd` schema, which defines the Windows Driver Object was upgraded to `3.0`.
- The namespace for the Windows Driver Object was changed from `http://cybox.mitre.org/objects#WinDriverObject-2` to `'http://cybox.mitre.org/objects#WinD`

Windows Mailslot Object

- The top-level `WinMailslotObj:Handle` container is removed, causing `Handle` child to take it its place. This can only be done if there is one `Handle` child. If more than one child `Handle` element is present, the top-level `WinMailslotObj:Handle` container is considered untranslatable.

  Example CybOX 2.0.1 WinMailslotObj:Handle

```
<cybox:Object>
    <cybox:Properties xsi:type="WinMailslotObj:WindowsMailslotObjectType">
        <WinMailslotObj:Handle>
            <WinHandleObj:Handle>
                <WinHandleObj:Name>Test</WinHandleObj:Name>
            </WinHandleObj:Handle>
        </WinMailslotObj:Handle>
    </cybox:Properties>
</cybox:Object>
```

  Example CybOX 2.1 WinMailslotObj:Handle

```
<cybox:Object>
    <cybox:Properties xsi:type="WinMailslotObj:WindowsMailslotObjectType">
        <WinHandleObj:Handle>
            <WinHandleObj:Name>Test</WinHandleObj:Name>
        </WinHandleObj:Handle>
    </cybox:Properties>
</cybox:Object>
```

Example Untranslatable CybOX 2.0.1 WinMailslotObj:Handle

```
    <cybox:Object>
        <cybox:Properties xsi:type="WinMailslotObj:WindowsMailslotObjectType">
            <WinMailslotObj:Handle>
                <WinHandleObj:Handle>
                    <WinHandleObj:Name>One Child</WinHandleObj:Name>
                </WinHandleObj:Handle>
                <WinHandleObj:Handle>
                    <WinHandleObj:Name>Cannot translate! Two Handle children present.</WinHandleObj:
                </WinHandleObj:Handle>
            </WinMailslotObj:Handle>
        </cybox:Properties>
    </cybox:Object>
```

Controlled Vocabulary Updates

At a minimum, controlled vocabulary updates include updates to the `vocab_name`, `vocab_reference`, and `xsi:type` attributes to refer to new data type names and versions. Instance values may be updated if typos were fixed in new versions.

The following updates were made to default CybOX controlled vocabularies, defined by the `cybox_default_vocabularies.xsd` schema.

- `ToolTypeVocab-1.0` updated to `ToolTypeVocab-1.1`.
  - Term `'A/V'` changed to `'AV'`.
- `ObjectRelationshipVocab-1.0` updated to `ObjectRelationshipVocab-1.1`.
- `ActionNameVocab-1.0` updated to `ActionNameVocab-1.1`.

Note: Controlled Vocabulary updates can be disabled in **stix-ramrod** via the *`ramrod.UpdateOptions`* class or the `--disable-vocab-update` option if using `ramrod_update.py`.

Empty Optional Fields Removed

The following elements were required in CybOX 2.0.1 but became optional in CybOX 2.1. Empty instances of these fields will be stripped during the update process.

- `DiskPartitionObj:Partition_ID`
- `DNSCacheObj:DNS_Entry`
- `DNSQueryObj:QName`
- `FileObj:Depth`
- `HTTPSessionObj:Message_Body`, `HTTPSessionObj:Domain_Name`
- `PacketObj:Address_Mask`, `PacketObj:Address_Mask_Reply`, `PacketObj:Address_Mask_Request`, `PacketObj:Destination_Unreachable`, `PacketObj:Echo_Reply`, `PacketObj:Echo_Request`, `PacketObj:Error_Msg`, `PacketObj:Frag_Reassembly_Time_Exceeded`, `PacketObj:Host_Redirect`,

  `PacketObj:IP_Addr_Prefix`, `PacketObj:IPv6_Addr`, `PacketObj:Info_Msg`, `PacketObj:Network_Redirect`, `PacketObj:Outbound_Packet_Forward_Success`, `PacketObj:Outbound_Packet_no_Route`, `PacketObj:Receive_Timestamp`, `PacketObj:Redirect_Message`, `PacketObj:Source_Quench`, `PacketObj:TTL_Exceeded_In_Transit`, `PacketObj:Time_Exceeded`, `PacketObj:Timestamp`, `PacketObj:Timestamp_Reply`, `PacketObj:Timestamp_Request`, `PacketObj:ToS_Host_Redirect`, `PacketObj:ToS_Network_Redirect`, `PacketObj:Traceroute`, `PacketObj:Transmit_Timestamp`

- `SystemObj:IP_Address`
- `URIObj:Value`
- `WinComputerAccountObj:Delegation`, `WinComputerAccountObj:Bitmask`, `WinComputerAccountObj:Service`
- `WinFileObj:Size_In_Bytes`
- `WinNetworkShareObj:Netname`
- `WinPrefetchObj:VolumeItem`, `WinPrefetchObj:DeviceItem`

Note: By default, **stix-ramrod** will remove empty instances of optional fields which are discovered in input content. This behavior can be disabled via the *`ramrod.UpdateOptions`* class, or the `--disable-remove-optionals` flag if using the bundled `ramrod_update.py`

CybOX v2.0 to v2.0.1

CybOX v2.0.1 was a bugfix release made to CybOX v2.0.1.

The sections below describe the changes **stix-ramrod** performs during an upgrade from CybOX 2.0 to CybOX 2.0.1.

General Updates

The following general changes are made to CybOX 2.0 content when updating to CybOX 2.0.1:

- The `xsi:schemaLocation` attribute updated to refer to CybOX 2.0.1 schemas, hosted at http://cybox.mitre.org/.
- The `cybox_major_version` attribute on `ObservableType` instances set to `2`.
- The `cybox_minor_version` attribute on `ObservableType` instances set to `0`.
- The `cybox_minor_version` attribute added to `ObservablesType` instances and set to `1`.

List Delimiters CybOX 2.0 allows for the definition of multiple Object Property field values through the use of a reserved list delimiter, which is defined to be `','` (a comma). Grammatical commas were expressed as `<![CDATA[&comma;]]>`.

CybOX 2.0.1 changed the reserved list delimiter to be `'##comma##'`, allowing for grammatical commas to be expressed without special syntax or `CDATA` wrappers.

Example CybOX 2.0 List

```
<!-- Describes two email subjects: 'Foo' and 'Bar' -->
<EmailObj:Subject condition="Equals" apply_condition="ANY">Foo,Bar</EmailObj:Subject>
```

Example CybOX 2.0 Grammatical Comma

```
<!-- Use of a grammatical comma -->
<EmailObj:Subject>Et tu<![CDATA[&comma;]]> Brute?</EmailObj:Subject>
```

CybOX 2.0.1 changed the default list delimiter to be `'##comma##'`, allowing for grammatical commas to be used naturally.

Example CybOX 2.0.1 List

```
<!-- Describes two email subjects: 'Foo' and 'Bar' -->
<EmailObj:Subject condition="Equals" apply_condition="ANY">Foo##comma##Bar</EmailObj:Subject>
```

Example CybOX 2.0.1 Grammatical Comma

```
<!-- Use of a grammatical comma -->
<EmailObj:Subject>Et tu, Brute?</EmailObj:Subject>
```

Untranslatable Fields

All CybOX 2.0 fields can be translated to 2.0.1. There should not be any need to perform a **forced** update.

Object Updates

All CybOX 2.0 Object content can be updated to 2.0.1 without any structural transformations or content translations.

Controlled Vocabulary Updates

At a minimum, controlled vocabulary updates include updates to the `vocab_name`, `vocab_reference`, and `xsi:type` attributes to refer to new data type names and versions. Instance values may be updated if typos were fixed in new versions.

The following updates were made to default CybOX controlled vocabularies, defined by the `cybox_default_vocabularies.xsd` schema.

- `EventTypeVocab-1.0` updated to `EventTypeVocab-1.0.1`
  - Term `'Anomoly Events'` changed to `'Anomaly Events'`

Note: Controlled Vocabulary updates can be disabled in **stix-ramrod** via the *`ramrod.UpdateOptions`* class or the `--disable-vocab-update` option if using `ramrod_update.py`.

Empty Optional Fields Removed

No existing fields were made optional in CybOX 2.0.1.

CHAPTER 2

API Reference

Version: 1.2.0

2.1 API Documentation

The **stix-ramrod** APIs provide methods for updating STIX and CybOX XML content. Listed below are the modules and packages provided by the **stix-ramrod** library.

For examples of how make use of all of this, check out the Example Code page.

Note: The **stix-ramrod** APIs are currently under heavy development. Feel free to check out our issue tracker to see what we're working on!

Version: 1.2.0

2.1.1 `ramrod` Module

`ramrod.`**`update`**(*doc*, *from_=None*, *to_=None*, *options=None*, *force=False*)

Updates an input STIX or CybOX document to align with a newer version of the STIX/CybOX schemas.

This will perform the following updates:

- Update namespaces
- Update schemalocations
- Update construct versions (`STIX_Package`, `Observables`, etc.)
- Update controlled vocabularies and fix typos
- Translate structures to new XSD data type instances where possible.
- Remove empty instances of attributes and elements which were required in one version of the language and declared optional in another.

Parameters

- **`doc`** – A STIX or CybOX document filename, file-like object, `etree._Element` or `etree._ElementTree` object instance.
- **`to`** (*optional, string*) – The expected output version of the update process. If not specified, the latest language version will be assumed.

- **from** (*optional, string*) – The version to update from. If not specified, the *from_* version will be retrieved from the input document.
- **options** (*optional*) – A `UpdateOptions` instance. If `None`, `ramrod.DEFAULT_UPDATE_OPTIONS` will be used.
- **force** (*boolean*) – Attempt to force the update process if the document contains untranslatable fields.

Returns An instance of `UpdateResults`.

Raises

- `UpdateError` – If any of the following occur:
  - The input *doc* does not contain a `STIX_Package` or `Observables` root-level node.
  - If'force' is `False` and an untranslatable field or non-unique ID is found in the input *doc*.
- `InvalidVersionError` – If the input document contains a version attribute that is incompatible with a STIX/CybOX Updater class instance.
- `UnknownVersionError` – If *from_* was not specified and the input document does not contain a version attribute.

class `ramrod.`**`UpdateOptions`**

Bases: `object`

Defines configurable options for STIX/CybOX updates.

`check_versions`

If `True`, input document version information will be collected and checked against what the Updater class expects. If `False` no version check operations will be performed. Default value is `True`.

`new_id_func`

A function for setting new IDs on an `etree._Element` node. The function must accept one `etree._Element` instance argument and assign it a new, unique `id` attribute value. Default value is `ramrod.utils.new_id()` function.

Example

```
>>> def my_id_func(node):
>>>     new_id = my_generate_unique_id()
>>>     node.attrib['id'] = new_id
>>>
>>> options = ramrod.UpdateOptions()
>>> options.new_id_func = my_id_func
```

`update_vocabularies`

If `True`, default controlled vocabulary instances will be updated and typos will be fixed. If `False`, no updates will be performed against controlled vocabulary instances. Default is `True`.

`remove_optionals`

Between revisions of language, some elements which were required are made optional. If `True`, an attempt is made to find and remove empty instances of once required elements/attributes. Default is `True`.

class `ramrod.`**`UpdateResults`**(*document*, *removed=None*, *remapped_ids=None*)

Bases: `object`

Returned from `ramrod.update()`, `ramrod.cybox.update()`, and `ramrod.stix.update()` methods.

document
The updated document. An instance of *ramrod.ResultDocument*.

removed
Untranslatable nodes that were removed from the document. An instance of `tuple`.

remapped_ids
An `{ id: [nodes] }` dictionary where the key is a non-unique ID that was discovered in the input document, and the nodes are all the nodes which had their `id` attribute reassigned to be unique.

document

class ramrod.**ResultDocument**(*document*)
Bases: `object`

Used to encapsulate an updated XML document. This is the type of the `document` attribute on *ramrod.UpdateResults*

Note: This class overrides the `__str__` and `__unicode__` methods and can be used with `str()` or `print`.

Parameters **document** – An instance of `etree._Element` or `etree._ElementTree`.

as_element()
Returns an `etree._Element` representation of the `ResultDocument` instance.

as_element_tree()
Returns an `etree._ElementTree` representation of the `ResultDocument` instance.

as_stringio()
Returns a `StringIO` representation of the `ResultDocument` instance.

Version: 1.2.0

2.1.2 ramrod.errors Module

exception ramrod.errors.**UnknownVersionError**
Bases: `exceptions.Exception`

Raised when an input document does not contain a `version` attribute and the user has not specified a document version.

exception ramrod.errors.**UpdateError**(*message=None*, *disallowed=None*, *duplicates=None*)
Bases: `exceptions.Exception`

Raised when non-translatable fields are encountered during the update process..

message
The error message.

disallowed
A list of nodes found in the input document that cannot be translated during the update process.

duplicates
A dictionary of nodes found in the input document that contain the same *id* attribute value.

exception ramrod.errors.**InvalidVersionError**(*message=None*, *node=None*, *expected=None*, *found=None*)
Bases: `exceptions.Exception`

Raised when an input document's `version` attribute does not align with the expected version number for a given `_BaseUpdater` implementation.

`message`
The error message.

`node`
The node containing an incompatible version number.

`expected`
The version that was expected.

`found`
The version that was found on the *node*.

Version: 1.2.0

2.1.3 `ramrod.cybox` Module

`ramrod.cybox.`**`get_version`**`(`*doc*`)`
Returns the version number for input CybOX document.

`ramrod.cybox.`**`register_updater`**`(`*cls*`)`
Registers a CybOX updater class.

`ramrod.cybox.`**`update`**`(`*doc*, *from_=None*, *to_=None*, *options=None*, *force=False*`)`
Updates a CybOX document to align with a given version of the CybOX Language.

Parameters

- **`doc`** – A CybOX document filename, file-like object, `etree._Element`, or `etree._ElementTree`.
- **`from`** (*optional, string*) – The base version for the update process. If `None`, an attempt will be made to extract the version number from *doc*.
- **`to`** (*optional, string*) – The version to update to. If `None`, the latest version of CybOX is assumed.
- **`options`** (*optional*) – A *`ramrod.UpdateOptions`* instance. If `None`, `ramrod.DEFAULT_UPDATE_OPTIONS` will be used.
- **`force`** (*boolean*) – Forces the update process. This may result in content being removed during the update process and could result in schema-invalid content. **Use at your own risk!**

Returns An instance of `ramrod.UpdateResults`.

Raises

- *`UpdateError`* – If any of the following conditions are encountered:
  - The *from_* or *to_* versions are invalid.
  - An untranslatable field is encountered and *force* is `False`.
  - A non-unique ID is encountered and *force* is `False`.
- *`InvalidVersionError`* – If the source document version and the *from_* value do not match and *force* is `False`.
- *`UnknownVersionError`* – If the source document does not contain version information and *force* is `False`.

Version: 1.2.0

2.1.4 `ramrod.cybox.cybox_2_0` Module

class `ramrod.cybox.cybox_2_0.`**`Cybox_2_0_Updater`**

Updates CybOX v2.0 content to CybOX v2.0.1.

The following fields are translated:

- `EventTypeVocab-1.0` updated to `EventTypeVocab-1.0.1`

Note: All fields can be translated from CybOX v2.0 to CybOX v2.0.1

`check_update`(*root*, *options=None*)

Determines if the input document can be upgraded.

Parameters

- **`root`** – The XML document. This can be a filename, a file-like object, an instance of `etree._Element` or an instance of `etree._ElementTree`.
- **`options`** (*optional*) – A `ramrod.UpdateOptions` instance. If `None`, `ramrod.DEFAULT_UPDATE_OPTIONS` will be used.

Raises

- *`UnknownVersionError`* – If the input document does not have a version.
- *`InvalidVersionError`* – If the version of the input document does not match the *VERSION* class-level attribute value.
- *`UpdateError`* – If the input document contains fields which cannot be updated or constructs with non-unique IDs are discovered.

`clean`(*root*, *options=None*)

Removes disallowed elements from *root* and remaps non-unique IDs to unique IDs for the sake of schema-validation.

Removed items can be retrieved via the `removed` attribute on the return value:

```
>>> results = updater.clean(root)
>>> print(results.removed)
(<Element at 0xffdcf234>, <Element at 0xffdcf284>)
```

Items which have been reassigned IDs can be retrieved via the `remapped_ids` attribute on the return value:

```
>>> results = updater.clean(root)
>>> print(results.remapped_ids)
{'example:Observable-duplicate': [<Element {http://cybox.mitre.org...
```

Note: This does not remap `idref` attributes to new ID values because it is impossible to determine which entity the `idref` was pointing to.

Parameters

- **`root`** – The XML document. This can be a filename, a file-like object, an instance of `etree._Element` or an instance of `etree._ElementTree`.
- **`options`** (*optional*) – A *`ramrod.UpdateOptions`* instance. If `None`, `ramrod.DEFAULT_UPDATE_OPTIONS` will be used.

Returns An instance of *`ramrod.UpdateResults`*.

get_version(*observables*)

Returns the version of the *observables* `Observables` node.

Returns A dotted-decimal a version string from the `cybox_major`, `cybox_minor` and `cybox_update` attribute values.

Raises `UnknownVersionError` – If *observables* does not contain any of the following attributes:

- `cybox_major_version`
- `cybox_minor_version`
- `cybox_update_version`

update(*root*, *options=None*, *force=False*)

Attempts to update *root* to the next version of its language specification.

If *force* is set to True, items may be removed during the translation process and IDs may be reassigned if they are not unique within the document.

Note: This does not remap `idref` attributes to new ID values because it is impossible to determine which entity the `idref` was pointing to.

Removed items can be retrieved via the `removed` attribute on the return value:

```
>>> results = updater.update(root, force=True)
>>> print(results.removed)
(<Element at 0xffdcf234>, <Element at 0xffdcf284>)
```

Items which have been reassigned IDs can be retrieved via the `remappped_ids` attribute on the return value:

```
>>> results = updater.update(root, force=True)
>>> print(results.remapped_ids)
{'example:Observable-duplicate-id-1': [<Element {http://cybox.mitre...
```

Parameters

- **root** – The XML document. This can be a filename, a file-like object, an instance of `etree._Element` or an instance of `etree._ElementTree`.
- **options** – A `ramrod.UpdateOptions` instance. If `None`, `ramrod.DEFAULT_UPDATE_OPTIONS` will be used.
- **force** – Forces the update process to complete by potentially removing untranslatable xml nodes and/or remapping non-unique IDs. This may result in non-schema=conformant XML. **USE AT YOUR OWN RISK!**

Returns An instance of `ramrod.UpdateResults`.

Raises

- `UpdateError` – If untranslatable fields or non-unique IDs are discovered in *root* and *force* is `False`.
- `UnknownVersionError` – If the *root* node contains no version information.
- `InvalidVersionError` – If the *root* node contains invalid version information (e.g., the class expects v1.0 content and the *root* node contains v1.1 content).

Version: 1.2.0

2.1.5 `ramrod.cybox.cybox_2_0_1` Module

class `ramrod.cybox.cybox_2_0_1.`**`Cybox_2_0_1_Updater`**

Updates CybOX v2.0.1 content to CybOX v2.1.

The following fields are translated:

- `ToolTypeVocab-1.0` updated to `ToolTypeVocab-1.1`
- `ObjectRelationshipVocab-1.0` updated to `ObjectRelationshipVocab-1.1`
- `ActionNameVocab-1.0` updated to `ActionNameVocab-1.1`
- `HTTPSessionObj:DNT` updated from `URIObjectType` to `StringObjectPropertyType`
- `HTTPSessionObj:Vary` updated from `URIObjectType` to `StringObjectPropertyType`
- `HTTPSessionObj:Refresh` updated from `IntegerObjectPropertyType` to `StringObjectPropertyType`
- `PacketObj:Protol_Addr_Size` renamed to `PacketObj:Proto_Addr_Size`
- `PacketObj:Excapsulating_Security_Payload` renamed to `PacketObj:Encapsulating_Security_Payload`
- `PacketObj:Authenication_Data` renamed to `PacketObj:Authentication_Data`
- `WinMailslotObj:Handle` container element removed and child bubbled up when only one child is defined.

Empty instances of the following optional elements are removed:

- `DiskPartitionObj:Partition_ID`
- `DNSCacheObj:DNS_Entry`
- `DNSQueryObj:QName`
- `FileObj:Depth`
- `HTTPSessionObj:Message_Bod`, `HTTPSessionObj:Domain_Name`
- `PacketObj:Address_Mask`, `PacketObj:Address_Mask_Reply`, `PacketObj:Address_Mask_Request`, `PacketObj:Destination_Unreachable`, `PacketObj:Echo_Reply`, `PacketObj:Echo_Request`, `PacketObj:Error_Msg`, `PacketObj:Frag_Reassembly_Time_Exceeded`, `PacketObj:Host_Redirect`, `PacketObj:IP_Addr_Prefix`, `PacketObj:IPv6_Addr`, `PacketObj:Info_Msg`, `PacketObj:Network_Redirect`, `PacketObj:Outbound_Packet_Forward_Success`, `PacketObj:Outbound_Packet_no_Route`, `PacketObj:Receive_Timestamp`, `PacketObj:Redirect_Message`, `PacketObj:Source_Quench`, `PacketObj:TTL_Exceeded_In_Transit`, `PacketObj:Time_Exceeded`, `PacketObj:Timestamp`, `PacketObj:Timestamp_Reply`, `PacketObj:Timestamp_Request`, `PacketObj:ToS_Host_Redirect`, `PacketObj:ToS_Network_Redirect`, `PacketObj:Traceroute`, `PacketObj:Transmit_Timestamp`
- `SystemObj:IP_Address`
- `URIObj:Value`
- `WinComputerAccountObj:Delegation`, `WinComputerAccountObj:Bitmask`, `WinComputerAccountObj:Service`
- `WinFileObj:Size_In_Bytes`
- `WinNetworkShareObj:Netname`

- `WinPrefetchObj:VolumeItem`, `WinPrefetchObj:DeviceItem`

The following fields **cannot** be translated:

- `HTTPSession:X_Forwarded_Proto` instances.
- `WinExecutableFileObj:PESectionType/Type` instances.
- `WinMailslotObj:Handle` when more than one child is defined.
- `WinTaskObj:Trigger_Type` instances.

`check_update`(*root*, *options=None*)

Determines if the input document can be upgraded.

Parameters

- **`root`** – The XML document. This can be a filename, a file-like object, an instance of `etree._Element` or an instance of `etree._ElementTree`.
- **`options`** (*optional*) – A `ramrod.UpdateOptions` instance. If `None`, `ramrod.DEFAULT_UPDATE_OPTIONS` will be used.

Raises

- *`UnknownVersionError`* – If the input document does not have a version.
- *`InvalidVersionError`* – If the version of the input document does not match the *VERSION* class-level attribute value.
- *`UpdateError`* – If the input document contains fields which cannot be updated or constructs with non-unique IDs are discovered.

`clean`(*root*, *options=None*)

Removes disallowed elements from *root* and remaps non-unique IDs to unique IDs for the sake of schema-validation.

Removed items can be retrieved via the `removed` attribute on the return value:

```
>>> results = updater.clean(root)
>>> print(results.removed)
(<Element at 0xffdcf234>, <Element at 0xffdcf284>)
```

Items which have been reassigned IDs can be retrieved via the `remapped_ids` attribute on the return value:

```
>>> results = updater.clean(root)
>>> print(results.remapped_ids)
{'example:Observable-duplicate': [<Element {http://cybox.mitre.org...
```

Note: This does not remap `idref` attributes to new ID values because it is impossible to determine which entity the `idref` was pointing to.

Parameters

- **`root`** – The XML document. This can be a filename, a file-like object, an instance of `etree._Element` or an instance of `etree._ElementTree`.
- **`options`** (*optional*) – A *`ramrod.UpdateOptions`* instance. If `None`, `ramrod.DEFAULT_UPDATE_OPTIONS` will be used.

Returns An instance of *`ramrod.UpdateResults`*.

get_version(*observables*)

Returns the version of the *observables* `Observables` node.

Returns A dotted-decimal a version string from the `cybox_major`, `cybox_minor` and `cybox_update` attribute values.

Raises *UnknownVersionError* – If *observables* does not contain any of the following attributes:

- `cybox_major_version`
- `cybox_minor_version`
- `cybox_update_version`

update(*root*, *options=None*, *force=False*)

Attempts to update *root* to the next version of its language specification.

If *force* is set to True, items may be removed during the translation process and IDs may be reassigned if they are not unique within the document.

Note: This does not remap `idref` attributes to new ID values because it is impossible to determine which entity the `idref` was pointing to.

Removed items can be retrieved via the `removed` attribute on the return value:

```
>>> results = updater.update(root, force=True)
>>> print(results.removed)
(<Element at 0xffdcf234>, <Element at 0xffdcf284>)
```

Items which have been reassigned IDs can be retrieved via the `remappped_ids` attribute on the return value:

```
>>> results = updater.update(root, force=True)
>>> print(results.remapped_ids)
{'example:Observable-duplicate-id-1': [<Element {http://cybox.mitre...
```

Parameters

- **root** – The XML document. This can be a filename, a file-like object, an instance of `etree._Element` or an instance of `etree._ElementTree`.
- **options** – A *ramrod.UpdateOptions* instance. If `None`, `ramrod.DEFAULT_UPDATE_OPTIONS` will be used.
- **force** – Forces the update process to complete by potentially removing untranslatable xml nodes and/or remapping non-unique IDs. This may result in non-schema=conformant XML. **USE AT YOUR OWN RISK!**

Returns An instance of `ramrod.UpdateResults`.

Raises

- *UpdateError* – If untranslatable fields or non-unique IDs are discovered in *root* and *force* is `False`.
- *UnknownVersionError* – If the *root* node contains no version information.
- *InvalidVersionError* – If the *root* node contains invalid version information (e.g., the class expects v1.0 content and the *root* node contains v1.1 content).

Version: 1.2.0

2.1.6 `ramrod.stix` Module

`ramrod.stix.`**`get_version`**`(`*doc*`)`
Returns the version number for input STIX document.

`ramrod.stix.`**`register_updater`**`(`*cls*`)`
Registers a STIX updater class.

`ramrod.stix.`**`update`**`(`*doc*, *from_=None*, *to_=None*, *options=None*, *force=False*`)`
Updates a STIX document to align with a given version of the STIX Language schemas.

Parameters

- **`doc`** – A STIX document filename, file-like object, `etree._Element`, or `etree._ElementTree`.
- **`from`** (*optional, string*) – The base version for the update process. If `None`, an attempt will be made to extract the version number from *doc*.
- **`to`** (*optional, string*) – The version to update to. If `None`, the latest version of STIX is assumed.
- **`options`** (*optional*) – A *`ramrod.UpdateOptions`* instance. If `None`, `ramrod.DEFAULT_UPDATE_OPTIONS` will be used.
- **`force`** (*boolean*) – Forces the update process. This may result in content being removed during the update process and could result in schema-invalid content. **Use at your own risk!**

Returns An instance of `ramrod.UpdateResults`.

Raises

- *`UpdateError`* – If any of the following conditions are encountered:
  - The *from_* or *to_* versions are invalid.
  - An untranslatable field is encountered and *force* is `False`.
  - A non-unique ID is encountered and *force* is `False`.
- *`InvalidVersionError`* – If the source document version and the *from_* value do not match and *force* is `False`.
- *`UnknownVersionError`* – If the source document does not contain version information and *force* is `False`.

Version: 1.2.0

2.1.7 `ramrod.stix.stix_1_0` Module

class `ramrod.stix.stix_1_0.`**`STIX_1_0_Updater`**
Updates STIX v1.0 content to STIX v1.0.1.

The following fields and types are translated:

- `MotivationVocab-1.0` upgraded to `MotivationVocab-1.0.1`
- `PlanningAndOperationalSupportVocab-1.0` updated to `PlanningAndOperationalSupportVocab 1.0.1`

The following fields and types **cannot** be translated:

- MAEC 4.0 Malware extension instances

- CAPEC 2.5 Attack Pattern extension instances
- `TTP:Malware` nodes that contain only MAEC Malware_Instance children
- `TTP:Attack_Patterns` nodes that contain only CAPEC Attack Pattern instance children

check_update(*root*, *options=None*)

Determines if the input document can be upgraded.

Parameters

- **root** – The XML document. This can be a filename, a file-like object, an instance of `etree._Element` or an instance of `etree._ElementTree`.
- **options** (*optional*) – A `ramrod.UpdateOptions` instance. If `None`, `ramrod.DEFAULT_UPDATE_OPTIONS` will be used.

Raises

- *UnknownVersionError* – If the input document does not have a version.
- *InvalidVersionError* – If the version of the input document does not match the *VERSION* class-level attribute value.
- *UpdateError* – If the input document contains fields which cannot be updated or constructs with non-unique IDs are discovered.

clean(*root*, *options=None*)

Removes disallowed elements from *root* and remaps non-unique IDs to unique IDs for the sake of schema-validation.

Removed items can be retrieved via the `removed` attribute on the return value:

```
>>> results = updater.clean(root)
>>> print(results.removed)
(<Element at 0xffdcf234>, <Element at 0xffdcf284>)
```

Items which have been reassigned IDs can be retrieved via the `remapped_ids` attribute on the return value:

```
>>> results = updater.clean(root)
>>> print(results.remapped_ids)
{'example:Observable-duplicate': [<Element {http://cybox.mitre.org...
```

Note: This does not remap `idref` attributes to new ID values because it is impossible to determine which entity the `idref` was pointing to.

Parameters

- **root** – The XML document. This can be a filename, a file-like object, an instance of `etree._Element` or an instance of `etree._ElementTree`.
- **options** (*optional*) – A *ramrod.UpdateOptions* instance. If `None`, `ramrod.DEFAULT_UPDATE_OPTIONS` will be used.

Returns An instance of *ramrod.UpdateResults*.

get_version(*package*)

Returns the version of the *package* `STIX_Package` element by inspecting its `version` attribute.

update(*root*, *options=None*, *force=False*)

Attempts to update *root* to the next version of its language specification.

If *force* is set to True, items may be removed during the translation process and IDs may be reassigned if they are not unique within the document.

Note: This does not remap `idref` attributes to new ID values because it is impossible to determine which entity the `idref` was pointing to.

Removed items can be retrieved via the `removed` attribute on the return value:

```
>>> results = updater.update(root, force=True)
>>> print(results.removed)
(<Element at 0xffdcf234>, <Element at 0xffdcf284>)
```

Items which have been reassigned IDs can be retrieved via the `remappped_ids` attribute on the return value:

```
>>> results = updater.update(root, force=True)
>>> print(results.remapped_ids)
{'example:Observable-duplicate-id-1': [<Element {http://cybox.mitre...
```

Parameters

- **`root`** – The XML document. This can be a filename, a file-like object, an instance of `etree._Element` or an instance of `etree._ElementTree`.
- **`options`** – A *`ramrod.UpdateOptions`* instance. If `None`, `ramrod.DEFAULT_UPDATE_OPTIONS` will be used.
- **`force`** – Forces the update process to complete by potentially removing untranslatable xml nodes and/or remapping non-unique IDs. This may result in non-schema=conformant XML. **USE AT YOUR OWN RISK!**

Returns An instance of `ramrod.UpdateResults`.

Raises

- *`UpdateError`* – If untranslatable fields or non-unique IDs are discovered in *root* and *force* is `False`.
- *`UnknownVersionError`* – If the *root* node contains no version information.
- *`InvalidVersionError`* – If the *root* node contains invalid version information (e.g., the class expects v1.0 content and the *root* node contains v1.1 content).

Version: 1.2.0

2.1.8 `ramrod.stix.stix_1_0_1` Module

class `ramrod.stix.stix_1_0_1.`**`STIX_1_0_1_Updater`**

Updates STIX v1.0.1 content to STIX v1.1.

The following fields and types are translated:

- `MotivationVocab-1.0.1` updated to `MotivationVocab-1.1`
- `IndicatorTypeVocab-1.0` updated to `IndicatorTypeVocab-1.1`
- `TTP/Exploit_Targets` instances are updated to align with `stixCommon:GenericRelationshipListType` data type.
- Instances of STIX v1.0.1 `stixCommon:ContributorsType` are converted into instances of STIX v1.1 `stixCommon:ContributingSourcesType`

Empty instances of the following optional items are removed:

- `marking:Controlled_Structure`
- `marking:Marking_Structure`

The following fields and types **cannot** be translated:

- MAEC 4.0.1 Malware extension
- CAPEC 2.6.1 Attack Pattern extension
- `TTP:Malware` nodes that contain only MAEC Malware_Instance children
- `TTP:Attack_Patterns` nodes that contain only CAPEC Attack Pattern instance children
- `stixCommon:Date_Time` fields that do not contain `xs:dateTime` values

`check_update`(*root*, *options=None*)

Determines if the input document can be upgraded.

Parameters

- **`root`** – The XML document. This can be a filename, a file-like object, an instance of `etree._Element` or an instance of `etree._ElementTree`.
- **`options`** (*optional*) – A `ramrod.UpdateOptions` instance. If `None`, `ramrod.DEFAULT_UPDATE_OPTIONS` will be used.

Raises

- *`UnknownVersionError`* – If the input document does not have a version.
- *`InvalidVersionError`* – If the version of the input document does not match the *VERSION* class-level attribute value.
- *`UpdateError`* – If the input document contains fields which cannot be updated or constructs with non-unique IDs are discovered.

`clean`(*root*, *options=None*)

Removes disallowed elements from *root* and remaps non-unique IDs to unique IDs for the sake of schema-validation.

Removed items can be retrieved via the `removed` attribute on the return value:

```
>>> results = updater.clean(root)
>>> print(results.removed)
(<Element at 0xffdcf234>, <Element at 0xffdcf284>)
```

Items which have been reassigned IDs can be retrieved via the `remapped_ids` attribute on the return value:

```
>>> results = updater.clean(root)
>>> print(results.remapped_ids)
{'example:Observable-duplicate': [<Element {http://cybox.mitre.org...
```

Note: This does not remap `idref` attributes to new ID values because it is impossible to determine which entity the `idref` was pointing to.

Parameters

- **`root`** – The XML document. This can be a filename, a file-like object, an instance of `etree._Element` or an instance of `etree._ElementTree`.

- **options** (*optional*) – A *ramrod.UpdateOptions* instance. If None, ramrod.DEFAULT_UPDATE_OPTIONS will be used.

Returns An instance of *ramrod.UpdateResults*.

get_version(*package*)
Returns the version of the *package* STIX_Package element by inspecting its version attribute.

update(*root*, *options=None*, *force=False*)
Attempts to update *root* to the next version of its language specification.

If *force* is set to True, items may be removed during the translation process and IDs may be reassigned if they are not unique within the document.

Note: This does not remap idref attributes to new ID values because it is impossible to determine which entity the idref was pointing to.

Removed items can be retrieved via the removed attribute on the return value:

```
>>> results = updater.update(root, force=True)
>>> print(results.removed)
(<Element at 0xffdcf234>, <Element at 0xffdcf284>)
```

Items which have been reassigned IDs can be retrieved via the remappped_ids attribute on the return value:

```
>>> results = updater.update(root, force=True)
>>> print(results.remapped_ids)
{'example:Observable-duplicate-id-1': [<Element {http://cybox.mitre...
```

Parameters

- **root** – The XML document. This can be a filename, a file-like object, an instance of etree._Element or an instance of etree._ElementTree.
- **options** – A *ramrod.UpdateOptions* instance. If None, ramrod.DEFAULT_UPDATE_OPTIONS will be used.
- **force** – Forces the update process to complete by potentially removing untranslatable xml nodes and/or remapping non-unique IDs. This may result in non-schema=conformant XML. **USE AT YOUR OWN RISK!**

Returns An instance of ramrod.UpdateResults.

Raises

- *UpdateError* – If untranslatable fields or non-unique IDs are discovered in *root* and *force* is False.
- *UnknownVersionError* – If the *root* node contains no version information.
- *InvalidVersionError* – If the *root* node contains invalid version information (e.g., the class expects v1.0 content and the *root* node contains v1.1 content).

Version: 1.2.0

2.1.9 ramrod.stix.stix_1_1 Module

class ramrod.stix.stix_1_1.**STIX_1_1_Updater**
Updates STIX v1.1 content to STIX v1.1.1.

The following update operations are performed:

- The `Source` field under instances of `StatementType` and `ConfidenceType` are translated from `ControlledVocabularyStringType` instances to `IdentityType` instances. The original value becomes the `Name` field of the `IdentityType` instance.
- The `Source` field under `IndicatorType/Sightings/Sighting` is converted into an instance of `IdentityType` where the original value becomes the value of the `Name` field.
- The `Related_Campaigns` field under `IndicatorType` is converted from a flat list of `RelatedCampaignType` instances into an instance of `GenericRelationshipListType`.
- Instances of the `GeneralTestMechanismType` extension have empty, optional fields removed.
- Instances of `AvailabilityLossVocab-1.0` are upgraded to `AvailabilityLossVocab-1.1`

Note: All fields can be translated between STIX v1.1 and STIX v1.1.1.

check_update(*root*, *options=None*)

Determines if the input document can be upgraded.

Parameters

- **root** – The XML document. This can be a filename, a file-like object, an instance of `etree._Element` or an instance of `etree._ElementTree`.
- **options** (*optional*) – A `ramrod.UpdateOptions` instance. If `None`, `ramrod.DEFAULT_UPDATE_OPTIONS` will be used.

Raises

- *UnknownVersionError* – If the input document does not have a version.
- *InvalidVersionError* – If the version of the input document does not match the *VERSION* class-level attribute value.
- *UpdateError* – If the input document contains fields which cannot be updated or constructs with non-unique IDs are discovered.

clean(*root*, *options=None*)

Removes disallowed elements from *root* and remaps non-unique IDs to unique IDs for the sake of schema-validation.

Removed items can be retrieved via the `removed` attribute on the return value:

```
>>> results = updater.clean(root)
>>> print(results.removed)
(<Element at 0xffdcf234>, <Element at 0xffdcf284>)
```

Items which have been reassigned IDs can be retrieved via the `remapped_ids` attribute on the return value:

```
>>> results = updater.clean(root)
>>> print(results.remapped_ids)
{'example:Observable-duplicate': [<Element {http://cybox.mitre.org...
```

Note: This does not remap `idref` attributes to new ID values because it is impossible to determine which entity the `idref` was pointing to.

Parameters

- **root** – The XML document. This can be a filename, a file-like object, an instance of `etree._Element` or an instance of `etree._ElementTree`.
- **options** (*optional*) – A *`ramrod.UpdateOptions`* instance. If `None`, `ramrod.DEFAULT_UPDATE_OPTIONS` will be used.

Returns An instance of *`ramrod.UpdateResults`*.

get_version(*package*)
Returns the version of the *package* `STIX_Package` element by inspecting its `version` attribute.

update(*root*, *options=None*, *force=False*)
Attempts to update *root* to the next version of its language specification.

If *force* is set to True, items may be removed during the translation process and IDs may be reassigned if they are not unique within the document.

Note: This does not remap `idref` attributes to new ID values because it is impossible to determine which entity the `idref` was pointing to.

Removed items can be retrieved via the `removed` attribute on the return value:

```
>>> results = updater.update(root, force=True)
>>> print(results.removed)
(<Element at 0xffdcf234>, <Element at 0xffdcf284>)
```

Items which have been reassigned IDs can be retrieved via the `remappped_ids` attribute on the return value:

```
>>> results = updater.update(root, force=True)
>>> print(results.remapped_ids)
{'example:Observable-duplicate-id-1': [<Element {http://cybox.mitre...
```

Parameters

- **root** – The XML document. This can be a filename, a file-like object, an instance of `etree._Element` or an instance of `etree._ElementTree`.
- **options** – A *`ramrod.UpdateOptions`* instance. If `None`, `ramrod.DEFAULT_UPDATE_OPTIONS` will be used.
- **force** – Forces the update process to complete by potentially removing untranslatable xml nodes and/or remapping non-unique IDs. This may result in non-schema=conformant XML. **USE AT YOUR OWN RISK!**

Returns An instance of `ramrod.UpdateResults`.

Raises

- *`UpdateError`* – If untranslatable fields or non-unique IDs are discovered in *root* and *force* is `False`.
- *`UnknownVersionError`* – If the *root* node contains no version information.
- *`InvalidVersionError`* – If the *root* node contains invalid version information (e.g., the class expects v1.0 content and the *root* node contains v1.1 content).

Version: 1.2.0

2.1.10 `ramrod.stix.stix_1_1_1` Module

class `ramrod.stix.stix_1_1_1.`**`STIX_1_1_1_Updater`**

Updates STIX v1.1.1 content to STIX v1.2.

The following update operations are performed:

- Instances of `DiscoveryMethodTypeVocab-1.0` are upgraded to `DiscoveryMethodTypeVocab-2.0`.
- Component versions are updated.
- Schemalocations are updated.

`check_update`(*root*, *options=None*)

Determines if the input document can be upgraded.

Parameters

- **`root`** – The XML document. This can be a filename, a file-like object, an instance of `etree._Element` or an instance of `etree._ElementTree`.
- **`options`** (*optional*) – A `ramrod.UpdateOptions` instance. If `None`, `ramrod.DEFAULT_UPDATE_OPTIONS` will be used.

Raises

- *`UnknownVersionError`* – If the input document does not have a version.
- *`InvalidVersionError`* – If the version of the input document does not match the *VERSION* class-level attribute value.

`get_version`(*package*)

Returns the version of the *package* `STIX_Package` element by inspecting its `version` attribute.

`update`(*root*, *options=None*, *force=False*)

Attempts to update *root* to the next version of its language specification.

If *force* is set to True, items may be removed during the translation process and IDs may be reassigned if they are not unique within the document.

Note: This does not remap `idref` attributes to new ID values because it is impossible to determine which entity the `idref` was pointing to.

Removed items can be retrieved via the `removed` attribute on the return value:

```
>>> results = updater.update(root, force=True)
>>> print(results.removed)
(<Element at 0xffdcf234>, <Element at 0xffdcf284>)
```

Items which have been reassigned IDs can be retrieved via the `remappped_ids` attribute on the return value:

```
>>> results = updater.update(root, force=True)
>>> print(results.remapped_ids)
{'example:Observable-duplicate-id-1': [<Element {http://cybox.mitre...
```

Parameters

- **`root`** – The XML document. This can be a filename, a file-like object, an instance of `etree._Element` or an instance of `etree._ElementTree`.

- **options** – A *ramrod.UpdateOptions* instance. If None, ramrod.DEFAULT_UPDATE_OPTIONS will be used.
- **force** – Forces the update process to complete by potentially removing untranslatable xml nodes and/or remapping non-unique IDs. This may result in non-schema=conformant XML. **USE AT YOUR OWN RISK!**

Returns An instance of `ramrod.UpdateResults`.

Raises

- *UpdateError* – If untranslatable fields or non-unique IDs are discovered in *root* and *force* is False.
- *UnknownVersionError* – If the *root* node contains no version information.
- *InvalidVersionError* – If the *root* node contains invalid version information (e.g., the class expects v1.0 content and the *root* node contains v1.1 content).

Version: 1.2.0

2.1.11 `ramrod.utils` Module

ramrod.utils.**children**(*node*)
Returns an iterable collection of etree Element nodes that are direct children of *node*.

ramrod.utils.**copy_xml_element**(*node*, *tag=None*)
Returns a copy of *node*. The copied node will be renamed to *tag* if *tag* is not None.

ramrod.utils.**create_new_id**(*orig_id*)
Creates a new ID from *orig_id* by appending '-cleaned-' and a UUID4 string to the end of the *orig_id* value.

Returns An ID string.

ramrod.utils.**descendants**(*node*)
Returns a list of etree Element nodes that are descendants of *node*.

ramrod.utils.**get_etree_root**(*doc*, *make_copy=False*)
Returns an instance of lxml.etree._Element for the given input.

Parameters

- **doc** – The input XML document. Can be an instance of `lxml.etree._Element`, `lxml.etree._ElementTree`, a file-like object, or a string filename.
- **make_copy** – If True, a `copy.deepcopy()` of the root node will be returned.

Returns An `lxml.etree._Element` instance for *doc*.

ramrod.utils.**get_ext_namespace**(*node*)
Returns the namespace which contains the type definition for the *node*. The type definition is specified by the `xsi:type` attribute which is formatted as `[alias]:[type name]`.

This method splits the `xsi:type` attribute value into an alias and a type name and performs a namespace lookup for the alias.

Parameters **node** – An instance of lxml.etree._Element which contains an `xsi:type` attribute.

Returns The namespace for the type defintion of this node.

Raises KeyError – if the node does not contain an `xsi:type` attribute.

ramrod.utils.**get_localname**(*node*)
Returns the localname portion the *node* QName

`ramrod.utils.`**`get_namespace`**`(`*node*`)`
Returns the namespace portion of the QName for *node*.

`ramrod.utils.`**`get_node_text`**`(`*node*`)`
Returns the text for a etree _Element *node*. If the node contains CDATA information, the text will be wrapped in an `etree.CDATA` instance.

Returns If the *node* contains a `<![CDATA[]]>` block, a `etree.CDATA` instance will be returned. If the node contains children, `None`. The *node* `text` value otherwise.

`ramrod.utils.`**`get_schemaloc_pairs`**`(`*node*`)`
Parses the xsi:schemaLocation attribute on *node*.

Returns A list of (ns, schemaLocation) tuples for the node.

Raises `KeyError` – If *node* does not have an xsi:schemaLocation attribute.

`ramrod.utils.`**`get_type_info`**`(`*node*`)`
Returns a (ns alias, typename) tuple which is generated from the `xsi:type` attribute on *node*.

Raises

- `KeyError` – If *node* does not contain an `xsi:type` attribute.
- `ValueError` – If the `xsi:type` attribute does not have a colon in it.

`ramrod.utils.`**`get_typed_nodes`**`(`*root*`)`
Finds all nodes under *root* which have an `xsi:type` attribute.

Returns A list of `etree._Element` instances.

`ramrod.utils.`**`get_xml_parser`**`()`
Returns an `etree.ETCompatXMLParser` instance.

`ramrod.utils.`**`ignored`**`(`**args*, ***kwds*`)`
Allows you to ignore exceptions cleanly using context managers. This exists in Python 3.

`ramrod.utils.`**`is_version_equal`**`(`*x*, *y*`)`
Attempts to determine if the *x* amd *y* version numbers are semantically equivalent.

Examples

The version strings “2.1.0” and “2.1” represent semantically equivalent versions, despite not being equal strings.

Parameters

- **`x`** – A string version number. Ex: ‘2.1.0’
- **`y`** – A string version number. Ex: ‘2.1’

`ramrod.utils.`**`iterchildren`**`(`*node*`)`
Returns an iterator which yields direct child elements of *node*.

`ramrod.utils.`**`iterdescendants`**`(`*node*`)`
Returns an iterator which yields descendant elements of *node*.

`ramrod.utils.`**`new_id`**`(`*node*`)`
Assigns a new, unique ID to *node* by appending ‘-cleaned-‘ and a UUID4 string to the end of the `id` attribute value of the node.

Example

```
>>> e = etree.Element('test')
>>> e.attrib['id'] = 'example:non-unique-id'
>>> e.attrib['id']
'example:non-unique-id'
>>> e = new_id(e)
>>> e.attrib['id']
'example:non-unique-id-cleaned-92eaa185-48e2-433e-82ba-a58f692bac32'
```

`ramrod.utils.`**`remove_xml_attribute`**(*node*, *attr*)
Removes an attribute from *node*.

Parameters

- **`node`** (*lxml.etree._Element*) – An _Element node.
- **`attr`** – A attribute tag to be removed.

`ramrod.utils.`**`remove_xml_attributes`**(*node*, *attrs*)
Removes xml attributes *attrs* from *node*.

`ramrod.utils.`**`remove_xml_element`**(*node*)
Removes *node* from the parent of *node*.

`ramrod.utils.`**`remove_xml_elements`**(*nodes*)
Removes each node found in *nodes* from the XML document.

`ramrod.utils.`**`replace_xml_element`**(*old*, *new*)
Replaces *old* node with *new* in the document which *old* exists.

`ramrod.utils.`**`strip_whitespace`**(*string*)
Returns a copy of *string* with its whitespace stripped.

`ramrod.utils.`**`validate_version`**(*version*, *allowed*)
Raises a `InvalidVersionError` if *version* is not found in *allowed*.

Parameters

- **`version`** – A version string.
- **`allowed`** – An iterable collection of version strings.

`ramrod.utils.`**`validate_versions`**(*from_*, *to_*, *allowed*)
Raises a `InvalidVersionError` if *from_* or *to_* are not found in *allowed* or *from_* is greater than or equal to *to_*.

Parameters

- **`from`** – A version string.
- **`to`** – A version string.
- **`allowed`** – An iterable collection of version strings.

Version: 1.2.0

2.2 Example Code

The following sections demonstrate how to use the **stix-ramrod** library to update STIX content. For more details about the **stix-ramrod** API, see the API Documentation page.

2.2.1 Import stix-ramrod

To use **stix-ramrod** for updating STIX and CybOX content, you must import the `ramrod` module There are lots of functions, classes, and submodules under `ramrod`, but the top-level module is all you need for most updates!

```
import ramrod  # That's it!
```

2.2.2 Calling the ramrod.update() Function

Once the imports are taken care of you only need to call the `ramrod.update()` method, which parses the content, updates it, and returns an instance of `UpdateResults`.

```
import ramrod

# Update the 'stix-content.xml' STIX document.
updated = ramrod.update('stix-content.xml')
```

Note: The example above passes the `stix-content.xml` filename into `ramrod.update()`, but `ramrod.update()` accepts file-like objects (such as files on disk or `StringIO` instances), `etree._Element` instances, or `etree._ElementTree` instances. Neato!

2.2.3 Retrieving Updated Content

After successfully calling `ramrod.update()`, the update document can be retrieved from the returned `UpdateResults` object instance via the `document` attribute. The `document` attribute is an instance of `ResultDocument`.

```
import ramrod
from lxml import etree  # Used for printing the updated XML document

# Update the document
updated = ramrod.update('stix-content.xml')

# Print the resulting document to stdout
print updated

# Retrieve the updated document from the returned UpdateResults object
new_stix_doc = updated.document

# Or retrieve the etree._Element root
root = new_stix_doc.as_element()
```

2.2.4 Forcing An Update

Sometimes an update doesn't go smoothly and a `UpdateError` is raised because untranslatable data or non-unique IDs are discovered in the source document. The following code and output demonstrates how to force the update and retrieve the data that is lost in the process.

```
import ramrod

# Attempt to update an untranslatable document
updated = ramrod.update('untranslatable-stix-content.xml')
```

The `untranslatable-stix-content.xml` contains untranslatable data, so a `UpdateError` gets raised:

```
ramrod.errors.UpdateError: Update Error: Found untranslatable fields in source document.
```

To find out *exactly* what couldn't be translated, you can inspect the `disallowed` and `duplicates` attributes on the `UpdateError` instance:

```
import ramrod
import ramrod.errors  # stix-ramrod error module

try:
    # Attempt to update an untranslatable document
    updated = ramrod.update('untranslatable-stix-content.xml')
except ramrod.errors.UpdateError as ex:
    # Print untranslatable items
    for node in ex.disallowed:
        print "TAG: %s, LINE: %s" % (node.tag, node.sourceline)  # etree API

    # Print non-unique IDs and each line they're found on
    for id_, nodes in ex.duplicates.iteritems():
        print "ID: %s, LINES: %s" % (id_, [x.sourceline for x in nodes])
```

To force the update, pass in `force=True` to the `ramrod.update()` method:

```
import ramrod

# Force-update the document
updated = ramrod.update('untranslatable-stix-content.xml', force=True)
```

After successfully force-updating the document, items that had IDs remapped or that were lost in translation can be retrieved from the returned `UpdateResults` object instance.

```
import ramrod

# Force-update the document
updated = ramrod.update('untranslatable-stix-content.xml', force=True)

# Iterate over the items which were lost in translation
for node in updated.removed:
    do_something_with_the_removed_item(node)

# Iterate over the {id: [nodes]} dictionary containing nodes
# with remapped IDs
for original_id, node_list in updated.remapped_ids.iteritems():
    do_something_with_remapped_items(original_id, node_list)
```

2.2.5 Using the UpdateOptions Class

Instances of the `UpdateOptions` class can be passed into the `ramrod.update()` method to tweak what gets updated in a STIX or CybOX document.

The following example shows how to use the `UpdateOptions` class to let the update code know **not** to update controlled vocabulary instances:

```
import ramrod
from lxml import etree  # used for parsing XML

# Create the UpdateOptions instance
```

```
options = ramrod.UpdateOptions()
options.update_vocabularies = False  # Don't Update Vocabs!

# Update the content
updated = ramrod.update('stix-content.xml', options=options)

# Print the results!
print updated
```

2.2.6 Working with python-stix

The python-stix library provides an API for developing and consuming STIX content. The python-stix library is designed to consume and produce specific versions of STIX, as detailed here.

Because python-stix consumes specific versions of STIX content, older content needs to be updated before it can be parsed. Luckily, updating old versions of STIX content is easy with **stix-ramrod**!

Example

The following example demonstrates one way of updating content so that python-stix can parse it. This code works with python-stix v1.1.1.1.

```
import ramrod
from stix.core import STIXPackage
from stix.utils.parser import UnsupportedVersionError

stix_filename = "stix-upgradable-content.xml"

try:
    package = STIXPackage.from_xml(stix_filename)
except UnsupportedVersionError as ex:
    updated  = ramrod.update(stix_filename)
    document = updated.document.as_stringio()
    package  = STIXPackage.from_xml(document)

# Work with the parsed STIXPackage instance.
print package.id_
```

Note: The example above assumes that the input content can be upgraded without raising a `UpdateError` or any other exceptions.

CHAPTER 3

Contributing

If a bug is found, a feature is missing, or something just isn't behaving the way you'd expect it to, please submit an issue to our tracker. If you'd like to contribute code to our repository, you can do so by issuing a pull request and we will work with you to try and integrate that code into our repository.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index