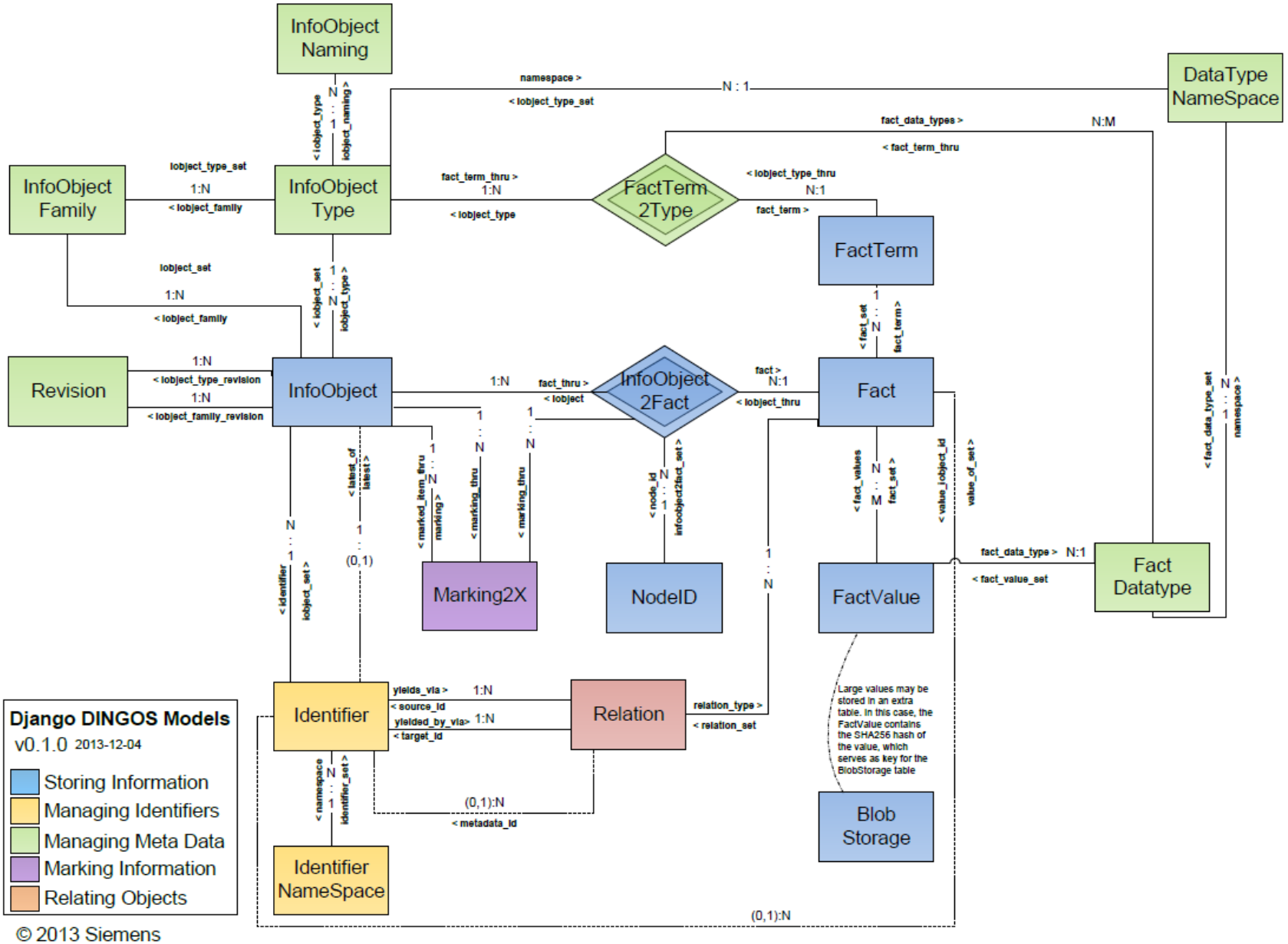
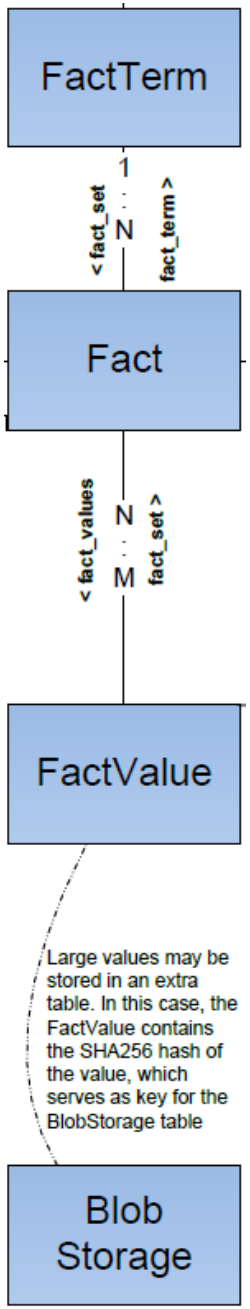
Django DINGOS: Developers' Overview





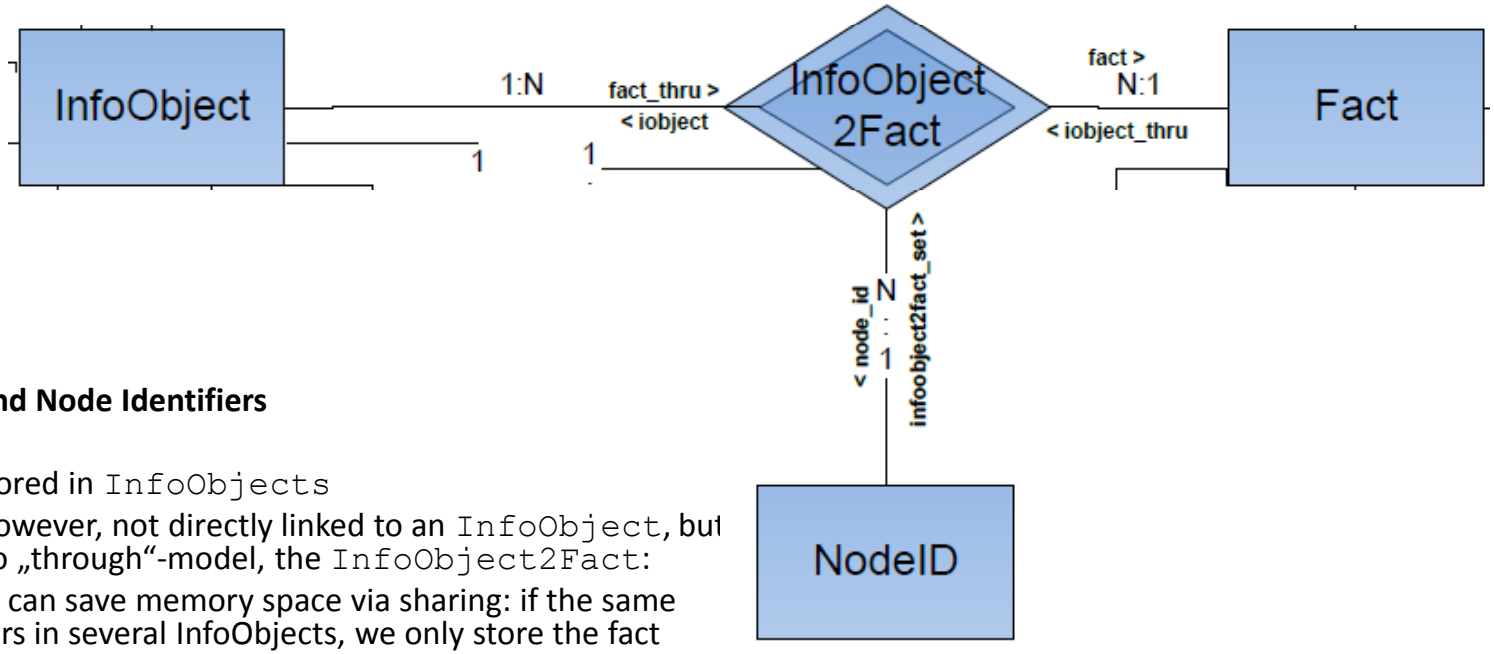
How Facts are stored

- DINGOS stores facts as combination of `FactTerm` and associated `FactValues`
- Example: consider the following XML:

```
<person>
    <firstName>John</firstName>
    <lastName>Smith</lastName>
    <age>25</age>
    <address>
        <streetAddress>21 2nd Street</streetAddress>
        <city>New York</city>
        <state>NY</state>
        <postalCode>10021</postalCode>
    </address>
    <phoneNumbers>
        <phoneNumber type="home">212 555-1234</phoneNumber>
        <phoneNumber type="fax">646 555-4567</phoneNumber>
    </phoneNumbers>
</person>
```

This yields the following `FactTerms` and `FactValues`:

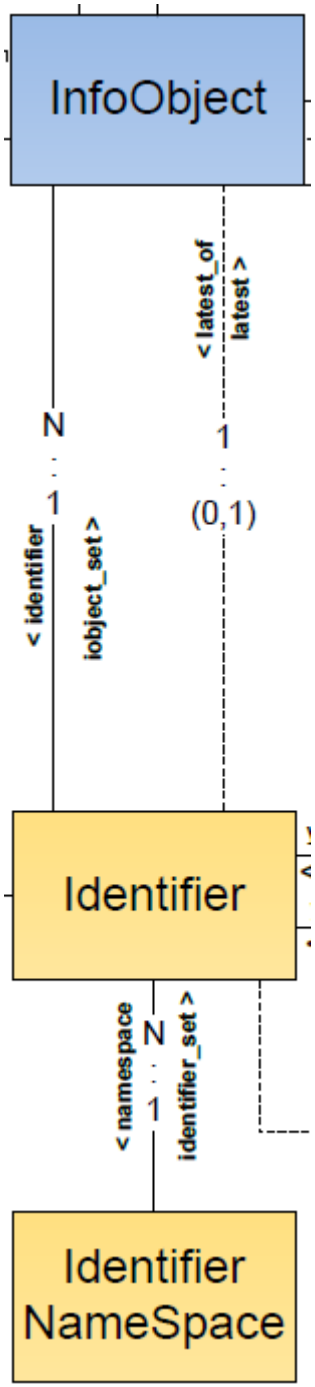
Fact Term	Fact Value
person/firstName	John
person/lastName	Smith
person/age	25
person/address/streetAddress	21 2nd Street
person/address/city	New York
person/address/state	NY
person/address/postalCode	10021
person/phoneNumbers/phoneNumber@type	home
person/phoneNumbers/phoneNumber	212 555-1234
person/phoneNumbers/phoneNumber@type	fax
person/phoneNumbers/phoneNumber	646 555-4567



InfoObjects and Node Identifiers

- Facts are stored in `InfoObjects`
- They are, however, not directly linked to an `InfoObject`, but via a Django „through"-model, the `InfoObject2Fact`:
  - Thus, we can save memory space via sharing: if the same fact occurs in several InfoObjects, we only store the fact once
  - We need a way to provide positional information: otherwise we loose structural information.
- Structural information is kept in the node identifier NodeID. For example:
  - the the fact with `FactTerm person/firstName` carries the node identifier „`N0000:N0000`"
    - ‚`person`' is first (and only) top-level node of the XML
    - ‚`firstName`' is the first child node of ‚person' in the XML
  - The fact with `FactTerm person/lastName` carries the node identifier „`N0000:N0001`", because „`lastName`" is the second child node of ‚`person`'
  - The fact with `FactTerm person/phoneNumbers/phoneNumber@type` carries the node identifier „`N0000:N0004:L0000:A0000`", because
    - ‚`phoneNumbers`' is the 5th child of ‚`person`'
    - ‚`phoneNumber`' is the first element in a list of ‚`phoneNumber`' elements under ‚`phoneNumbers`'
    - ‚`type`' is the first attribute at this position.

Fact Term	Fact Value
person/firstName	John
person/lastName	Smith
person/age	25
person/address/streetAddress	21 2nd Street
person/address/city	New York
person/address/state	NY
person/address/postalCode	10021
person/phoneNumbers/phoneNumber@type	home
person/phoneNumbers/phoneNumber	212 555-1234
person/phoneNumbers/phoneNumber@type	fax
person/phoneNumbers/phoneNumber	646 555-4567



InfoObjects, Identifiers and Revisions

- Each InfoObject has an Identifier and (stored internally) a timestamp
- InfoObjects of a given identifier may occur in several revisions, each revision marked with its specific timestamp
- The most recent (i.e. latest) revision of an InfoObject is marked in the data model by a pointer ‚latest' from the Identifier

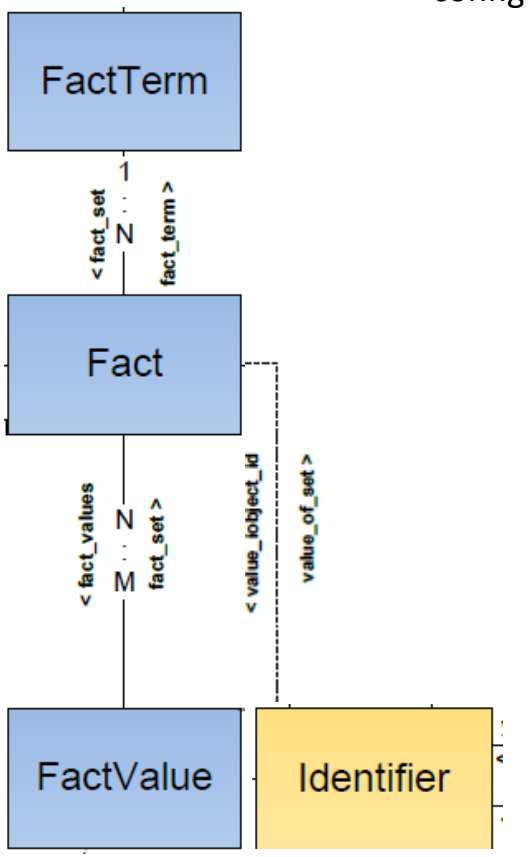
The structure of Identifiers

- Each Identifier has a namespace
- The intended use of the namespace is to communicate the „owner organization" of an object
- For the STIX/CybOX-example below, the STIX/CybOX importer has been configured so as to extract the identifier "`Object-3a7aa9db-d082-447c-a422-293b78e24238`" with namespace "`http://example.com`"

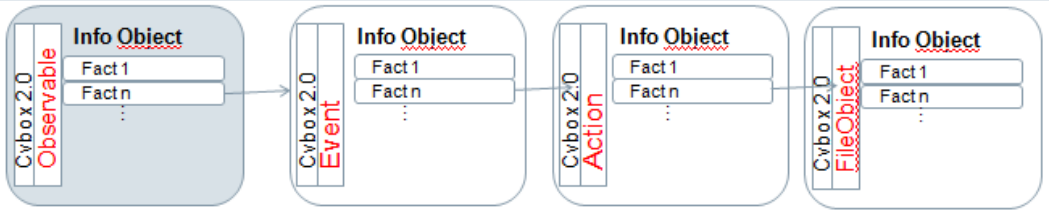
```
<stix:STIX_Package xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  (...)
  xmlns:examle="http://example.com"
  xmlns:EmailMessageObj="http://cybox.mitre.org/objects#EmailMessageObject-1"
  xmlns:cybox="http://cybox.mitre.org/cybox-2"
  (...)>
  (...)

    <cybox:Object id="example:Object-3a7aa9db-d082-447c-a422-293b78e24238">
     <cybox:Properties xsi:type="EmailMessageObj:EmailMessageObjectType">
      <EmailMessageObj:Header>
       <EmailMessageObj:From category="e-mail">
        (...)
```

Referring from one InfoObject to another (I): Embedded objects

- A Fact can contain a reference to another InfoObject rather than FactValue
- This reference points to an Identifier rather than an InfoObject model. This is because the Identifier groups different revisions of an information object (and each of these revisions is stored as separate InfoObject).
- DINGOS importers can be configured to factor out embedded objects into separate InfoObjects.
- For example, for the following STIX object, the STIX/CybOx importer has been configured to extract a chain of embedded objects as shown below:

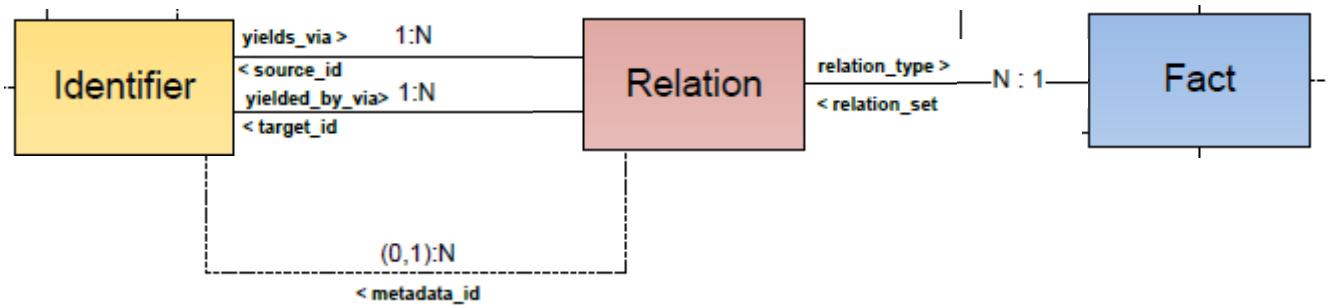



```
<cybox:Observable id="example:Observable-a727a717-1852-4c79-9a16-2f3a8b4632c2">
    <cybox:Event id="example:Event-44578866-b0c5-4551-84dd-0f1f02f8210f">
        <cybox:Actions>
            <cybox:Action id="example:Action-a18a058c-effa-4060-b8be-25e1b1ade75f" action_status="Succe
                          context="Host" timestamp="2013-04-08T09:22:00.0Z">
                <cybox:Type xsi:type="cyboxVocabs:ActionTypeVocab-1.0">Create</cybox:Type>
                <cybox:Name xsi:type="cyboxVocabs:ActionNameVocab-1.0">Create File</cybox:Name>
                <cybox:Associated_Objects>
                    <cybox:Associated_Object id="example:Object-5ec92e95-a31f-470b-97c4-aa9046189fbb">
                        <cybox:Properties xsi:type="FileObj:FileObjectType">
                            <FileObj:File_Name>foobar.dll</FileObj:File_Name>
                            <FileObj:File_Path>C:\Windows\system32</FileObj:File_Path>
                            <FileObj:Hashes>
                                <cyboxCommon:Hash>
                                    <cyboxCommon:Type>MD5</cyboxCommon:Type>
                                    <cyboxCommon:Simple_Hash_Value datatype="hexBinary">
                                     6E48C348D742A931EC2CE90ABD7DAC6A
                                    </cyboxCommon:Simple_Hash_Value>
                                </cyboxCommon:Hash>
                            </FileObj:Hashes>
                        </cybox:Properties>
                        <cybox:Association_Type
                         xsi:type="cyboxVocabs:ActionObjectAssociationTypeVocab-1.0">
                         Affected</cybox:Association_Type>
                    </cybox:Associated_Object>
                </cybox:Associated_Objects>
            </cybox:Action>
        </cybox:Actions>
    </cybox:Event>
</cybox:Observable>
```



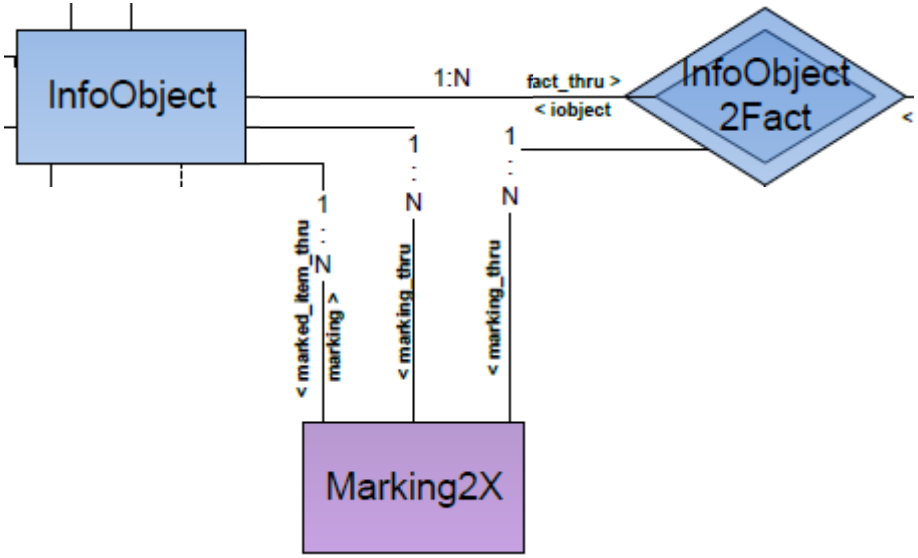
Referring from one InfoObject to another (II): Relationships

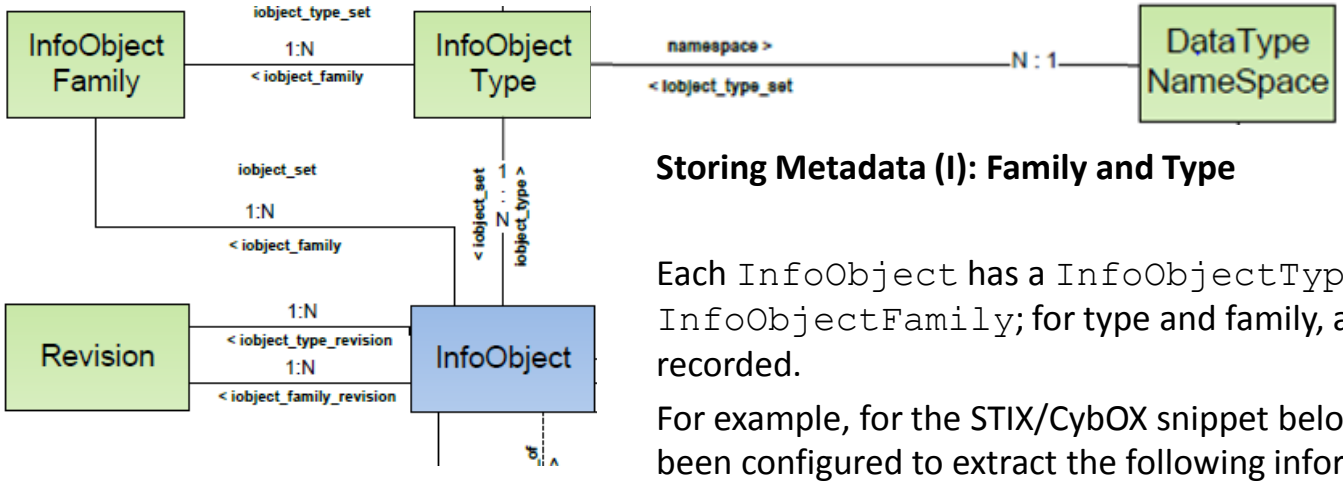
- `Relations` can be used to relate `InfoObjects` to each other.
- As before, these references point `Identifiers` rather than `InfoObjects`: this is because the `Identifier` groups different revisions of an information object (and each of these revisions is stored as separate `InfoObject`).
- A '`relation_type`' can be specified as a single `Fact` (the `Fact` contains the relation type as fact value and a configurable default `FactTerm`)
- Relationship metadata can be specified in an `InfoObject`



Marking Information

- Markings are InfoObjects that 'mark' existing information.
- We use Django's content type mechanism to relate markings to different models: currently, we can mark `InfoObject`s as well as facts within an `InfoObject` (this through the `InfoObject2Fact` model)
- The intention is to implement markings as used in the STIX standard
- Not that
  - Unlike relationships and embeddings, here the reference is directly to the `InfoObject` rather than an `Identifier`, i.e., markings do not automatically carry over between revisions
  - Of all DINGOS 0.1.0 models, markings are most likely to change in future.





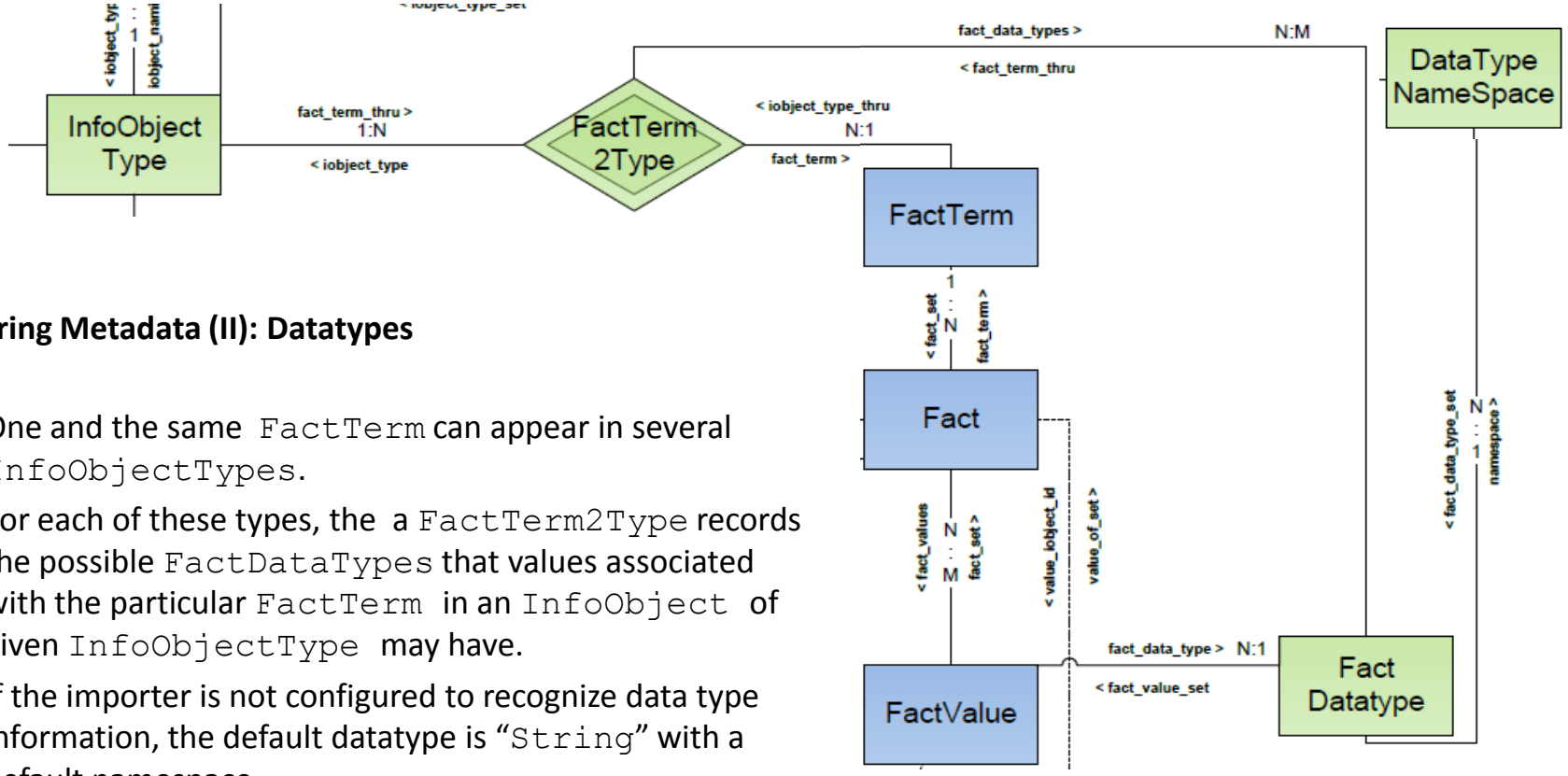
Storing Metadata (I): Family and Type

Each `InfoObject` has a `InfoObjectType` and a `InfoObjectFamily`; for type and family, a `Revision` is recorded.

For example, for the STIX/CybOX snippet below, the importer has been configured to extract the following information for a resulting "EmailMessageObj" InfoObject:

- Family is "`cybox.mitre.org`" and family revision is "`2`"
- Type is "`EmailMessageObj`" and type revision is "`1`"; the `DataTypeNamespace` associated with the `InfoObjectType` is "`http://cybox.mitre.org/objects#EmailMessageObject`".

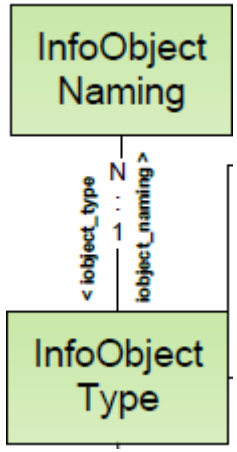
```
<stix:STIX_Package xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  (...)
  xmlns:examle="http://example.com"
  xmlns:EmailMessageObj="http://cybox.mitre.org/objects#EmailMessageObject-1"
  xmlns:cybox="http://cybox.mitre.org/cybox-2"
  (...)>
  (...)

    <cybox:Object id="example:Object-3a7aa9db-d082-447c-a422-293b78e24238">
     <cybox:Properties xsi:type="EmailMessageObj:EmailMessageObjectType">
      <EmailMessageObj:Header>
       <EmailMessageObj:From category="e-mail">
        <AddressObj:Address_Value
         condition="Contains">@state.gov
        </AddressObj:Address_Value>
       </EmailMessageObj:From>

      </EmailMessageObj:Header>
     </cybox:Properties>
    </cybox>
  (...)
</stix>
```



Storing Metadata (II): Datatypes

- One and the same `FactTerm` can appear in several `InfoObjectTypes`.
- For each of these types, the a `FactTerm2Type` records the possible `FactDataTypes` that values associated with the particular `FactTerm` in an `InfoObject` of given `InfoObjectType` may have.
- If the importer is not configured to recognize data type information, the default datatype is "`String`" with a default namespace.
- For the STIX/CybOX snippet below, the importer has been configured to extract the following data type information about the value associated with FactTerm "`Properties/Hashes/Hash`":
  - FactDataType is "`HashNameVocab-1.0`"
  - Namespace of datatype "`HashNameVocab-1.0`" is as specified in the top-level element of the XML file for '`cyboxVocabs`'.

```
<cybox:Properties xsi:type="FileObj:FileObjectType"  >
 (...)
 <FileObj:Hashes>
  <cyboxCommon:Hash>
   <cyboxCommon:Type xsi:type="cyboxVocabs:HashNameVocab-1.0">
    MD5
   </cyboxCommon:Type>
   <cyboxCommon:Simple_Hash_Value>
    cf2b3ad32a8a4cfb05e9dfc45875bd70
   </cyboxCommon:Simple_Hash_Value>
  </cyboxCommon:Hash>
 </FileObj:Hashes>
</cybox:Properties>
```



Using Metadata: Extracting object names from facts

- An `InfoObjectType` can be associated with several `InfoObjectNaming` schemas.
- When importing an InfoObject, the naming schemas are used to derive an object name:
  - The schemas are tried out in order (each `InfoObjectNaming` carries an ordinal).
  - The first schema for which all required facts are present in the object is used to extract a name
  - If no schema yields a name, then a default name (currently the name of the InfoObjectType followed by the number of facts in the InfoObject) is used.
- For example, the STIX/CybOX importer defines the following naming schemas for a CybOX "FileObject":
  - `[Properties/File_Name] ([fact_count] facts)`
  - `[Properties/Hashes/Hash/Type]:[Properties/Hashes/Hash/Simple_Hash_Value] ([fact_count] facts)`
  - `[fact_count_equal_1?][term_of_fact_num_0] = [value_of_fact_num_0]`
- Now:
  - if a file object has a fact with fact term '`Properties/FileName`' and value "`evil.exe`" and contains 13 facts, then the name given to the object is "`evil.exe (13 facts)`".
  - if a file object specifies no file name but one or more hash values, the first hash value is used to extrat the object name as "`<hash_value> (<# of facts> facts)`".
  - If neither file name nor hash value is specified, but the object contains exactly one fact, then "`<fact_term> = <fact value> (<# of facts> facts)`".
    Is extracted as object name
- *Note*: you can specify/change naming schemas in the Django admin interface for `InfoObjectType` models (the standard URI is '`<server>/admin/dingos/factdatatype/`'.