STIX XML Tutorial

Creating and understanding STIX XML

HS SEDI
Homeland Security Systems Engineering and Development Institute

Homeland Security

Authoring STIX XML

- **Step 1: Decide what you want to model before writing XML!**
  - Are you talking about an incident? An indicator? A campaign?
  - Make sure the data you're adding makes sense where you're adding it
    - An IP address in an Indicator is different than an IP address in an Incident
  - Think about: what do I want to tell my consumers?
  - Build a high-level model

Authoring STIX XML

- **Step 2: Understand what to represent and how to represent it**
  – Start with the major constructs (indicator, threat actor, etc.) and build down from there
  – Use the data model documentation and suggested practices to guide you

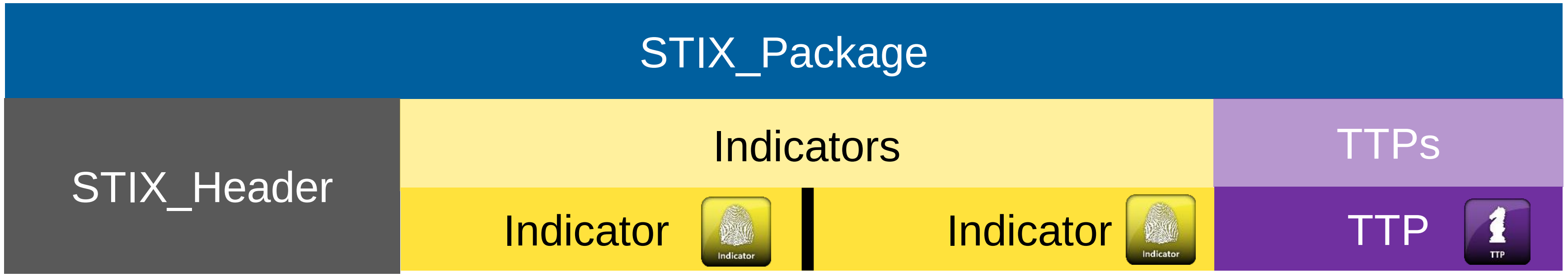
Tutorial Scenario

- A malware analysis team does an in-depth review of a piece of malware that was submitted. They identify:
  - That the malware is a variant of Poison Ivy
  - The file hash for the malware file itself
  - A set of two IP addresses that the malware attempted to connect to
- They decide to share this information in STIX. What would the resulting STIX document look like?
  - How would you generate this in your tool?

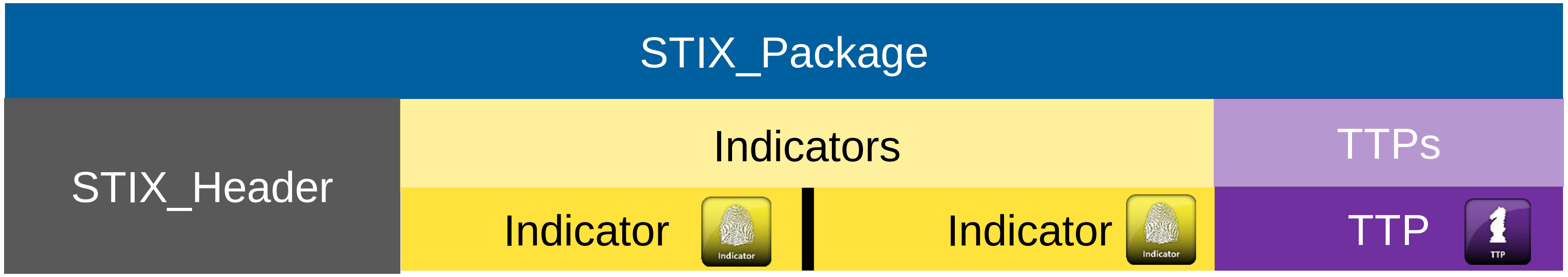
High-level Model

- **We want to describe….**
  - The malware itself by name and type
  - The file hash
  - The C2 IP addresses
  - Metadata describing the report (STIX_Header)
  - Relationships tying it all together

STIX Document Structure



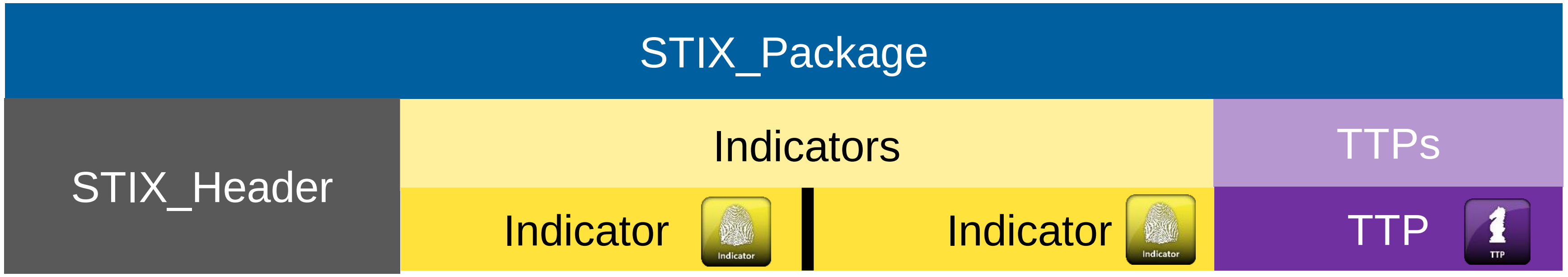
STIX Document Structure



STIX_Package

- XML stuff
- ID to uniquely identify package
- Timestamp indicating when package was published
- Version of STIX that is being conformed to

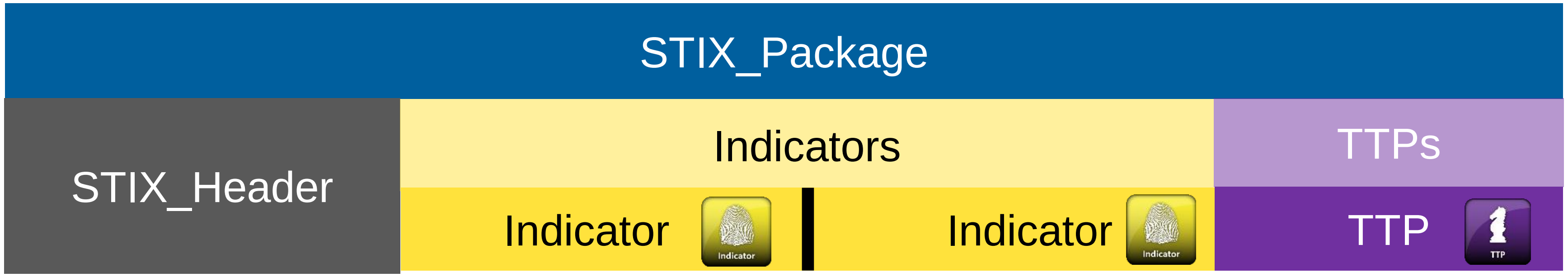
STIX Document Structure



STIX_Header

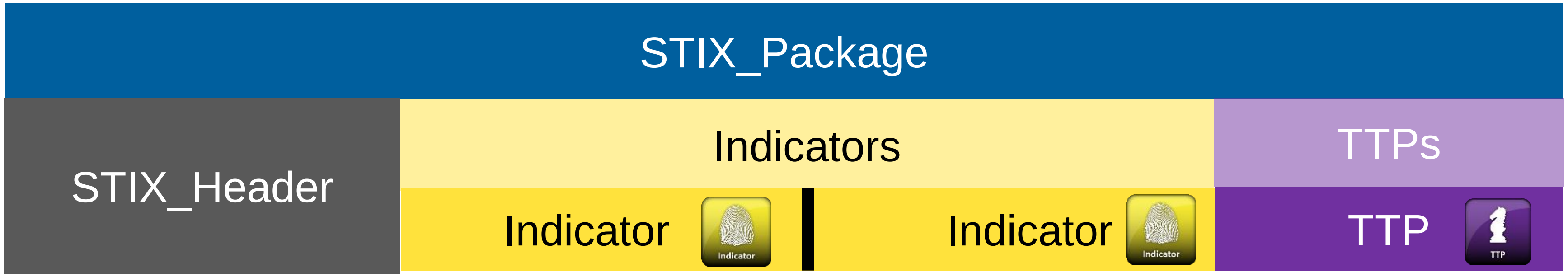
- Report title
- Package intent
- Information source

STIX Document Structure



- ID and timestamp
- Some metadata about the indicator
- Pattern for what to look for (file hash)
- Context if it’s seen (relationship to TTP)

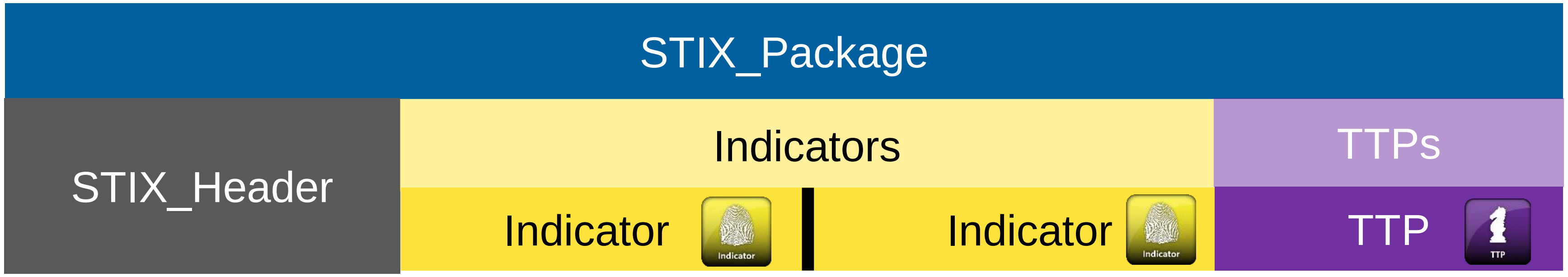
STIX Document Structure



Indicator

- ID and timestamp
- Some metadata about the indicator
- Pattern for what to look for (IP addresses)
- Context if it’s seen (relationship to TTP)

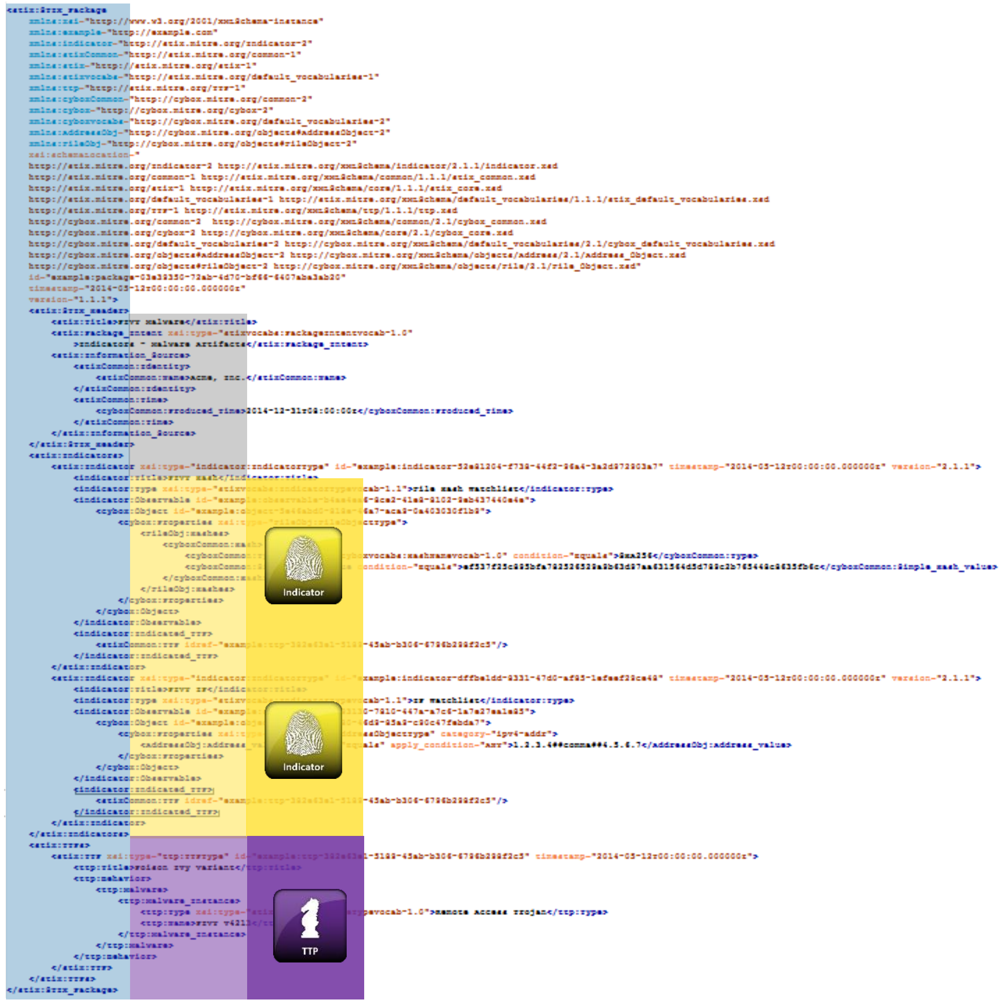
STIX Document Structure



- ID and Timestamp
- Malware Instance Information
  - Name (Poison Ivy)
  - Type (Remote Access)

STIX Document Structure (XML)

HS SEDI
Homeland Security Systems Engineering and Development Institute



```xml
<stix:STIX_Package
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:example="http://example.com"
    xmlns:indicator="http://stix.mitre.org/Indicator-2"
    xmlns:stixCommon="http://stix.mitre.org/common-1"
    xmlns:stix="http://stix.mitre.org/stix-1"
    xmlns:stixVocabs="http://stix.mitre.org/default_vocabularies-1"
    xmlns:ttp="http://stix.mitre.org/TTP-1"
    xmlns:cyboxCommon="http://cybox.mitre.org/common-2"
    xmlns:cybox="http://cybox.mitre.org/cybox-2"
    xmlns:cyboxVocabs="http://cybox.mitre.org/default_vocabularies-2"
    xmlns:AddressObj="http://cybox.mitre.org/objects#AddressObject-2"
    xmlns:FileObj="http://cybox.mitre.org/objects#FileObject-2"
    xsi:schemaLocation="
    http://stix.mitre.org/Indicator-2 http://stix.mitre.org/XMLSchema/indicator/2.1.1/indicator.xsd
    http://stix.mitre.org/common-1 http://stix.mitre.org/XMLSchema/common/1.1.1/stix_common.xsd
    http://stix.mitre.org/stix-1 http://stix.mitre.org/XMLSchema/core/1.1.1/stix_core.xsd
    http://stix.mitre.org/default_vocabularies-1 http://stix.mitre.org/XMLSchema/default_vocabularies/1.1.1/stix_default_vocabularies.xsd
    http://stix.mitre.org/TTP-1 http://stix.mitre.org/XMLSchema/ttp/1.1.1/ttp.xsd
    http://cybox.mitre.org/common-2  http://cybox.mitre.org/XMLSchema/common/2.1/cybox_common.xsd
    http://cybox.mitre.org/cybox-2 http://cybox.mitre.org/XMLSchema/core/2.1/cybox_core.xsd
    http://cybox.mitre.org/default_vocabularies-2 http://cybox.mitre.org/XMLSchema/default_vocabularies/2.1/cybox_default_vocabularies.xsd
    http://cybox.mitre.org/objects#AddressObject-2 http://cybox.mitre.org/XMLSchema/objects/Address/2.1/Address_Object.xsd
    http://cybox.mitre.org/objects#FileObject-2 http://cybox.mitre.org/XMLSchema/objects/File/2.1/File_Object.xsd"
    id="example:package-03a39350-72ab-4d70-bf66-6407aba3ab20"
    timestamp="2014-05-12T00:00:00.000000Z"
    version="1.1.1">
    <stix:STIX_Header>
        <stix:Title>PIVY Malware</stix:Title>
        <stix:Package_Intent xsi:type="stixVocabs:PackageIntentVocab-1.0"
            >Indicators - Malware Artifacts</stix:Package_Intent>
        <stix:Information_Source>
            <stixCommon:Identity>
                <stixCommon:Name>Acme, Inc.</stixCommon:Name>
            </stixCommon:Identity>
            <stixCommon:Time>
                <cyboxCommon:Produced_Time>2014-12-31T08:00:00Z</cyboxCommon:Produced_Time>
            </stixCommon:Time>
        </stix:Information_Source>
    </stix:STIX_Header>
    <stix:Indicators>
        <stix:Indicator xsi:type="indicator:IndicatorType" id="example:indicator-52e81204-f738-44f2-96a4-3a2d872903a7" timestamp="2014-05-12T00:00:00.000000Z" version="2.1.1">
            <indicator:Title>PIVY Hash</indicator:Title>
            <indicator:Type xsi:type="stixVocabs:IndicatorTypeVocab-1.1">File Hash Watchlist</indicator:Type>
            <indicator:Observable id="example:observable-b4ae4ea6-9ca2-41e8-9102-9eb437440e4e">
                <cybox:Object id="example:object-3e46abd0-919e-46a7-aca8-0a403030f1b9">
                    <cybox:Properties xsi:type="FileObj:FileObjectType">
                        <FileObj:Hashes>
                            <cyboxCommon:Hash>
                                <cyboxCommon:T[illegible]cyboxVocabs:HashNameVocab-1.0" condition="Equals">SHA256</cyboxCommon:Type>
                                <cyboxCommon:S[illegible] condition="Equals">ef537f25c895bfa782526529a9b63d97aa631564d5d789c2b765448c8635fb6c</cyboxCommon:Simple_Hash_Value>
                            </cyboxCommon:Hash[illegible]
                        </FileObj:Hashes>
                    </cybox:Properties>
                </cybox:Object>
            </indicator:Observable>
            <indicator:Indicated_TTP>
                <stixCommon:TTP idref="example:ttp-332e63e1-5188-45ab-b306-6796b298f2c5"/>
            </indicator:Indicated_TTP>
        </stix:Indicator>
        <stix:Indicator xsi:type="indicator:IndicatorType" id="example:indicator-dffbe1dd-8331-47d0-af85-1efeef29ce48" timestamp="2014-05-12T00:00:00.000000Z" version="2.1.1">
            <indicator:Title>PIVY IP</indicator:Title>
            <indicator:Type xsi:type="stixVocabs:IndicatorTypeVocab-1.1">IP Watchlist</indicator:Type>
            <indicator:Observable id="example[illegible]0-7810-447a-a7c6-1a7e27eale85">
                <cybox:Object id="example:obj[illegible]6d8-85a8-c80c47febda7">
                    <cybox:Properties xsi:type[illegible]AddressObjectType" category="ipv4-addr">
                        <AddressObj:Address_Va[illegible]Equals" apply_condition="ANY">1.2.3.4##comma##4.5.6.7</AddressObj:Address_Value>
                    </cybox:Properties>
                </cybox:Object>
            </indicator:Observable>
            <indicator:Indicated_TTP>
                <stixCommon:TTP idref="example:ttp-332e63e1-5188-45ab-b306-6796b298f2c5"/>
            </indicator:Indicated_TTP>
        </stix:Indicator>
    </stix:Indicators>
    <stix:TTPs>
        <stix:TTP xsi:type="ttp:TTPType" id="example:ttp-332e63e1-5188-45ab-b306-6796b298f2c5" timestamp="2014-05-12T00:00:00.000000Z">
            <ttp:Title>Poison Ivy Variant</ttp:Title>
            <ttp:Behavior>
                <ttp:Malware>
                    <ttp:Malware_Instance>
                        <ttp:Type xsi:type="stix[illegible]TypeVocab-1.0">Remote Access Trojan</ttp:Type>
                        <ttp:Name>PIVY v4213</t[illegible]
                    </ttp:Malware_Instance>
                </ttp:Malware>
            </ttp:Behavior>
        </stix:TTP>
    </stix:TTPs>
</stix:STIX_Package>
```

Homeland Security

Make some XML

- **By Hand**
  - Oxygen or XMLSpy
  - Eclipse
  - Code editor (Sublime Text, atom.io, etc.)

- **In Code**
  - Via bindings (stay tuned)
    - python-stix or python-java
  - Via native XML tooling (focus for now)

STIX_Package

STIX_Package

- XML stuff
- ID to uniquely identify package
- Timestamp indicating when package was published
- Version of STIX that is being conformed to

STIX_Package: XML Stuff

- **Namespace prefix declarations and schemaLocation information**
- **Mostly done for you if you use the template**
- **Otherwise, you'll need to add them as you go**
  - Note that schemaLocation is optional

Exception: Add ID namespace

HS SEDI
Homeland Security Systems Engineering and Development Institute

- **Per suggested practices, STIX IDs should be namespaced by the producer: [ns prefix]:[construct type]-[guid]**

- **Add the ns prefix and the namespace to the STIX_Package**

```
<stix:STIX_Package
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:example="http://example.com"
```

Short prefix (nickname)

Full namespace

Homeland Security

Set ID, timestamp, version

- Use the namespace prefix we just defined in the pattern: [ns prefix]:[construct type]-[guid]

- Timestamp is used for versioning and should be set to the time the document will be considered “published”

```
<stix:STIX_Package
       ……
       id="example:package-03e39350-72ab-4d70-bf66-6407aba3ab20"
       timestamp="2014-05-12T00:00:00.000000Z"
       version="1.1.1">

</stix:STIX_Package>
```

STIX_Header

HS SEDI
Homeland Security Systems Engineering and Development Institute

STIX_Header

- Package title
- Package intent
- Information source

Homeland Security

Quiz: What do I add?

- **What do I need to express?**
- **Look in the data model for inspiration or help**
- **Look at the suggested practices**

STIX_Header

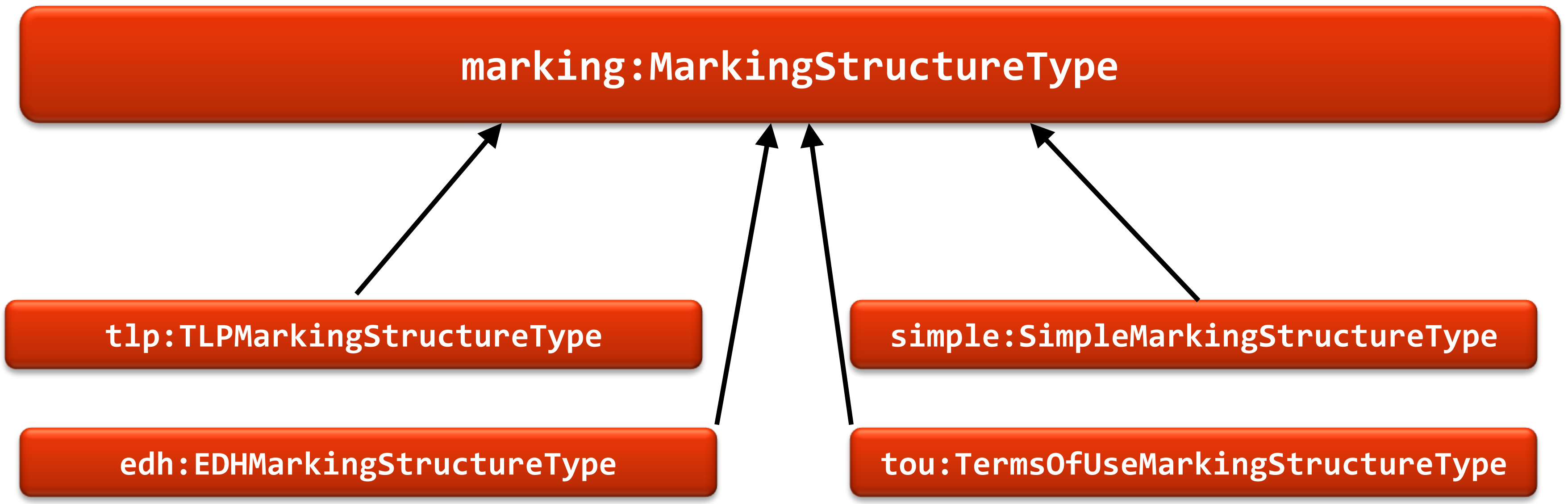
```
<stix:STIX_Header>




</stix:STIX_Header>
```

What's going on here?

Concept: xsi:type

```
xsi:type="stixVocabs:PackageIntentVocab-1.0"
```

- **xsi:type is a standard XML and XML schema mechanism for enabling type hierarchies**
  - I.e., inheritance for the OO programmers

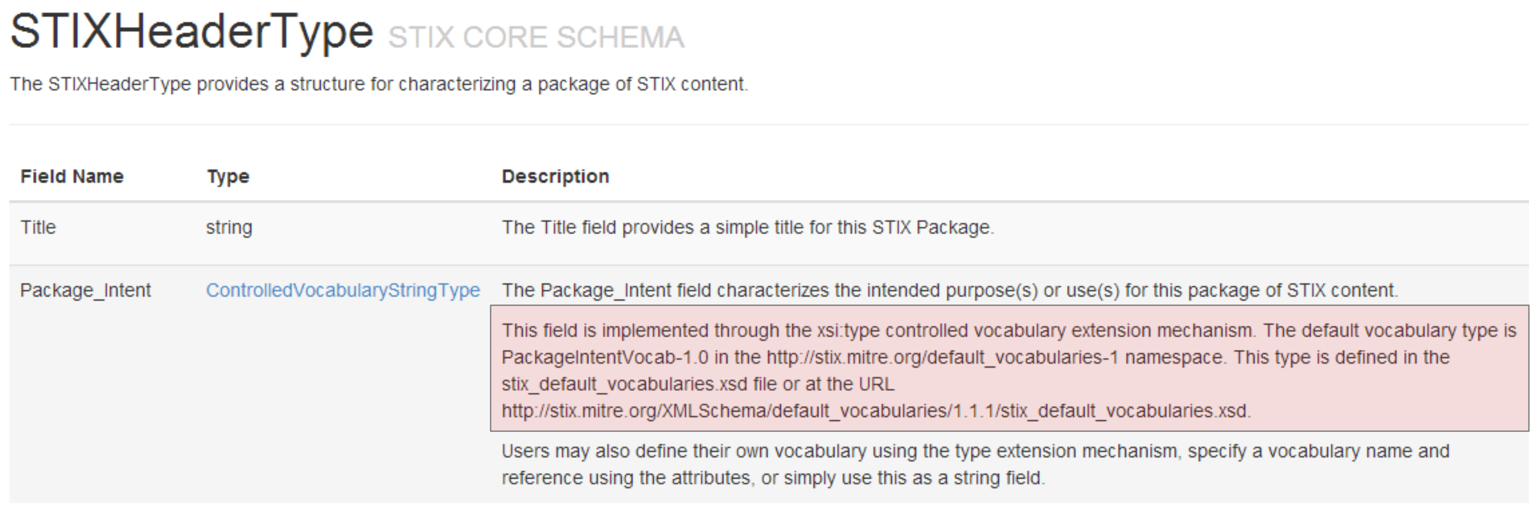

marking:MarkingStructureType
tlp:TLPMarkingStructureType
simple:SimpleMarkingStructureType
edh:EDHMarkingStructureType
tou:TermsOfUseMarkingStructureType

xsi:type in STIX

- **Core constructs use xsi:type to de-couple themselves from each other and from core**
- **Controlled vocabularies use xsi:type to allow for:**
  - STIX-defined default vocabularies
  - Free-form text
  - 3rd party-defined vocabularies
- **Extension points use xsi:type to avoid coupling to specific implementations when there's no strong community consensus towards a single option**
  - Test mechanisms, identity, data markings, etc.

Using xsi:type

- **The documentation will indicate that a field is an extension point**
  - In many cases it will also list the “default” implementation



STIXHeaderType STIX CORE SCHEMA

The STIXHeaderType provides a structure for characterizing a package of STIX content.

Field Name	Type	Description
Title	string	The Title field provides a simple title for this STIX Package.
Package_Intent	ControlledVocabularyStringType	The Package_Intent field characterizes the intended purpose(s) or use(s) for this package of STIX content. This field is implemented through the xsi:type controlled vocabulary extension mechanism. The default vocabulary type is PackageIntentVocab-1.0 in the http://stix.mitre.org/default_vocabularies-1 namespace. This type is defined in the stix_default_vocabularies.xsd file or at the URL http://stix.mitre.org/XMLSchema/default_vocabularies/1.1.1/stix_default_vocabularies.xsd. Users may also define their own vocabulary using the type extension mechanism, specify a vocabulary name and reference using the attributes, or simply use this as a string field.

Hash Indicator

Indicator

- ID and timestamp
- Some metadata about the indicator
- Pattern for what to look for (file hash)
- Context if it’s seen (relationship to TTP)

Hash Indicator: Set ID and timestamp

- **Nearly identical to the metadata on the package**

```
<stix:Indicator xsi:type="indicator:IndicatorType"
       id="example:indicator-52e81204-f738-44f2-96a4-3a2d972903a7"
       timestamp="2014-05-12T00:00:00.000000Z">
</stix:Indicator>
```

Hash Indicator: Set Metadata

```
<stix:Indicator xsi:type="indicator:IndicatorType"
       id="example:indicator-52e81204-f738-44f2-96a4-3a2d972903a7"
       timestamp="2014-05-12T00:00:00.000000Z">
       <indicator:Title>PIVY Hash</indicator:Title>
       <indicator:Type xsi:type="stixVocabs:IndicatorTypeVocab-
1.1">File Hash Watchlist</indicator:Type>
</stix:Indicator>
```

Hash Indicator: Add Pattern

- **The indicator patterns specifies what to look for**

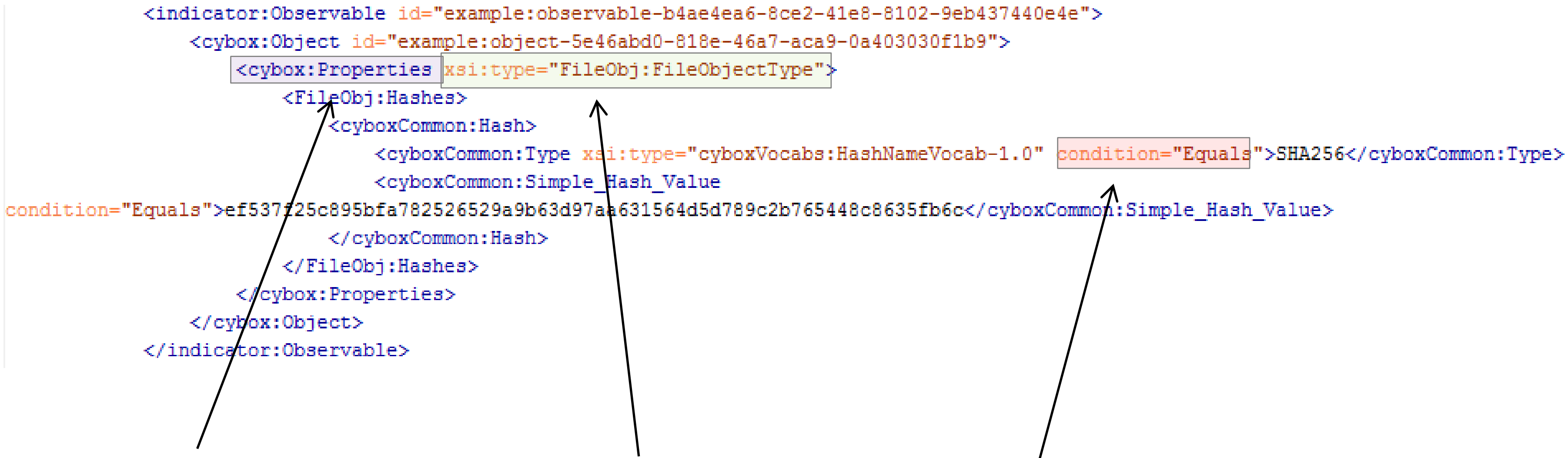
- **Consists of either a single test or a logical combination of tests against the CybOX object model**
  – Composition mechanisms can be used to create those logical combinations at various levels of abstraction

- **To create the pattern:**
  1. Determine the logical combination of tests
  2. Identify the CybOX objects and fields for each test
     - Use the cybox.mitre.org website and documentation

Hash Indicator: Add Pattern

- No combination of tests: just a single check for a hash
- A hash is an aspect of the file, so we’ll use the file object

Hash Indicator: Add Observable



```
        <indicator:Observable id="example:observable-b4ae4ea6-8ce2-41e8-8102-9eb437440e4e">
            <cybox:Object id="example:object-5e46abd0-818e-46a7-aca9-0a403030f1b9">
                <cybox:Properties xsi:type="FileObj:FileObjectType">
                    <FileObj:Hashes>
                        <cyboxCommon:Hash>
                            <cyboxCommon:Type xsi:type="cyboxVocabs:HashNameVocab-1.0" condition="Equals">SHA256</cyboxCommon:Type>
                            <cyboxCommon:Simple_Hash_Value
condition="Equals">ef537f25c895bfa782526529a9b63d97aa631564d5d789c2b765448c8635fb6c</cyboxCommon:Simple_Hash_Value>
                        </cyboxCommon:Hash>
                    </FileObj:Hashes>
                </cybox:Properties>
            </cybox:Object>
        </indicator:Observable>
```

`Properties` field holds CybOX

object properties

`xsi:type` indicates type of object

`condition` (required for patterns) indicates type of match

IP Indicator

Indicator

- ID and timestamp
- Some metadata about the indicator
- Pattern for what to look for (IP addresses)
- Context if it’s seen (relationship to TTP)

Quiz: What steps do I need to follow?

- **Set ID and timestamp**
- **Add the indicator metadata**
- **Determine whether we need a logical combination of tests**
- **Determine type of object(s)**
  - Determine the fields
  - Set conditions appropriately

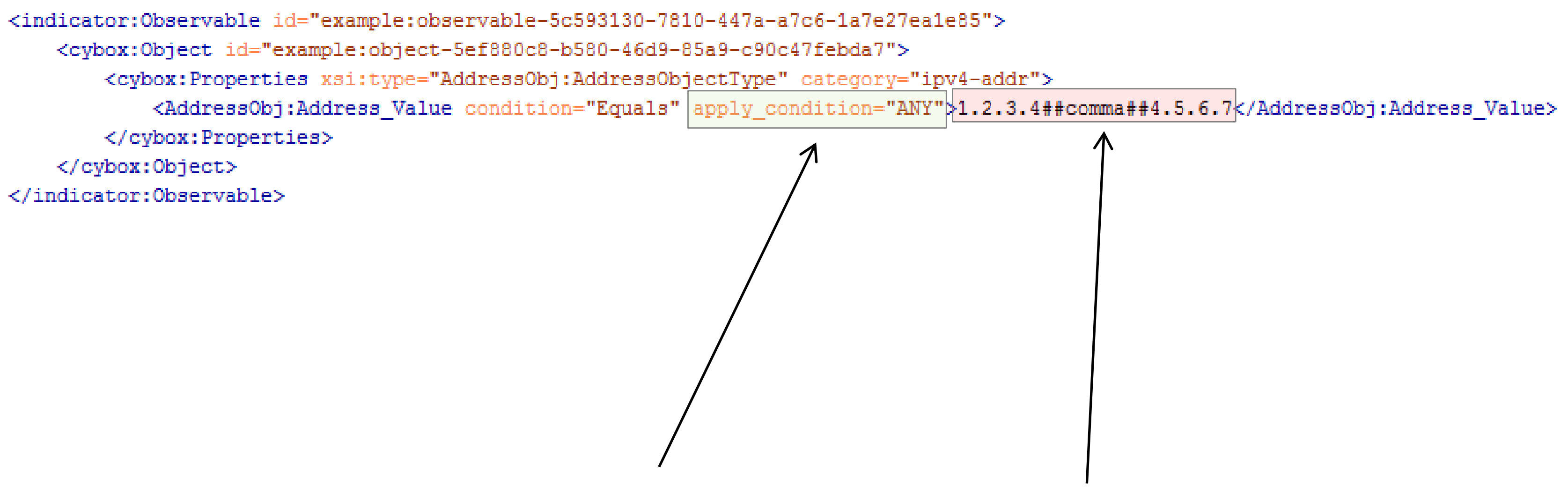
IP Indicator: The Basics

```
<stix:Indicator xsi:type="indicator:IndicatorType"



</stix:Indicator>
```

IP Indicator Pattern

- **Simple list of IP addresses**
  - Address Object
- **Means we have a logical combination: one IP OR another IP**
- **Check the documentation**
  - Cross-cutting feature idiom (future)
  - Field-level documentation for Observable_Composition, Composite_Indicator_Expression, and apply_condition

Adding the observable



```
<indicator:Observable id="example:observable-5c593130-7810-447a-a7c6-1a7e27ea1e85">
    <cybox:Object id="example:object-5ef880c8-b580-46d9-85a9-c90c47febda7">
        <cybox:Properties xsi:type="AddressObj:AddressObjectType" category="ipv4-addr">
            <AddressObj:Address_Value condition="Equals" apply_condition="ANY">1.2.3.4##comma##4.5.6.7</AddressObj:Address_Value>
        </cybox:Properties>
    </cybox:Object>
</indicator:Observable>
```

Indicates that the pattern should match if ANY of the list items “Equals” the tested IP

Separate list items with ##comma##

TTP

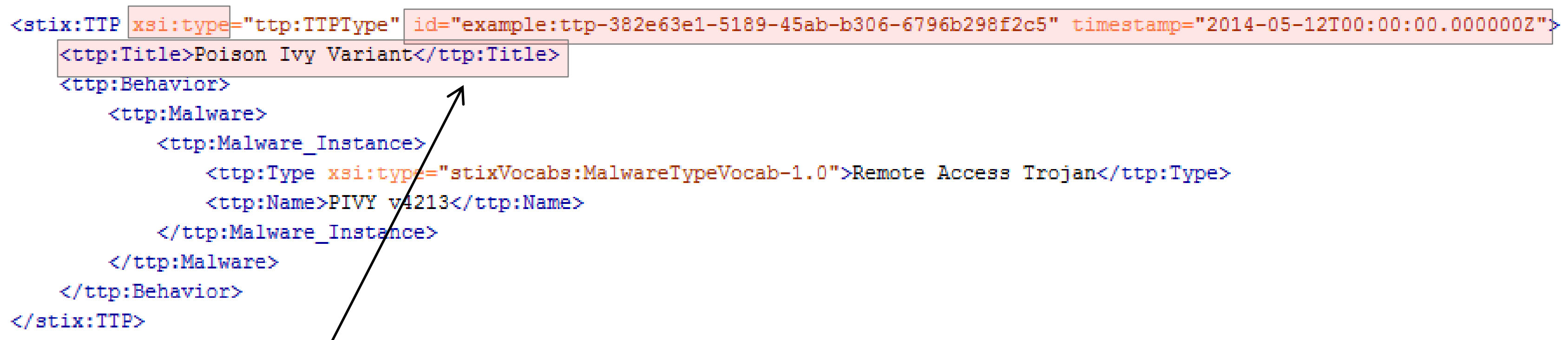
TTP

- ID and Timestamp
- Malware Instance Information
  - Name (Poison Ivy)
  - Type (Remote Access)

Quiz 2: What steps do I need to follow?

- **Set ID and timestamp**
- **Add fields from data model and suggested practices**
  - TTP Title
  - Malware Instance
    - Name
    - Type

Completed TTP

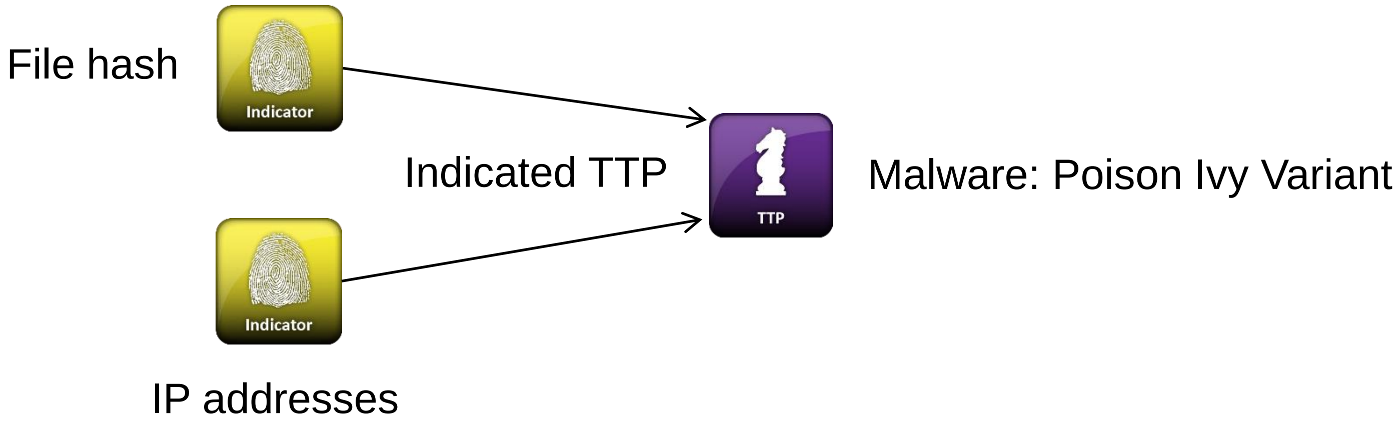


```
<stix:TTP xsi:type="ttp:TTPType" id="example:ttp-382e63e1-5189-45ab-b306-6796b298f2c5" timestamp="2014-05-12T00:00:00.000000Z">
    <ttp:Title>Poison Ivy Variant</ttp:Title>
    <ttp:Behavior>
        <ttp:Malware>
            <ttp:Malware_Instance>
                <ttp:Type xsi:type="stixVocabs:MalwareTypeVocab-1.0">Remote Access Trojan</ttp:Type>
                <ttp:Name>PIVY v4213</ttp:Name>
            </ttp:Malware_Instance>
        </ttp:Malware>
    </ttp:Behavior>
</stix:TTP>
```

Same as indicator. STIX follows consistent design patterns for many things, meaning lessons learned for Indicator can be re-applied to TTP

Linking it all together

- **Add the relationships as described in the original model**



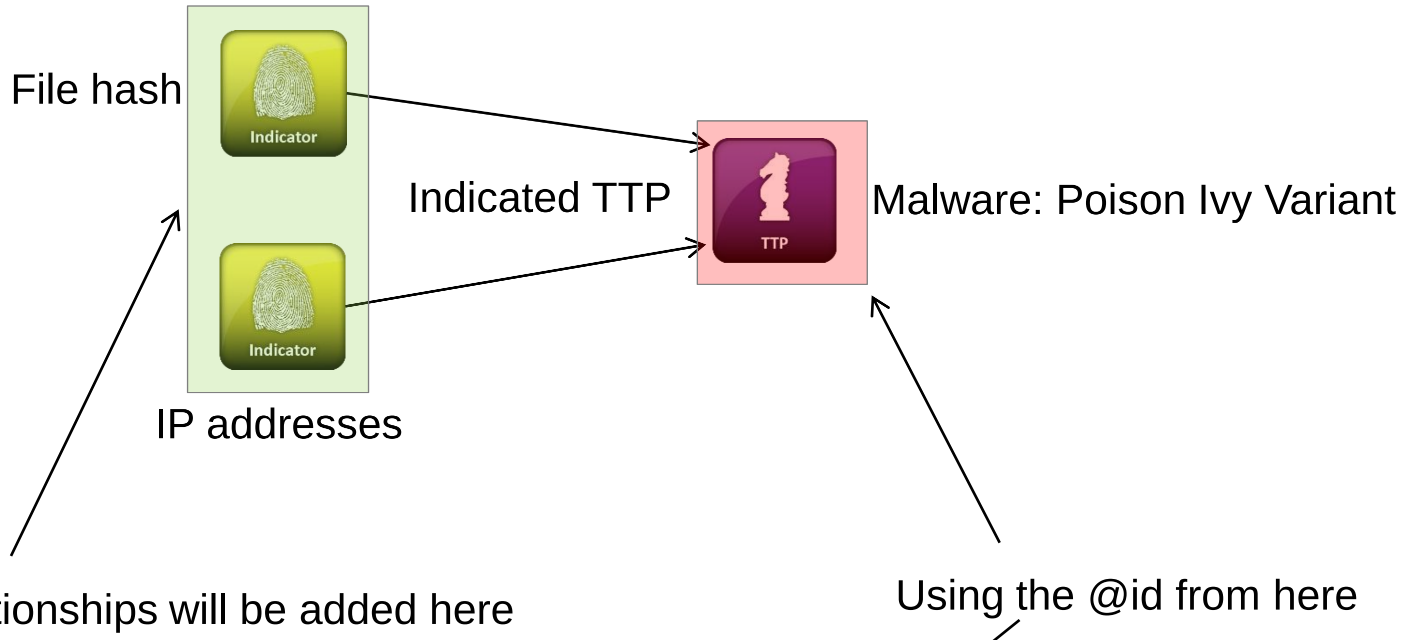
STIX Relationships

- **STIX relationships allow you to represent knowledge graphs**
  - Intelligence isn’t about disconnected facts, it’s about relationships between those facts
- **STIX relationships are all implemented identically**
  - See the cross-cutting feature idiom
- **Relationships can either be:**
  - Embedded: Relationship target is embedded in the source
  - Referenced: Relationship references the target defined elsewhere

Indicated_TTP Relationships

- **Give indicator context for what it means**
- **Without some context, all you have is a pattern with no idea what it means**
- **Relationship points FROM the indicators TO the TTPs**
- **Typically should reference the TTPs so that they can be used elsewhere**

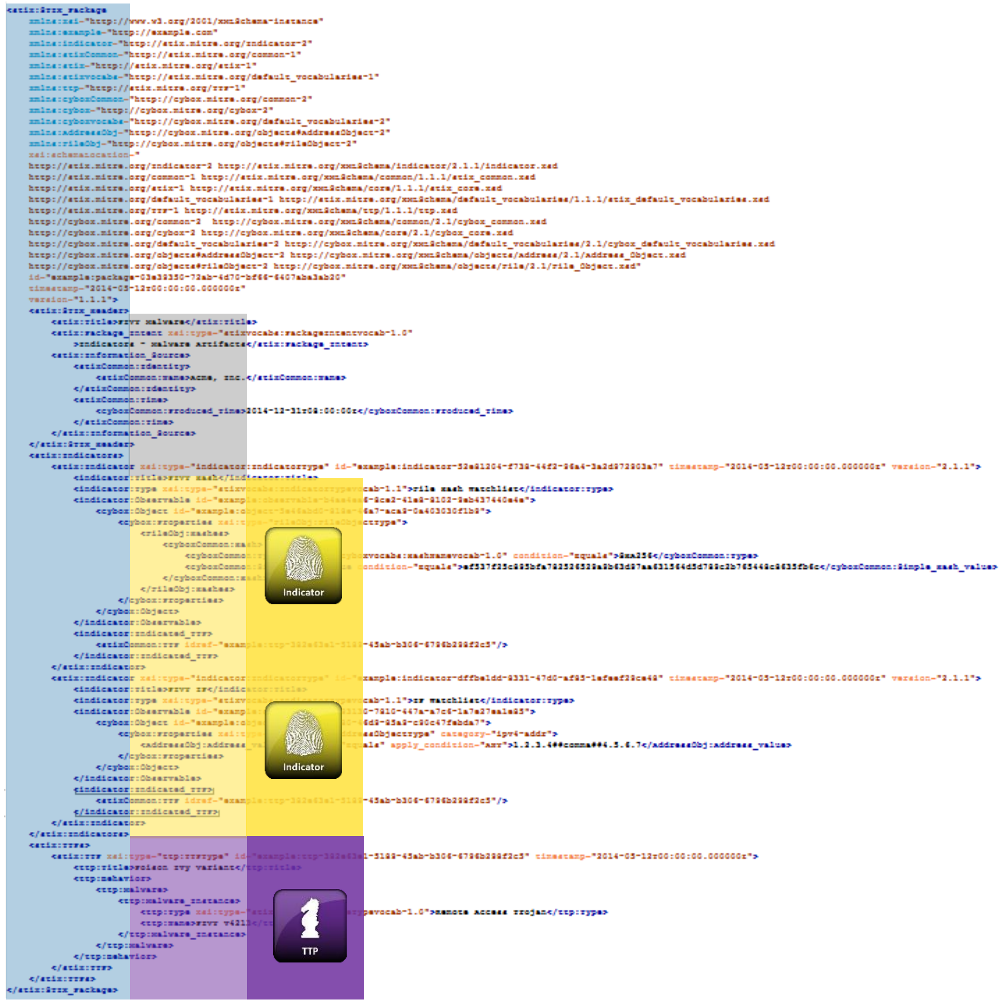
Adding a reference relationship



```
<stix:TTP xsi:type="ttp:TTPType" id="example:ttp-382e63e1-5189-45ab-b306-6796b298f2c5" timestam
```

```
<indicator:Indicated_TTP>
    <stixCommon:TTP idref="example:ttp-382e63e1-5189-45ab-b306-6796b298f2c5"/>
</indicator:Indicated_TTP>
```

Completed document



```
<stix:STIX_Package
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:example="http://example.com"
    xmlns:indicator="http://stix.mitre.org/Indicator-2"
    xmlns:stixCommon="http://stix.mitre.org/common-1"
    xmlns:stix="http://stix.mitre.org/stix-1"
    xmlns:stixVocabs="http://stix.mitre.org/default_vocabularies-1"
    xmlns:ttp="http://stix.mitre.org/TTP-1"
    xmlns:cyboxCommon="http://cybox.mitre.org/common-2"
    xmlns:cybox="http://cybox.mitre.org/cybox-2"
    xmlns:cyboxVocabs="http://cybox.mitre.org/default_vocabularies-2"
    xmlns:AddressObj="http://cybox.mitre.org/objects#AddressObject-2"
    xmlns:FileObj="http://cybox.mitre.org/objects#FileObject-2"
    xsi:schemaLocation="
    http://stix.mitre.org/Indicator-2 http://stix.mitre.org/XMLSchema/indicator/2.1.1/indicator.xsd
    http://stix.mitre.org/common-1 http://stix.mitre.org/XMLSchema/common/1.1.1/stix_common.xsd
    http://stix.mitre.org/stix-1 http://stix.mitre.org/XMLSchema/core/1.1.1/stix_core.xsd
    http://stix.mitre.org/default_vocabularies-1 http://stix.mitre.org/XMLSchema/default_vocabularies/1.1.1/stix_default_vocabularies.xsd
    http://stix.mitre.org/TTP-1 http://stix.mitre.org/XMLSchema/ttp/1.1.1/ttp.xsd
    http://cybox.mitre.org/common-2  http://cybox.mitre.org/XMLSchema/common/2.1/cybox_common.xsd
    http://cybox.mitre.org/cybox-2 http://cybox.mitre.org/XMLSchema/core/2.1/cybox_core.xsd
    http://cybox.mitre.org/default_vocabularies-2 http://cybox.mitre.org/XMLSchema/default_vocabularies/2.1/cybox_default_vocabularies.xsd
    http://cybox.mitre.org/objects#AddressObject-2 http://cybox.mitre.org/XMLSchema/objects/Address/2.1/Address_Object.xsd
    http://cybox.mitre.org/objects#FileObject-2 http://cybox.mitre.org/XMLSchema/objects/File/2.1/File_Object.xsd"
    id="example:package-03e39350-72ab-4d70-bf66-6407aba3ab20"
    timestamp="2014-05-12T00:00:00.000000Z"
    version="1.1.1">
    <stix:STIX_Header>
        <stix:Title>Poison Ivy Malware</stix:Title>
        <stix:Package_Intent xsi:type="stixVocabs:PackageIntentVocab-1.0"
            >Indicators - Malware Artifacts</stix:Package_Intent>
        <stix:Information_Source>
            <stixCommon:Identity>
                <stixCommon:Name>ACME, Inc.</stixCommon:Name>
            </stixCommon:Identity>
            <stixCommon:Time>
                <cyboxCommon:Produced_Time>2014-12-31T08:00:00Z</cyboxCommon:Produced_Time>
            </stixCommon:Time>
        </stix:Information_Source>
    </stix:STIX_Header>
    <stix:Indicators>
        <stix:Indicator xsi:type="indicator:IndicatorType" id="example:indicator-52e81204-f738-44f2-96a4-3a2d872903a7" timestamp="2014-05-12T00:00:00.000000Z" version="2.1.1">
            <indicator:Title>PIVY Hash</indicator:Title>
            <indicator:Type xsi:type="stixVocabs:IndicatorTypeVocab-1.1">File Hash Watchlist</indicator:Type>
            <indicator:Observable id="example:Observable-b4ae4ea6-8ce2-41e8-8102-8eb437440e4e">
                <cybox:Object id="example:Object-5e46abd0-819e-46a7-aca8-0a403030f1b9">
                    <cybox:Properties xsi:type="FileObj:FileObjectType">
                        <FileObj:Hashes>
                            <cyboxCommon:Hash>
                                <cyboxCommon:T[illegible]cyboxVocabs:HashNameVocab-1.0" condition="Equals">SHA256</cyboxCommon:Type>
                                <cyboxCommon:S[illegible] condition="Equals">ef537f25c895bfa782526529a9b63d97aa631564d5d789c2b765448c8635fb6c</cyboxCommon:Simple_Hash_Value>
                            </cyboxCommon:Hash>
                        </FileObj:Hashes>
                    </cybox:Properties>
                </cybox:Object>
            </indicator:Observable>
            <indicator:Indicated_TTP>
                <stixCommon:TTP idref="example:ttp-382e63e1-5199-45ab-b306-6796b299f2c5"/>
            </indicator:Indicated_TTP>
        </stix:Indicator>
        <stix:Indicator xsi:type="indicator:IndicatorType" id="example:indicator-dffbeldd-8331-47d0-af95-1efeef29ce48" timestamp="2014-05-12T00:00:00.000000Z" version="2.1.1">
            <indicator:Title>PIVY IP</indicator:Title>
            <indicator:Type xsi:type="stixVocabs[illegible]cab-1.1">IP Watchlist</indicator:Type>
            <indicator:Observable id="examp[illegible]0-7810-447a-a7c6-1a7e27ea1e85">
                <cybox:Object id="example:obje[illegible]6d8-85a8-c80c47febda7">
                    <cybox:Properties xsi:type[illegible]ressObjectType" category="ipv4-addr">
                        <AddressObj:Address_Va[illegible]uals" apply_condition="ANY">1.2.3.4##comma##4.5.6.7</AddressObj:Address_Value>
                    </cybox:Properties>
                </cybox:Object>
            </indicator:Observable>
            <indicator:Indicated_TTP>
                <stixCommon:TTP idref="example:ttp-382e63e1-5199-45ab-b306-6796b299f2c5"/>
            </indicator:Indicated_TTP>
        </stix:Indicator>
    </stix:Indicators>
    <stix:TTPs>
        <stix:TTP xsi:type="ttp:TTPType" id="example:ttp-382e63e1-5199-45ab-b306-6796b299f2c5" timestamp="2014-05-12T00:00:00.000000Z">
            <ttp:Title>Poison Ivy Variant</ttp:Title>
            <ttp:Behavior>
                <ttp:Malware>
                    <ttp:Malware_Instance>
                        <ttp:Type xsi:type="stix[illegible]TypeVocab-1.0">Remote Access Trojan</ttp:Type>
                        <ttp:Name>PIVY V4213</[illegible]
                    </ttp:Malware_Instance>
                </ttp:Malware>
            </ttp:Behavior>
        </stix:TTP>
    </stix:TTPs>
</stix:STIX_Package>
```

Postscript: Reading and Understanding STIX

- **Use STIXViz and STIX2HTML if at all possible**
- **Document as you go along**
- **Keep track of major constructs, titles, etc.**
- **Understand the overall purpose of what you're reading, don't just look at the details**
  - Always be thinking "what is the author saying with this document"

python-stix

Producing and consuming STIX via the Python APIs

Installing the APIs

- **Prerequisites**
  - Python 2.7 (not compatible with Python 3)
  - A modern version of libxml2 and lxml
    - Windows has an installer
    - Mac/Linux have packages

- **Installation (command line)**
  - pip install stix

Common Problems

- **Getting a weird error parsing or serializing documents:**

```
Fatal error occurred: local union type: A type, derived by list or union, must have the
simple ur-type definition as base type, not '{http://cybox.mitre.org/common-2}(NULL)'., line 350
```

  – Upgrade your version of libxml2 to at least v2.9.1

- **Old version (features that should be supported are not)**
  – pip install stix --upgrade
    - Make sure lxml is up-to-date at this time, in particular on Windows

- **Parsing incompatible STIX documents**
  – Currently, python-stix is designed to parse and create a specified version of STIX.
  – The first three numbers of the python-stix version define the supported version of STIX.

Consuming STIX in Python

Load the document

- **Often via file:**

```
from stix.core import STIXPackage

package = STIXPackage.from_xml('pivy-hash.xml')
```

- **Or a string:**

```
from stix.core import STIXPackage
from StringIO import StringIO

package=STIXPackage.from_xml(StringIO(stix_xml_string))
```

- **Or a TAXII Message body**

```
io = StringIO(taxii_message.content_blocks[0].content.to_xml())
package = STIXPackage.from_xml(io)
```

Load the document (alternative)

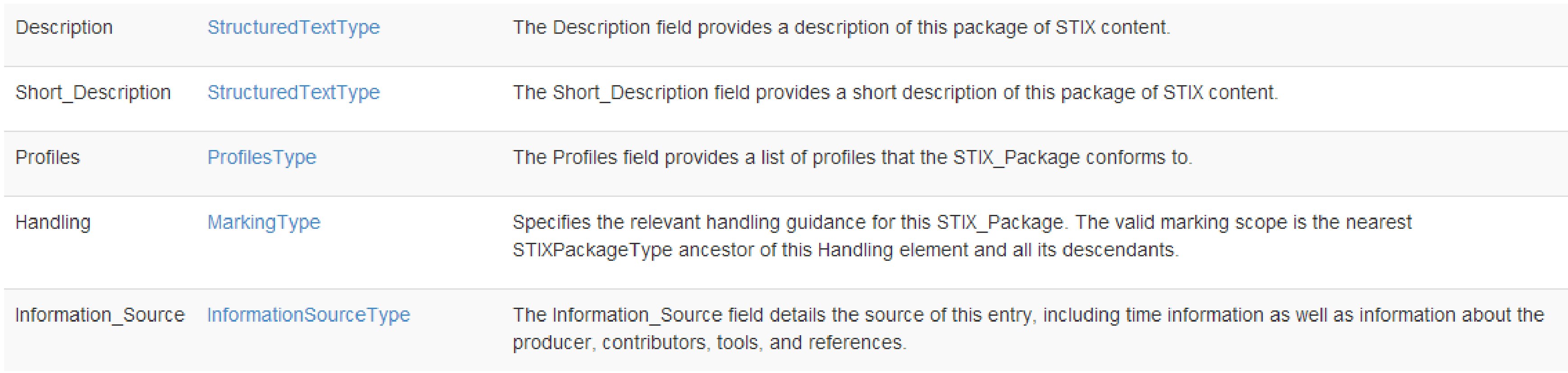
- **stix.utils.parser.EntityParser**
  - Used internally by STIXPackage.from_xml()
  - `def parse_xml()` parameters
    - `xml_file – The file-like object or filename to parse`
    - `check_version – Checks that the document version is supported by python-stix. Default is True.`
    - `check_root – Checks the root element of the xml instance document to verify that it is a STIX_Package element. Default is True`

```
from stix.utils.parser import EntityParser
stix_document = “stix.xml”
parser = EntityParser()
stix_package = parser.parse_xml(stix_document, check_version=False)
```

Data model vs. python-stix

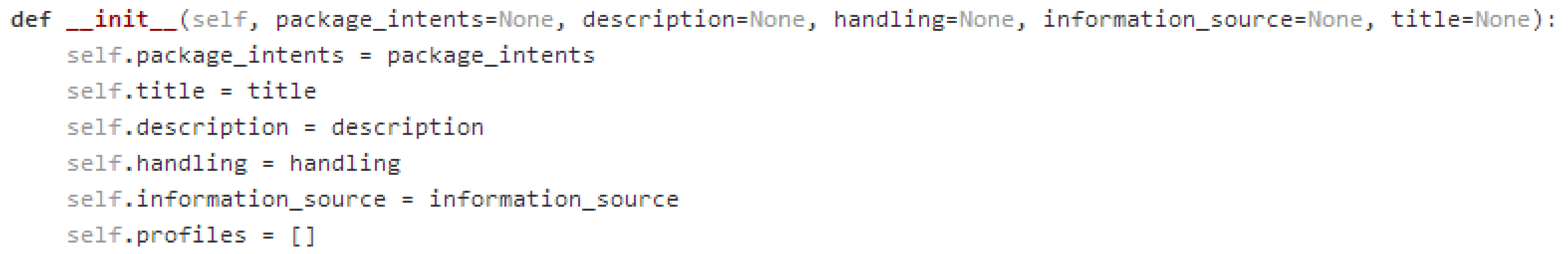
HS SEDI
Homeland Security Systems Engineering and Development Institute

- Use the data model to find field meanings and contents



Description	StructuredTextType	The Description field provides a description of this package of STIX content.
Short_Description	StructuredTextType	The Short_Description field provides a short description of this package of STIX content.
Profiles	ProfilesType	The Profiles field provides a list of profiles that the STIX_Package conforms to.
Handling	MarkingType	Specifies the relevant handling guidance for this STIX_Package. The valid marking scope is the nearest STIXPackageType ancestor of this Handling element and all its descendants.
Information_Source	InformationSourceType	The Information_Source field details the source of this entry, including time information as well as information about the producer, contributors, tools, and references.

- Use the API documentation (stix.readthedocs.org) to find out how to retrieve that in the API



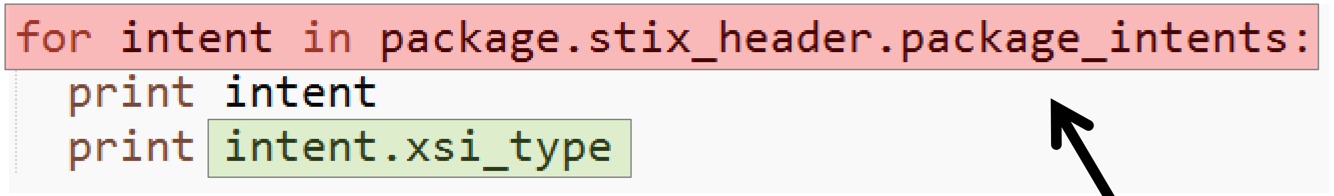
```
def __init__(self, package_intents=None, description=None, handling=None, information_source=None, title=None):
    self.package_intents = package_intents
    self.title = title
    self.description = description
    self.handling = handling
    self.information_source = information_source
    self.profiles = []
```

Homeland Security

Data model vs. python-stix (continued)

- **Python != XML Schema**
  - python-stix tries to be comfortable for Python developers working with STIX
  - Package layout is similar, but not the same to schema layout
- **Lists**
  - Multiple elements in XML (`maxOccurs="unbounded"`)
  - List objects in Python
- **Naming conventions**
  - XML vs. Python
  - Reserved words, naming conflicts, and PEP8
    - `id_`, `file_`, `object_`, `type_`, etc.

Accessing data

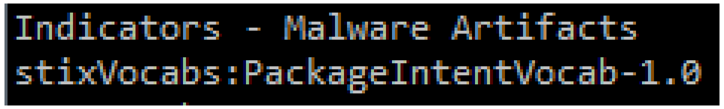


```
for intent in package.stix_header.package_intents:
    print intent
    print intent.xsi_type
```

Strategies like this can be used to get the vocabulary that is being used.

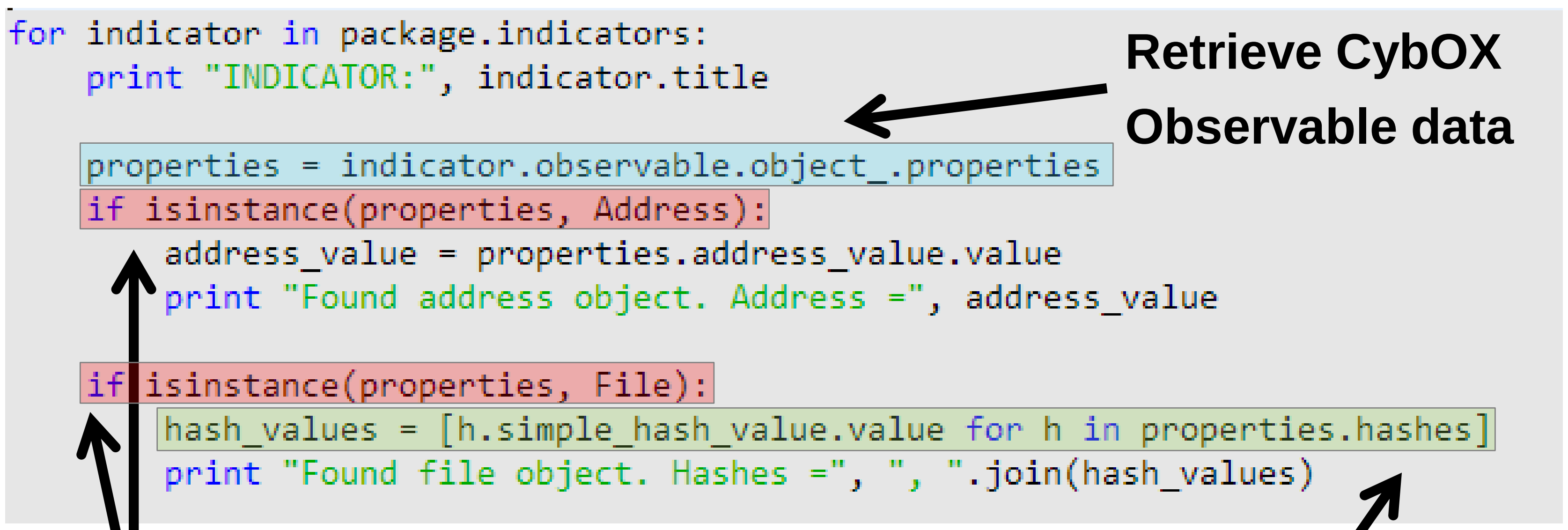
Pay attention to which elements are lists. Iterate over these appropriately.

Output:



```
Indicators - Malware Artifacts
stixVocabs:PackageIntentVocab-1.0
```

Working with indicators



```
for indicator in package.indicators:
    print "INDICATOR:", indicator.title

    properties = indicator.observable.object_.properties
    if isinstance(properties, Address):
        address_value = properties.address_value.value
        print "Found address object. Address =", address_value

    if isinstance(properties, File):
        hash_values = [h.simple_hash_value.value for h in properties.hashes]
        print "Found file object. Hashes =", ", ".join(hash_values)
```

Retrieve CybOX Observable data

Detect classes in order to properly handle different types

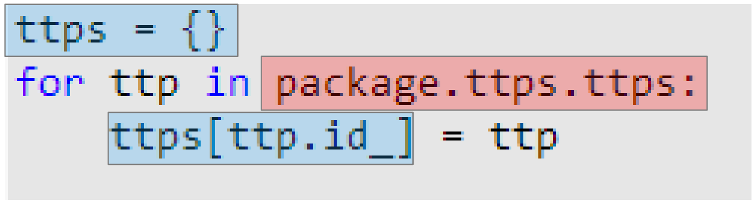
Use python list features to collect lists of data

Things we skipped

- **Condition and other matching**
- **Other fields (be prepared for whatever your sources might send)**
  - Use profiles to agree upon this
- **Other types of combinations (composite indicators, composite observables)**

Working with relationships

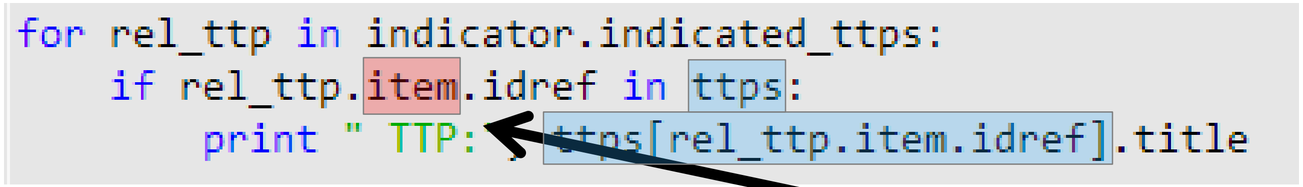
- **Build up dictionaries of IDs for reference targets**



```
ttps = {}
for ttp in package.ttps.ttps:
    ttps[ttp.id_] = ttp
```

Note the gotcha on TTPs

- **When parsing relationships, find idrefs and look them up**



```
for rel_ttp in indicator.indicated_ttps:
    if rel_ttp.item.idref in ttps:
        print " TTP: ", ttps[rel_ttp.item.idref].title
```

Relationship targets are retrieved via item

- **(Above logic could be extracted to a common class)**

Producing STIX in Python

HS SEDI
Homeland Security Systems Engineering and Development Institute

Homeland Security

Creating basic content

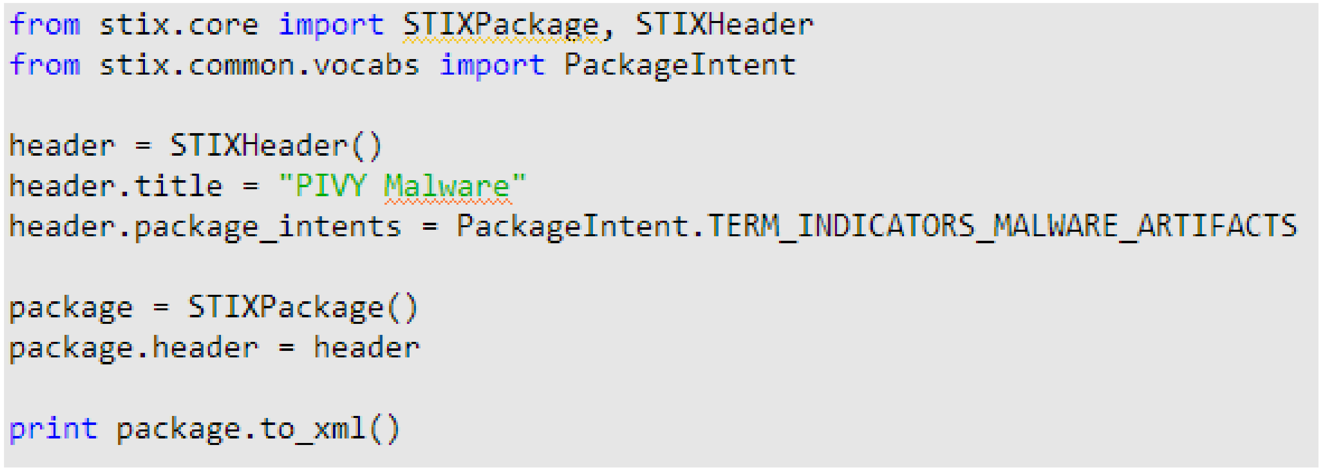
```
from stix.core import STIXPackage, STIXHeader

header = STIXHeader(title="PIVY Malware", package_intents=["Indicators - Malware Artifacts"])
package = STIXPackage(stix_header=header)

print package.to_xml()
```

Namespaces are handled automatically, but classes must be imported (per Python)

Content is often created “inside out” so inner content can be added to outer content

No need to set the xsi:type or create that type. Types are “promoted” whenever possible.

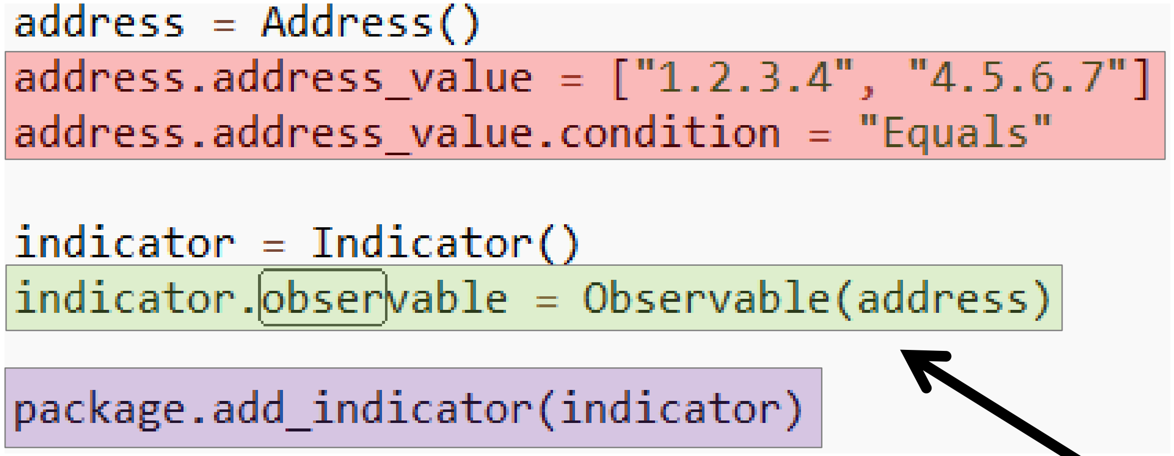
Creating Basic Content (Alternative)



```
from stix.core import STIXPackage, STIXHeader
from stix.common.vocabs import PackageIntent

header = STIXHeader()
header.title = "PIVY Malware"
header.package_intents = PackageIntent.TERM_INDICATORS_MALWARE_ARTIFACTS

package = STIXPackage()
package.header = header

print package.to_xml()
```

- **Again, content created "inside-out"**
  - Header first, then STIX Package
- **Set class attributes via Python "properties" rather than `__init__()` parameters.**
- **Uses `stix.common.vocabs.PackageIntent` for statically defined STIX Controlled Vocabulary terms**

Adding components (indicators, TTPs)



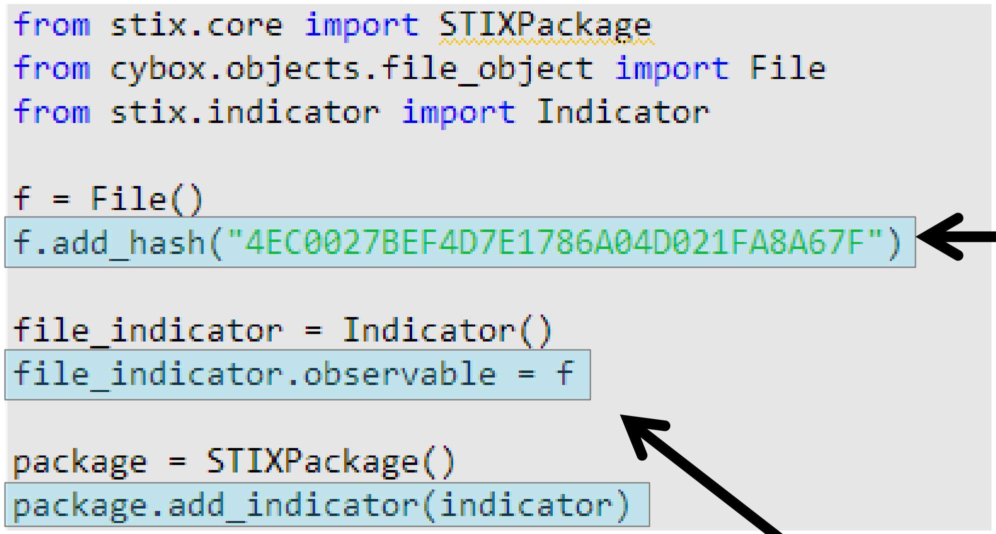
```
address = Address()
address.address_value = ["1.2.3.4", "4.5.6.7"]
address.address_value.condition = "Equals"

indicator = Indicator()
indicator.observable = Observable(address)

package.add_indicator(indicator)
```

Example of “promotion”

Helper that will automatically create “indicators” wrapper

Helper used to set the observable content. It will handle objects, composition, or – as here – properties of a type.

Quiz: File Hash Indicator



```python
from stix.core import STIXPackage
from cybox.objects.file_object import File
from stix.indicator import Indicator

f = File()
f.add_hash("4EC0027BEF4D7E1786A04D021FA8A67F")

file_indicator = Indicator()
file_indicator.observable = f

package = STIXPackage()
package.add_indicator(indicator)
```

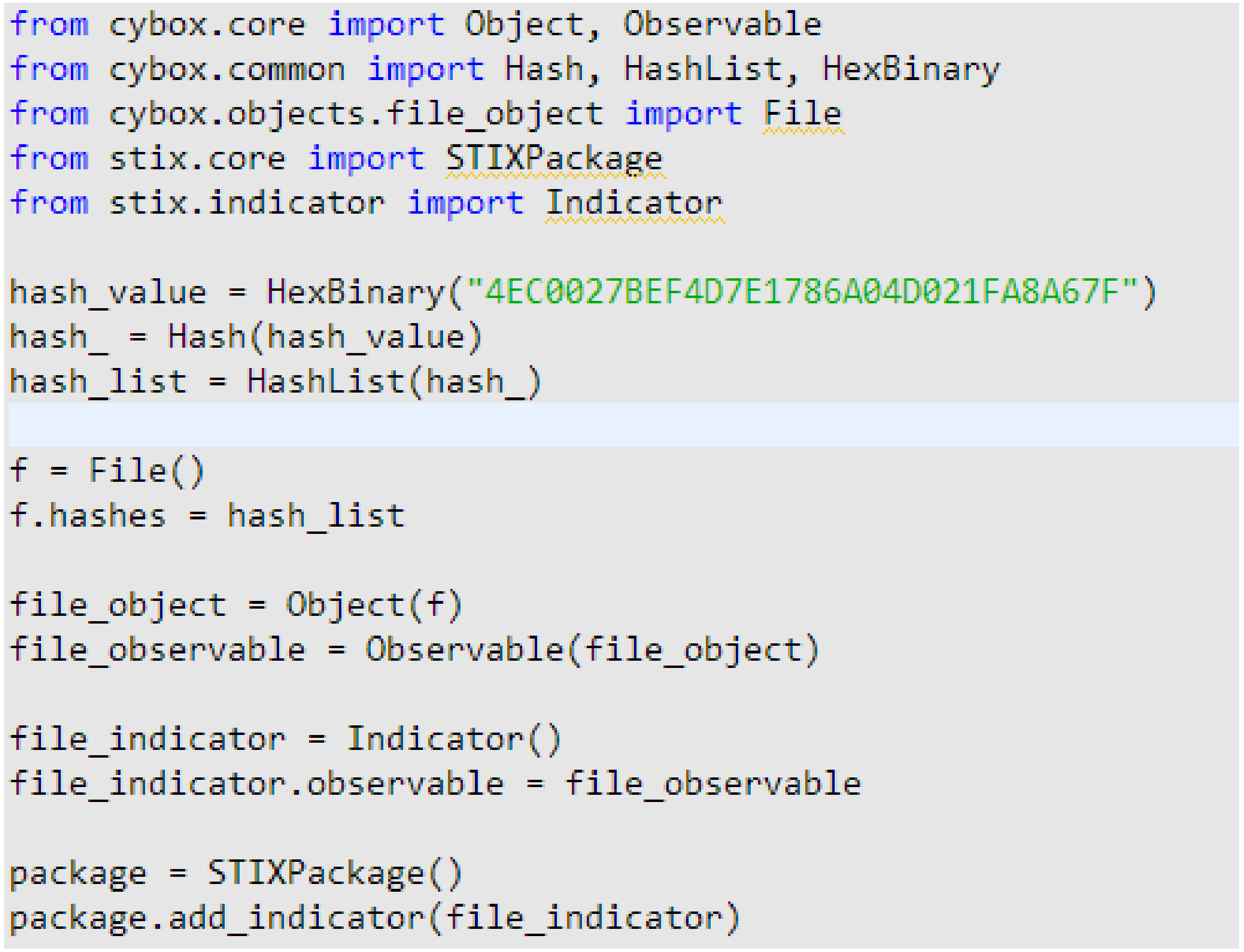
1. Promoted string value to HexBinary instance
2. Automatically determined the hash type and set the appropriate attribute
3. Created a Hash object to hold the HexBinary data
4. Created a HashList object to contain the Hash instance
5. Added HashList instance to File instance

- **Notice any other helpers?**

1. Created Indicators collection wrapper and added File indicator to that collection

1. Wrapped File "properties" in Object instance
2. Created Observable instance and added File Object instance to it

File Hash Indicator Without “Help”

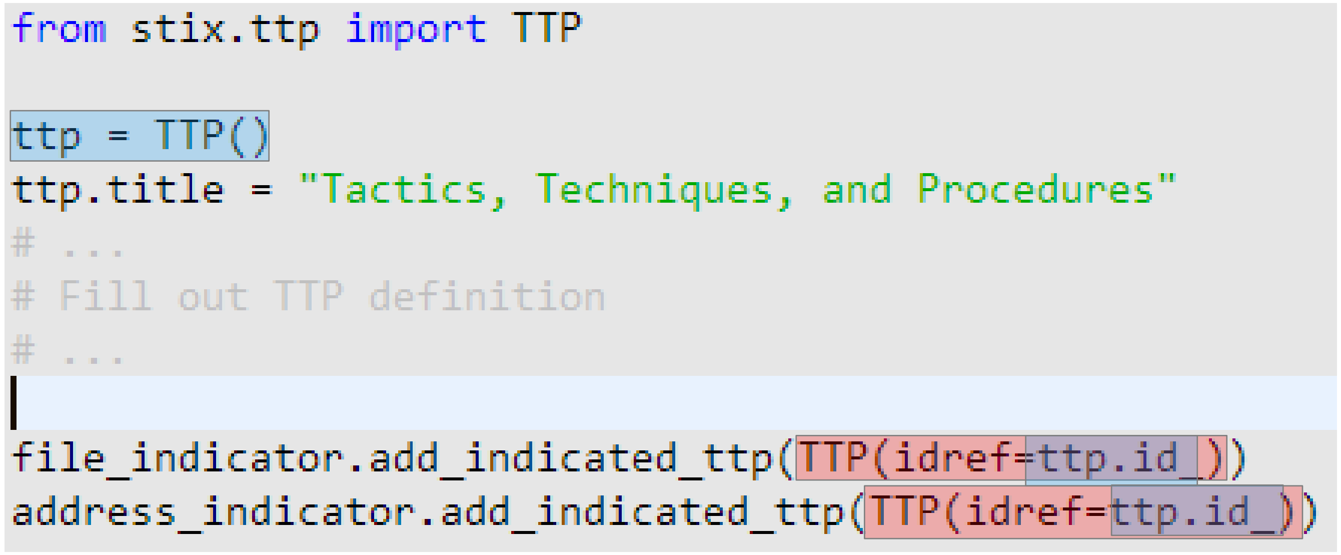


```
from cybox.core import Object, Observable
from cybox.common import Hash, HashList, HexBinary
from cybox.objects.file_object import File
from stix.core import STIXPackage
from stix.indicator import Indicator

hash_value = HexBinary("4EC0027BEF4D7E1786A04D021FA8A67F")
hash_ = Hash(hash_value)
hash_list = HashList(hash_)

f = File()
f.hashes = hash_list

file_object = Object(f)
file_observable = Observable(file_object)

file_indicator = Indicator()
file_indicator.observable = file_observable

package = STIXPackage()
package.add_indicator(file_indicator)
```

- **Create HexBinary instance**
- **Create Hash instance with HexBinary value**
- **Create HashList with single Hash instance value**
- **Create File object properties instance**
- **Set File.hashes to HashList instance**
- **Create Object layer and add ObjectProperties to it**
- **Create Observable layer and add Object to it**
- **Create Indicator and add Observable to it**
- **Create STIX Package and add indicator to it**

Creating Relationships

- **Add the content you're pointing to, then create a new object of that type and set the idref in the constructor**



```
from stix.ttp import TTP

ttp = TTP()
ttp.title = "Tactics, Techniques, and Procedures"
# ...
# Fill out TTP definition
# ...

file_indicator.add_indicated_ttp(TTP(idref=ttp.id_))
address_indicator.add_indicated_ttp(TTP(idref=ttp.id_))
```

A new TTP is created, the idref is set to the source TTPs ID

What to do with it?

- **Depends on your use case**
  - Write to a file?
  - Publish via TAXII?
  - Save into a data store for later analysis, correlation, publishing?
- **Use .to_xml() to write to string**

Summary

HS SEDI
Homeland Security Systems Engineering and Development Institute

Homeland Security

Many things are handled automatically

- IDs (mostly)
- Timestamps
- apply_condition and CybOX lists
- xsi:type
  - Just use the correct subclass and the xsi:type will be set when necessary
- Namespaces
- Schema locations

Other things are still manual

- Does not tell you what content to produce or consume: you still need to understand the modeling
- Relationships are not automatically dereferenced
- There are still bugs, though we work hard to squash them

java-stix

Producing and consuming STIX via the JAXB bindings

HS SEDI
Homeland Security Systems Engineering and Development Institute

Homeland Security

Compared to python-stix

- **Less happens automatically**
  - No helpers for IDs, lists, etc.

- **Does insulate you from the XML**
  - Automatically generates namespaces, etc.
  - Automatic handling of xsi:type
  - Helpers beyond JAXB for generating and parsing XML

- **Less documentation**

Getting the bindings

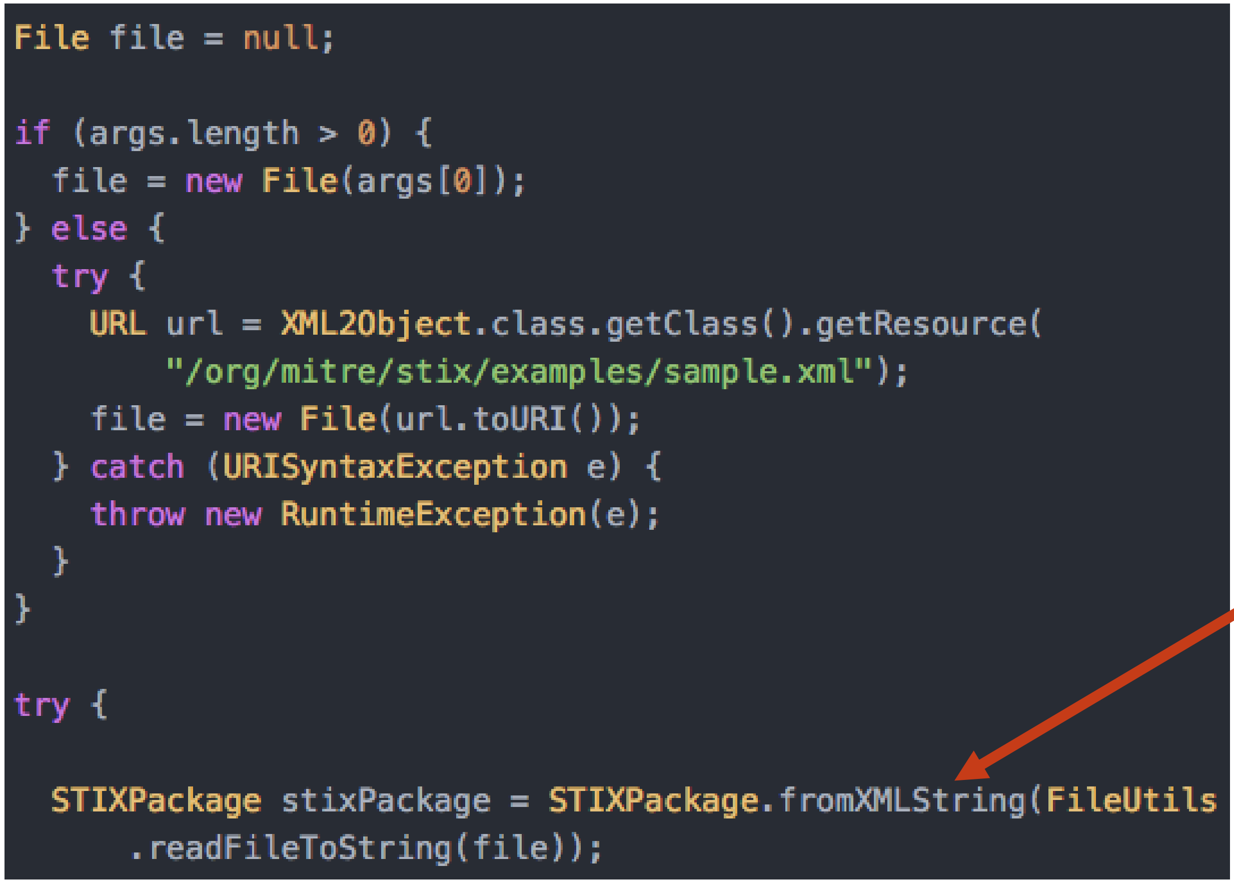
- **Distributed via Maven Central Repository**
  - Group: org.mitre
  - Artifact: stix
  - Version 1.2.0.2

- **Installable in either Maven, Gradle, or Ivy**

Consuming STIX in Java

Load the document

- **Often via file:**

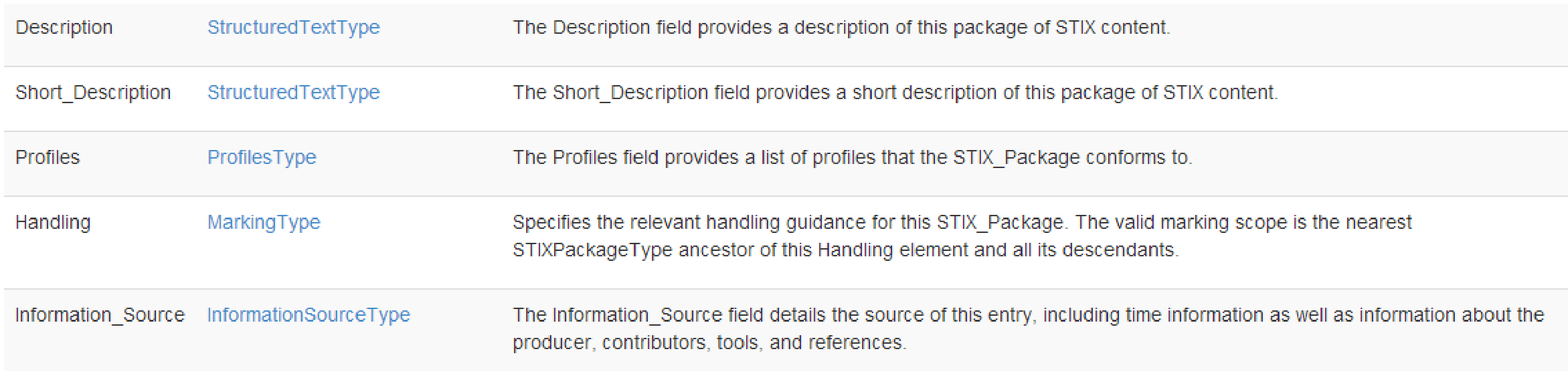


```
File file = null;

if (args.length > 0) {
  file = new File(args[0]);
} else {
  try {
    URL url = XML2Object.class.getClass().getResource(
        "/org/mitre/stix/examples/sample.xml");
    file = new File(url.toURI());
  } catch (URISyntaxException e) {
    throw new RuntimeException(e);
  }
}

try {

  STIXPackage stixPackage = STIXPackage.fromXMLString(FileUtils
      .readFileToString(file));
```

Or a string

Data model vs. java-stix

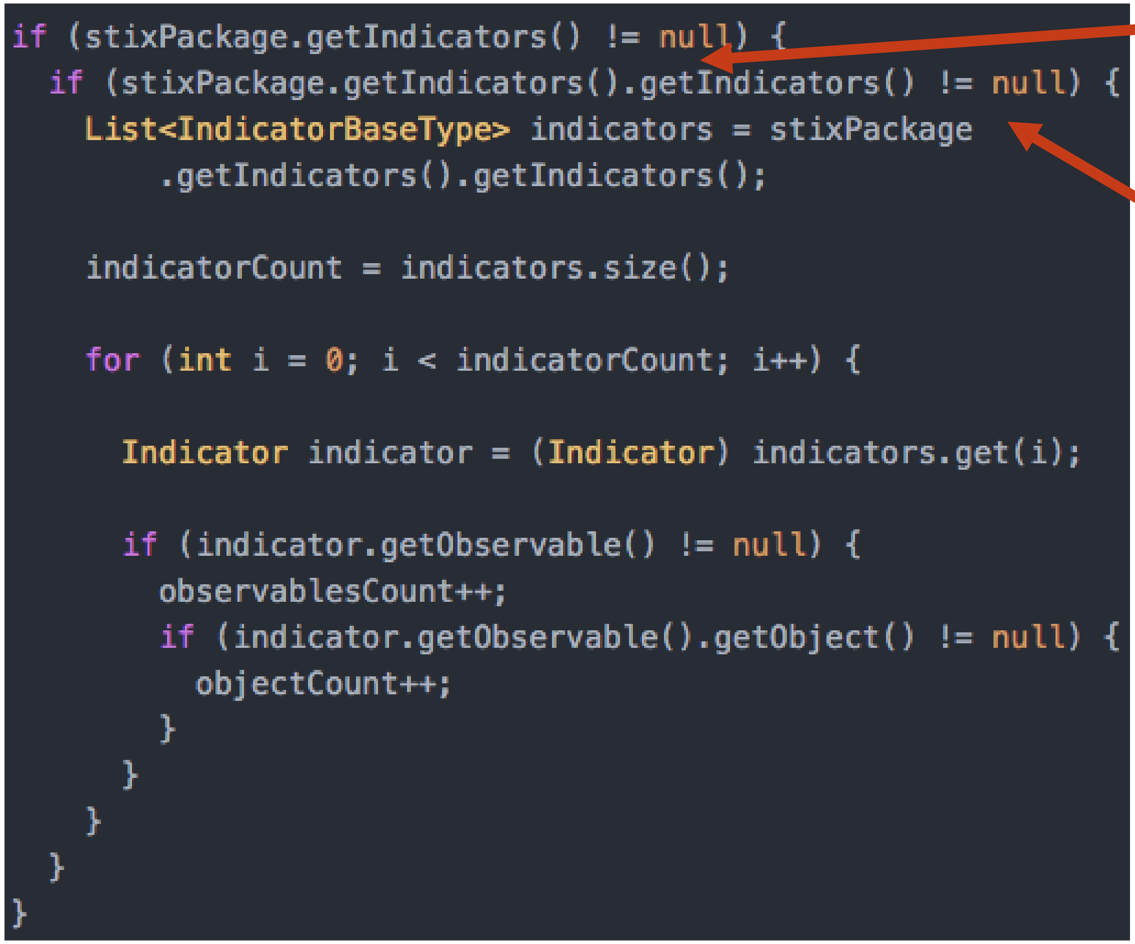
- **Use the data model to find field meanings and contents**



Description	StructuredTextType	The Description field provides a description of this package of STIX content.
Short_Description	StructuredTextType	The Short_Description field provides a short description of this package of STIX content.
Profiles	ProfilesType	The Profiles field provides a list of profiles that the STIX_Package conforms to.
Handling	MarkingType	Specifies the relevant handling guidance for this STIX_Package. The valid marking scope is the nearest STIXPackageType ancestor of this Handling element and all its descendants.
Information_Source	InformationSourceType	The Information_Source field details the source of this entry, including time information as well as information about the producer, contributors, tools, and references.

- **Those will map to standard JAXB-transformed getters and setters**
  – You can also build JavaDoc

Accessing data



```
if (stixPackage.getIndicators() != null) {
  if (stixPackage.getIndicators().getIndicators() != null) {
    List<IndicatorBaseType> indicators = stixPackage
        .getIndicators().getIndicators();

    indicatorCount = indicators.size();

    for (int i = 0; i < indicatorCount; i++) {

      Indicator indicator = (Indicator) indicators.get(i);

      if (indicator.getObservable() != null) {
        observablesCount++;
        if (indicator.getObservable().getObject() != null) {
          objectCount++;
        }
      }
    }
  }
}
```

Note this pattern for list types

Null checks to make sure elements are there

Producing STIX in Java

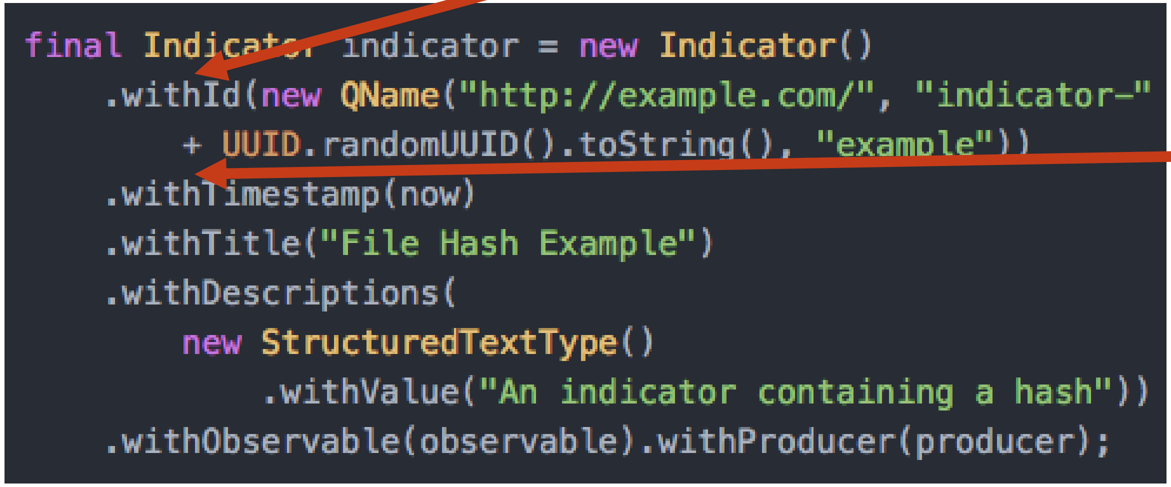
HS SEDI
Homeland Security Systems Engineering and Development Institute

Homeland Security

Creating basic content

Use with* methods to set content

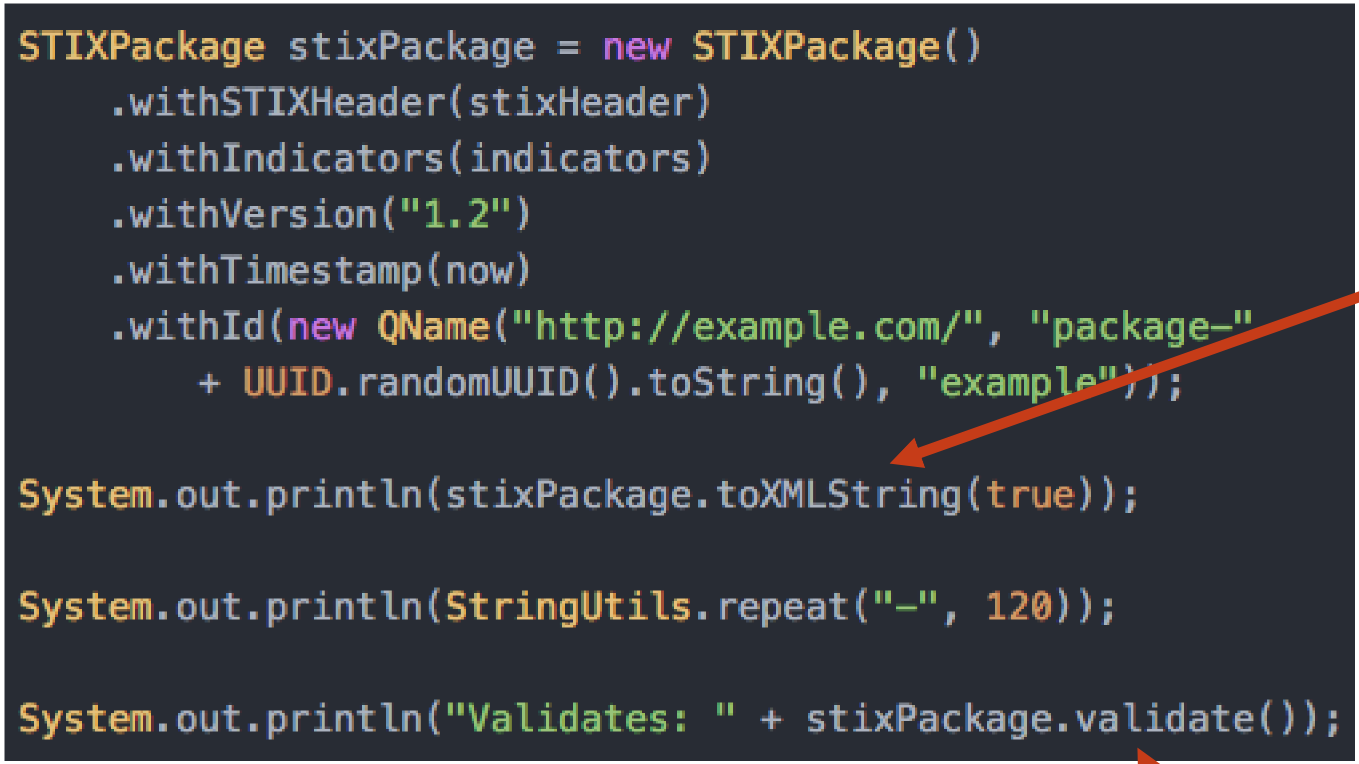
Note that they can be chained



```java
final Indicator indicator = new Indicator()
    .withId(new QName("http://example.com/", "indicator-"
        + UUID.randomUUID().toString(), "example"))
    .withTimestamp(now)
    .withTitle("File Hash Example")
    .withDescriptions(
        new StructuredTextType()
            .withValue("An indicator containing a hash"))
    .withObservable(observable).withProducer(producer);
```

HS SEDI
Homeland Security Systems Engineering and Development Institute

Package helpers



```
STIXPackage stixPackage = new STIXPackage()
    .withSTIXHeader(stixHeader)
    .withIndicators(indicators)
    .withVersion("1.2")
    .withTimestamp(now)
    .withId(new QName("http://example.com/", "package-"
        + UUID.randomUUID().toString(), "example"));

System.out.println(stixPackage.toXMLString(true));

System.out.println(StringUtils.repeat("-", 120));

System.out.println("Validates: " + stixPackage.validate());
```

Print to string

You can even validate! Better than python!

Summary and Tips

Creating and understanding STIX in XML, Python, Java

HS SEDI
Homeland Security Systems Engineering and Development Institute

Homeland Security

Thoughts on Consuming STIX

- **Make sure you understand the content you will be working with**
  – Use profiles, informal agreements, writeups, or other mechanisms

- **Be sure to check for all possibilities**
  – Iterate over lists, don't just retrieve the first item
  – Items might or might not always be in default vocabularies

- **Handle relationships that don't resolve in the document**

- **Handle relationships with cyclic loops (STIX graphs may be cyclic)**

- **Read the security considerations guide to avoid potential attacks (in particular DoS)**

Thoughts on Producing STIX

- As always, start with some idea of what to model
- Be consistent: remember that consumers need to parse this so restrain yourself to as limited a subset as necessary
- Use the data model docs and (when available) API docs in conjunction to understand what to produce and how to produce it
- If possible, describe your service or content production via a profile, write-up, or informal agreement
- Validate your content! There’s so many ways!