Building the Ontology Policies

1.0 Naming Conventions

Naming conventions are critical for every project and organization. The policies outlined here are designed to ensure consistency across all of our domain areas and ontologies, to maximize usability across a wide variety of tools, and generally follow best practices used in ontology engineering.

1. Namespace definitions for ontologies are critical, along with policies for their management that are well understood. An approach that generally follows the pattern http[s]://<authority>/<subdomain>/<topic>/<date, in YYYYMMDD form>/filename.extension is common practice today, with content negotiation pointing to the latest version. Levels of hierarchy may be added for large organizations or modularized ontologies. Namespace prefixes (abbreviations) for individual modules, especially where there are multiple modules, can be important, too, as people often recall prefixes far better than lengthy URIs. For FIBO, there is an extensive strategy for our URIs, but the basic pattern for non-versioned IRIs, repeated here for the sake of clarity, is:

   https://spec.edmcouncil.org/fibo/<domain abbreviation>/<module>/.../<module>/<ontology name>/

   Note that the strategy defined herein reflects what we do in the "gold source" GitHub versions of our ontologies, rather than the IRIs generated via the publication process.

2. Domain-level names and abbreviations are assigned by the FLT. Near-term FCT activities reflect the naming architecture already present in the released and provisional ontologies. Module architecture and ontology naming can be done by an FCT, with approval for module level naming from the FLT. At the domain level, for FIBO URIs that are mirrored in OMG standards, the abbreviations must be upper case, all caps, and should reflect 3-4 character initialisms or acronyms. At the module and ontology level, we use upper camel case, no intervening special characters such as "_" or "-". Filename extensions should be ".rdf" for RDF/XML serialized OWL, which the serializer supports.

3. Namespace prefixes for FIBO follow the pattern: fibo-<domain abbreviation>-<module abbreviation>-...-<module abbreviation>-<ontology abbreviation>, where all elements between the dashes are lower-case alpha. We use the same domain abbreviations in the prefixes as assigned by the FLT. Module and ontology prefixes can be specified by the FLT, with approval for the module level abbreviations from the FLT.

4. Typically for FIBO, we have used plurals in ontology names, and singular forms for all class and property names, which limits confusion and provides an additional level of consistency across our ontology architecture, for example, FinancialInstruments for the ontology name, FinancialInstrument for the class name within the FinancialInstruments ontology. Module and ontology names should be unique (there are a couple of legacy modules that have non-unique names, but no new duplication should be introduced).

For naming entities in an ontology, there are also rules of thumb that may vary by community of practice. For example:

- data modelers often use underscores at word boundaries, spaces in names – which semantic web tools may not handle well (ok in labels)

- some name properties <domain><predicate><range>, others <predicate><range>, & others <predicate>, but not necessarily consistently
- ***semantic web practitioners typically use camel case (upper camel case for classes and datatypes, lower camel case for properties),*** *which allows our ontologies to be consumed by the broadest range of tooling possible*

  example class: **F**inancial**I**nstrument, example property: **h**as**P**art

- ***using verbs to the degree possible for property naming, without incorporating the domain/range is preferable***

For entity naming in FIBO, the following conventions apply:

5. Class names should be expressed in upper camel case, no special characters, no abbreviations in names. There are rare examples where we deviate from this policy, such as for code lists that are automatically generated, but otherwise, abbreviations should not be used despite resulting in lengthy names in some cases. Abbreviations should be included as annotations on the class where appropriate. Class names should not be duplicated - *i.e.*, having a class named Lifecycle in two different ontologies, regardless of the domain or module, is strictly prohibited. There have been cases where a class introduced at some domain level has been needed in another domain, and thus the class has been promoted to a higher, common level in the ontology architecture - in these cases, the lower level class should be deprecated.

6. Property names, both object and data properties, should be expressed in lower camel case, no special characters, no abbreviations in names. Verbs should be used for property naming where possible, without inclusion of the name of the domain or range, with some exceptions for properties of the form "has x", for readability purposes. Property names should not be duplicated - i.e., having a property named hasJurisdiction in two different ontologies, regardless of the domain or module, is strictly prohibited. There have been cases where, due to legacy naming or moving properties from a lower level domain to an upper-level domain have resulted in temporary duplication - in these cases, the lower level property should be deprecated.

7. Naming conventions for individuals are less consistent across practitioners. ISO 704 promotes the use of lower case for individuals unless they incorporate proper names, which we have largely followed in FIBO for labels, but not at the URI level. Individual names should be expressed in upper camel case, no special characters, no abbreviations in names. There are rare examples where we deviate from this policy, such as for code lists that are automatically generated, but otherwise, abbreviations should not be used despite resulting in lengthy names in some cases. Abbreviations should be included as annotations on the individual where appropriate. Individual names should not be duplicated if at all possible, but there are times when this is difficult to avoid, such as with respect to stages or states related to something. In these cases, the FCT should determine the right approach, given that the namespace will distinguish the individuals from one another.

8. Every (Classes, Properties and Individuals) curated entity must have a label and a definition, at a minimum, with additional annotations, such as the source for the term or definition, strongly encouraged.

9. Labels should be expressed in lower case, English, with proper spacing as if they were written as text. The only exception to the lower case rule in labels is for proper names, which may be capitalized, as appropriate. Abbreviations should be captured in upper case, no spacing, with special characters only in

cases where the abbreviation is commonly used with an embedded hyphen, for example. There should be only one rdfs:label for any entity, and it should be the natural language (English) representation of that entity, space separated, with all other names represented using synonym or abbreviation annotations.

10. Definitions should be ISO 704 conformant, meaning, expressed as partial sentences that can be used to replace the term in a sentence. Any additional clarification, scope notes, explanatory notes, or other comments on the use of a given concept should be incorporated in other annotations.

2. High Level Disjoints

FIBO allows High Level Disjoint Class Axioms. But, they are to be used with discretion.

3. OWL Documentation

There should be an RDFS label and a SKOS definition for every class and for every property.

The definition should ideally have a source reference to back it up

For concepts and properties the definition should use skos:definition;
rdfs:comment is not to be used at all.

For individuals, do not use skos:definition, maybe use rdfs:comment. Exception is for individuals that are parts of a vocabulary (e.g instance of skos:Concept) in which case skos:definition is fine.

Examples: The (suggested) policy is that if a note explains more than just examples, this would go in the explanatoryNote annotation. We would use the SKOS example if there are discrete examples with no additional explanatory material.

4. Alternative Labels

Do not use altLabel, use synonym or abbreviation. Abbreviation should be a subproperty of synonym.

5. When to use datatype properties

When you want to use a property to point to some kind of value you can either use a datatype property or you can reify the value and use a object property. In general it is good practice to use object properties. If you are tempted to use a datatype property, be sure the answer to the following question is NO.

"Given a property that has some kind of value, do you now or might you in the future have the need to say something about that value?"

6. Qualified Relations Pattern

FIBO will be developed and published using a 3 class pattern, commonly known as the qualified relation pattern. http://patterns.dataincubator.org/book/qualified-relation.html There may be use cases where it can be proven that 5 classes are necessary. For testing we will develop multiple instances

7. FIBO uses an accepted industry standard for definition citations

FIBO definition citations are only any ISO standard, any OMG standard, any other standards such as FPML, GLEIF, ISDA, CFTC, OFR, FINRA, any government document/site/law (for the US, ending in .gov), including European Union (europa.eu), academic sites allowing free citations. Existing definitions no cited to any of the above will be deprecated as FIBO is updated.

FIBO citations will be based on the IEEE citation format. Guidelines on reference citations for the IEEE are spelled out in:

https://ieee-dataport.org/sites/default/files/analysis/27/IEEE%20Citation%20Guidelines.pdf.

The IEEE citation format will be followed for online, print, personal, and all other references.

8. FIBO encourages maximum use of Inverse Properties

When creating a property, the question arises as to when to create explicitly named inverse property.

1. If the property is symmertic, it is its own inverse, hence it should never have a separate named inverse property.
2. Unless there is a good reason, do not create an explicit named inverse for a given property
3. A good reason to create an inverse is if it is likley to be desired and used. Example include:
   a. The inverse property is meaningful to the business, has a common name and users are likely to expect to see it.
   b. The inverse property is likely to be used when creating triples data.
   c. The inverse property is needed for creating a restriction. This commonly arises when using the mediating pattern.

When there is no named inverse:

1. Because some tools require inverses for all properties, it is a good idea to have a special annotation to indicate what the name of the inverse property should be. This way, a tool can automatically create inverses as needed.
   a. `e.g. hasBorrower :inverseName "isBOrrowerOn"^^xsd:string`
   b. Does anyone know if there is a standard annotation for this? I just made one up.
2. When to add the inverse name annotation? It is often difficult to think of a good name for an inverse, and it may not be worth the effort to think up names for every property. If we go that route, then the policy would be to indicate a name if one can be thought of with a minimum amount of thought, and other wise forget about it.
3. Automatically creating inverses
   a. There is a possibility of a name collision using this annotation, the algorithm would have to check for that some how.
   b. In the absence of an annotation, some convention can be used to create one. For a property 'p', examples are:
      i. ^p (if that character is allowed)
      ii. p_inv
      iii. p_inverse

iv.inverse_of_p

9. Cardinality Restrictions

In the event that Pellet throws an error, (e.g. if the property is transitive), options include:

1. use a fibo-fnd-utl-av:**usageNote** annotation.
2. use an alternative property that is not transitive (e.g. use non-transitive **comprises** which means approx same as transitive **hasPart**

Use a **min 0** restriction to say that a property is important for a class but is not always used. This is logically meaningless, but is picked up by tools.

10. Alternative Labels

Do not use altLabel, use synonym or abbreviation. Abbreviation should be a subproperty of synonym

11. isDefinedBy

FIBO will use isDefinedBy to identify what ontology an entity is defined in.

See: Metadata Recommendations For Linked Open Data Vocabularies. It is also recommended in section 4.6 of the Cool URIs paper.

12. email address policy for multiple applications

FIBO policy is the same email address for each person needing to be a JIRA/Confluence user and a user of the No Magic tools. To change their email address in JIRA, users must go to https://id.atlassian.com/manage/change-email.

13. All URI's should be resolvable

14. Release notes will be generated automatically from GitHub Pull requests. This requires that each person clearly documents the effect of each Pull Request.

15. Each FCT will deal with the circularities in their work. Circularity is one of the FIBO automated hygiene tests.