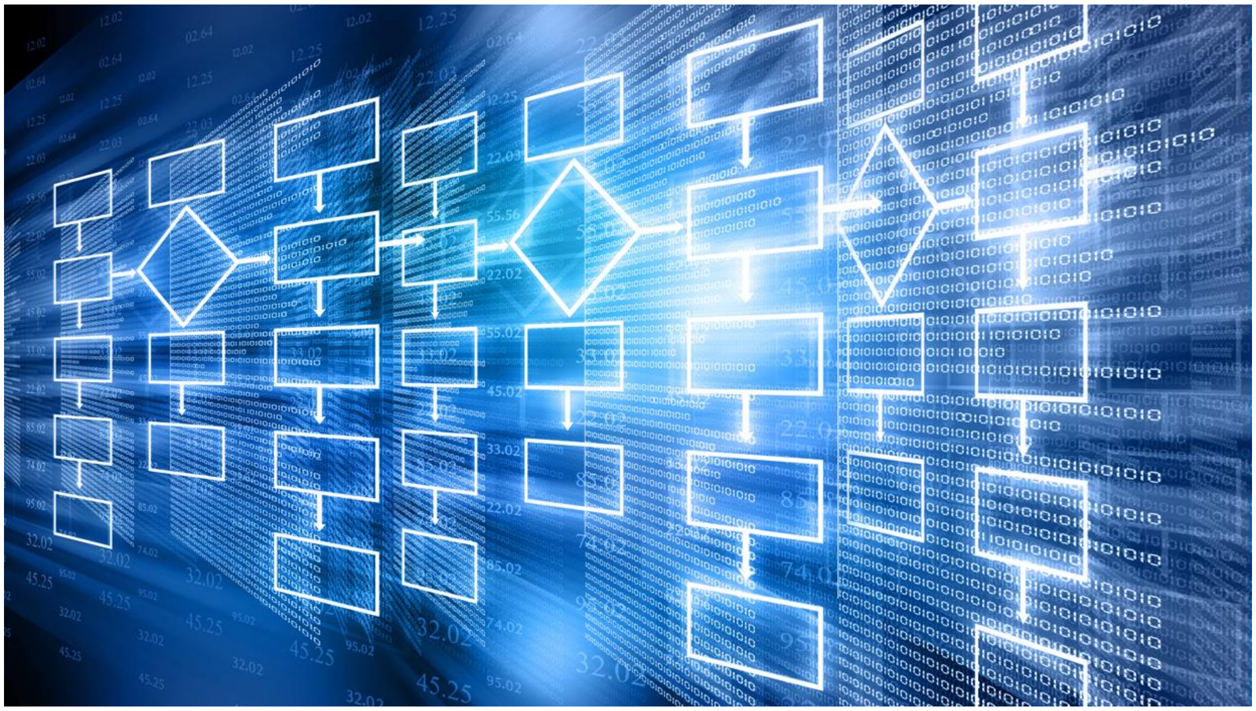
XODEL Language Syntax for XBRL - V1.0

Version 1.0



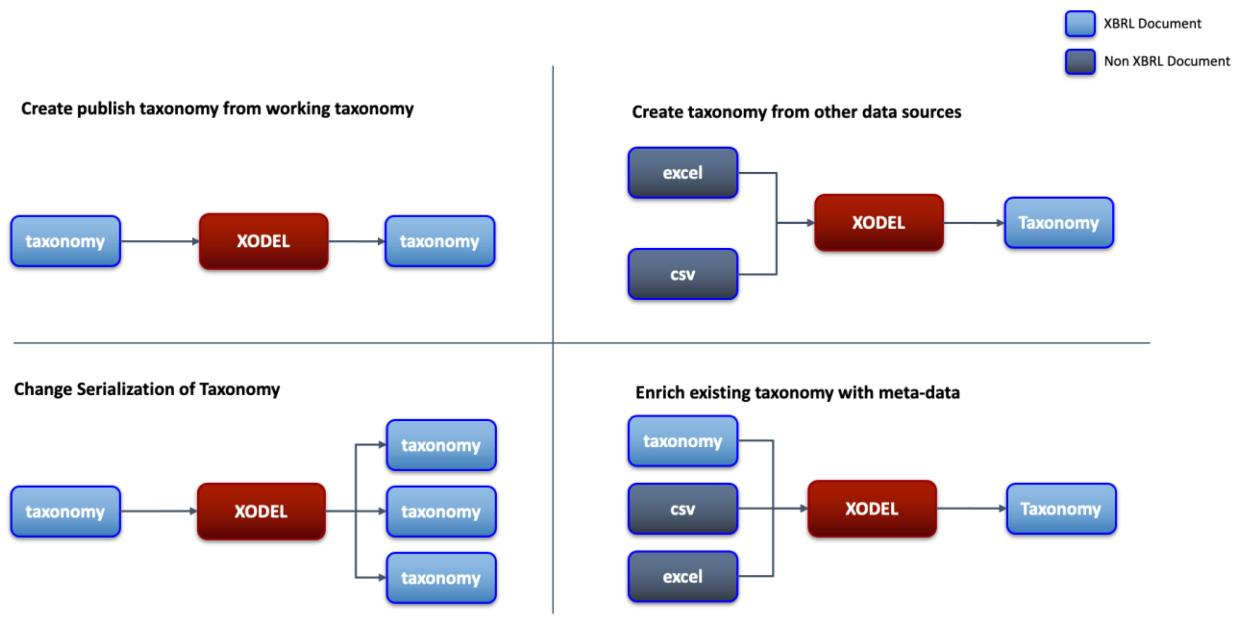
Overview

The XODEL syntax is a domain specific language used to define and create XBRL taxonomies. The XODEL language uses the XULE syntax to define taxonomy objects.

XODEL allows the creation of taxonomies using existing taxonomies, data in files such as spreadsheets, CSV and other data formats. XODEL allows metadata from various sources to be pulled together and output as a valid XBRL taxonomy. XODEL allows the user to control the serialization of the taxonomy, define various entry points and automate the creation of the taxonomy package.

The primary purpose of XODEL is to allow the automation of processes that are performed manually. Specific examples include adding references and labels to a taxonomy. Defining relationships between concepts, building hypercubes and defining individual elements. XODEL also allows the serialization of the taxonomy to be controlled and changed as needed. This means taxonomies can be updated and overhauled quickly and efficiently. It also means that multiple taxonomies can be created simultaneously. This is extremely useful for the creation of test cases that need custom extension taxonomies to be built. This is a time consuming and tedious process using existing tools. The diagram below shows how different components can be combined to create taxonomies.

Taxonomy Creation Components


XBRL Document
Non XBRL Document
Create publish taxonomy from working taxonomy
taxonomy
XODEL
taxonomy
Create taxonomy from other data sources
excel
csv
XODEL
Taxonomy
Change Serialization of Taxonomy
taxonomy
XODEL
taxonomy
taxonomy
taxonomy
Enrich existing taxonomy with meta-data
taxonomy
csv
excel
XODEL
Taxonomy

XODEL allows the definition of the following XBRL objects:

1. Relationships
2. Concepts
3. Types
4. Hypercubes
5. Labels
6. References
7. Reference Parts
8. Custom arcroles
9. Extended Link Roles
10. Label Roles
11. Reference Roles

XODEL Ruleset Definition

Output Attributes

The output attributes define the taxonomy components that are generated. The output attributes can use variables defined as part of the output body or constants

The XODEL syntax defines a number of XULE output attributes that are used to control the content of the taxonomy created.

The standard output attributes understood by XODEL are as follows:

- Packages
- Documents
- Roles
- Arcroles
- Relationships
- Concepts
- Labels
- References
- Networks
- Types

Output Attribute Definitions

These need to be defined as part of the XODEL ruleset. These are defined as follows:

Package

Output Attribute	Definition	Examples
`package-name`	The name of the taxonomy package being defined. Artifacts being created have to be assigned to a taxonomy package. This attribute is required on **every** rule.	`package-name 'My Taxonomy'`
`package-url`	The actual location of the package.	`package-url 'https://taxonomies.xbrl.us/xodel/MyTaxonomy'`
`package-entry-point`	A list of the entry points associated with the package defined as a list of relative URI. Entry points can also be added by flagging that a schema is an entry point when creating a schema document. This uses the attribute `document-package-entry-point`	`package-entry-point list('MyTaxonomy.xsd')`
`package-namespace-start-part`	Used if a namespace is not defined for a document created as part of the package.	`package-namespace-start-part 'https://taxonomies.xbrl.us/xodel/MyTaxonomy'`

Documents

Allows the creation and definition of documents for serialization of the taxonomy package. If no documents are defined XODEL will create a default serialization of the files. The serialization will create:

- A schema file for each namespace where named items are defined (i.e. concepts, types, reference parts)
- A schema file for defined roles

- A schema file for defined arcroles
- An entry point schema file
- Separate files for labels, references, presentation, calculator, definition, generic relationships (the linkbases).

The files will be named using package name.

Output Attribute	Definition	Examples
`document-uri`	A string defining the url of the document to be created. This can be a uri relative to the package or an absolute URI.	**`document-uri`** `'elts/concepts.xsd'`
`document-namespace`	A url that represents the namespace of the document. This is not required if a concept is defined in the schema, by default XODEL will take the namespace of the concepts defined in the schema file.	**`document-namespace`** `'http://myorg.org/ll-entry-point/'`
`document-import`	A set or list of document URIs or a single URI that adds import/linkbase ref statements to a specific document. Must have the document uri so the processor knows the file it is adding the import to. The imported document uri can be relative to the package or an absolute uri.	**`document-import`** `set('elts/concepts.xsd', 'elts/labels.xml')`
`document-imported-in`	A list or set of document URIs or a single URI. Updates the identified document so that it imports/linkbase refs to the current document.	**`document-imported-in`** `list('schedule/{get_sched_name($network)}/schedule-{get_name($network.role)}.xsd')`
`document-package-entry-point`	Boolean that indicates if this file is included as a package entry point.	**`document-package-entry-point`** `true`
`document-package-entry-point-description`	The description of the entry point.	**`document-package-entry-point`** `"Main entry point"`

Concept

Allows the creation of XBRL concepts. The following attributes allow concepts to be copied from an existing taxonomy or to define new concepts.

Output Attribute	Definition	Examples
concept	Used to pass a concept object usually from another taxonomy.	**concept** $concept
concept-name	Defines the concept name. If a variable is not used the qname is provided with the full namespace in curly brackets. The curly brackets must be escaped with a backslash.	**concept-name** '\{http://fasb.org/us-gaap/2022\}Assets'
concept-local-name	Provide a new or updated local name for a concept as a string value.	**concept-name** 'Assets2'
concept-namespace	Provide a new or updated namespace for a concept as a string value.	**concept-namespace** 'http://fasb.org/us-gaap/2024'
concept-data-type	Defines the datatype of the concept. This is provided as a qname with the full namespace in curly brackets.	**concept-data-type** '\{http://fasb.org/us-gaap/2022\}PosInteger'
concept-abstract	Provides a boolean value as true to represent an abstract and false to be non abstract	**concept-abstract** false
concept-nillable	Provides a boolean value as true to represent a nillable and false to be non abstract	**concept-nillable** false
concept-period-type	Provide a string to indicate if the period type is duration or instant	**concept-period-type** instant
concept-balance-type	Provide a string to indicate if the balance is a debit or a credit. This attribute can	**concept-balance-type** debit

	be excluded to indicate no balance type or an empty string.	
concept-substitution-group	Indicates the substitution group as a qname.	**concept-substitution-group** '\{http://xbrl.org\}:item
concept-attributes	Add an attribute or multiple attributes to a concept. This requires that a dictionary is provided with the attribute and the value.	**concept-attributes** dict(list('myattribute', 'the best value'),list('mySecondAttribute', 'The second best value'))

Role

Output Attribute	Definition	Examples
role	Used to pass a role object usually from another taxonomy. Any other role attributes defined in the rule will override the value of the role object.	**role** $role
role-uri	The role uri used by the role. This is provided as a string.	**role-uri** 'https://my.role.uri/2034/BalanceSheet'
role-definition	The role definition provided as a string.	**role-definition** '100330 - Statement - Balance Sheet'
role-used-on	Indicates the linkbase types that the role is used on. This is provided as a list. Can contain one of the built in links: presentation, calculation, definition, generic, label, reference or a qname.	**role-used-on** list('Presentation', 'Calculation', 'Definition')

Arcrole

Output Attribute	Definition	Examples
arcrole	Used to pass an arcrole object usually from another taxonomy.	**arcrole** $arcrole
arcrole-uri	A uri string of the arcrole.	**arcrole-uri** 'http://www.xbrl.org/2003/arcrole/parent-child'
arcrole-description	A string representing a description of the arcrole.	**arcrole-description** 'Parent/Child relationship'
arcrole-cycles-allowed	A string value of 'any', 'undirected' or 'none'	**Arcrole-cycles-allowed** 'none'

Label

Output Attribute	Definition	Examples
label	Used to pass a label object usually from another taxonomy.	**label** $label
label-text	A string value representing the text of the label.	**label-text** 'Assets'
label-lang	A string value using a valid iso code for the language of the label.	**label-lang** 'en-us'
label-role	The standard label uri used to indicate the type of label. Such as total, documentation verbose etc.	**label-role** 'http://www.xbrl.org/2003/role/label'
label-name	A unique string value used to identify the label.	**label-name** 'standardLabelForAssets'
label-concept-name	The qname of the concept that the label applies to.	**label-concept-name** Assets

Reference

Output Attribute	Definition	Examples
`reference`	Used to pass a reference object usually from another taxonomy.	**`reference`** `$ref`
`reference-role`	The reference role uri that indicates what type of guidance it is such as a Disclosure Reference or Presentation reference	**`Reference-role`** `'http://www.xbrl.org/2003/role/presentationRef'`
`reference-name`	A string value used to associate the reference with the reference parts	**`reference-name`** `'SECReferenceForAssets'`
`reference-concept-name`	The qname of the concept that the label applies to.	**`reference-concept-name`** `Assets`

Reference Parts

Output Attribute	Definition	Examples
`reference-part`	A reference part object	**`reference-part`** `$partObject`
`reference-part-value`	A string value representing the part value	**`reference-part-value`** `810`
`part-qname`	The qname of the part such as Topic, Subject, Publisher etc.	**`part-qname`** `fasb:Topic`
`reference-part-order`	A decimal representing the order of the part in the reference.	**`Reference-part-order`** `1`
`reference-name`	A string used to associate the reference part with the reference	**`reference-name`** `'SECReferenceForAssets'`

Part

The part object allows the definition of a new part such as Topic, Paragraph etc. XODEL allows the user to define new part names that can be used in references.

Output Attribute	Definition	Examples
`part`	A part object	**part** $part
`part-qname`	A qname representing the name of the part.	**`part-qname`** `xbrlus:FormLocation`
`part-type`	A qname that indicates the part type	**`part-type`** `xsd:string.to-xodel`

Relationship

Output Attribute	Definition	Examples
`relationship`	A relationship object	**`relationship`** `$relationship`
`relationship-source`	A qname or a concept object that represents the source of the relationship.	**`relationship-source`** `Assets`
`relationship-target`	A qname or a concept object that represents the target of the relationship.	**`relationship-target`** `AssetsCurrent`
`relationship-order`	A decimal representing the order of the relationship. If not provided the value is created as a sequence on the order of relationship creation.	**`relationship-order 1`**
`relationship-weight`	An integer representing the order of the calculation relationship.	**`relationship-weight 1`**
`relationship-preferred-label`	The uri of the preferred label to use on a presentationship relationship.	**`relationship-preferred-label`** `'http://www.xbrl.org/2003/role/label'`
`relationship-role`	The extended link role uri of	**`relationship-role`**

	the role the relationship appears in.	`'http://www.abc.com/role/CONSOLIDATEDBALANCESHEET'`
`relationship-type`	The relationship type represented as a string of presentation, calculation, definition, generic	**`relationship-type`** `presentation`
`relationship-arcrole`	A uri or a short arcrole name used for the relationship such as parent-child or summation-item	**`relationship-arcrole`** `parent-child`
`relationship-attributes`	A dictionary that allows the definition of additional attributes on an arc. The dictionary key as a qname representing the attribute of the arc.	
`relationship-name`	A unique id of the relationship used to associate the relationship with a network	

Type

Output Attribute	Definition	Examples
`type`	A type object	**`type`** `$type`
`type-name`	The qname of the type being defined.	**`type-name`** myorg:myItemType
`type-parent`	The type of the parent type.	**`Type-parent $type`**
`type-parent-name`	The qname of the parent type.	**`type-parent-name`** xbrli:string
`type-min-inclusive`	A decimal value to indicate a min inclusive cardinal value for a type.	**`type-min-inclusive`** `1`
`type-max-inclusive`	A decimal value to indicate a max inclusive cardinal value for a type.	**`Type-max-inclusive`** `10`

`type-min-exclusive`	A decimal value to indicate a min exclusive cardinal value for a type.	**`Type-min-exclusive`** `1`
`type-max-exclusive`	A decimal value to indicate a max exclusive cardinal value for a type.	**`Type-max-exclusive`** `10`
`type-total-digits`	A decimal value to indicate total digits of a value.	**`Type-total-digits`** `6`
`type-fraction-digits`	integer	**`Type-fraction-digits`** `3`
`type-length`	Integer value used to define the length of a string value.	**`Type-length`** `4`
`type-min-length`	Integer value used to define minimum length of a string value.	**`Type-min-length`** `2`
`type-max-length`	Integer value used to define maximum length of a string value.	**`Type-max-length`** `4`
`type-enumerations`	list/set of strings	**`Type-enumerations`** `list('a','b','c')`
`type-white-space`	one of 'preserve', 'replace' or 'collapse'	**`Type-white-space`** `'preserve'`
`type-pattern`	A single regex expression or a list/set of regex expressions	

Cube

Output Attribute	Definition	Examples
cube-from-presentation	Allows the definition of cubes based on the presentation linkbase in the taxonomy. Must provide the options of ‘us-gaap’ or ‘ifrs’. In us-gaap taxonomy presentations the line items comprising a cube appear as children of the hypercubeItem. In IFRS	**`Cube-from-presentation`** `'us-gaap'`

	taxonomies they appear as siblings of the hypercubeItem.	
cube-from-presentation-role	Defines the roles that will be defined as a definition linkbase. The role is defined as the parameter.	

Network Object (Relationship Group)

Output Attribute	Definition	Examples
`network`	A network object. This is a convenience to save copying all the relationships in a network. This allows you to copy a network from an existing taxonomy into a new taxonomy.	`network $network`
`network-name`	An identifier defined as a string used to associate the network with a package or document	

Namespace Map

A namespace map allows adding a prefix, namespace and schema location. This is necessary when using non standard XBRL components such as extended link elements and arc elements.

Output Attribute	Definition	Examples
`namespace-map`	A list of 1 to 3 items. 1. Namespace URI 2. Namespace prefix 3. Schema location The first item is required. The prefix and the schemalocation are optional. If only the location is needed, none should be used for the prefix	namespace-map list('http://mydomain.com/myNamespace', 'mn', 'http://mydomain.com/mySchema.xsd')

namespace-maps	A list that contains namespace-map lists as described above. This is a way to set multiple namespace maps within a single rule.	namespace-maps list(ist('http://mydomain.com/myNamespace1', 'mn1', 'http://mydomain.com/mySchema1.xsd'),ist('http://mydomain.com/myNamespace2', 'mn2', 'http://mydomain.com/mySchema2.xsd'))

Duplicate namespace maps are allowed as long as they are complete duplicates. That is they have the same namespace uri, prefix and location. Otherwise duplicate namespace uris or duplicate prefixes will be flagged as an error.

Examples

Concept Creation from Excel

Define concepts based on an excel file.

```
constant $conceptsList = csv-
data("https://mydata.com/files/data.csv",false,list('qname','string','string','string'))

output concepts
for $concept in $conceptsList
      $conceptName = $concept[1]
      $conceptPeriod = $concept[2]
      $conceptDataType = $concept[3]
      $conceptAbstract = false
      $conceptBalance = $concept[4]

concept-name $conceptName.to-xodel
package-name 'My Taxonomy'
concept-data-type $conceptDataType.to-xodel
concept-abstract $conceptAbstract.to-xodel
concept-nillable true.to-xodel
concept-period-type $conceptPeriod.to-xodel
concept-balance-type $conceptBalance.to-xodel
concept-substitution-group xbrli:item.to-xodel
```

Rollover an Existing Package to a Subsequent Taxonomy

This rule takes a taxonomy and updates it to a subsequent year. This requires updating the namespace of the concepts and the version name of the files.

```
constant $ALL_CONCEPTS = taxonomy(https://www.zz.org/zz-gaap-
2023.xsd).concepts
constant $PUBLISH_TAXONOMY = 'MyTaxonomy'
constant $VERSION = '2024-01-01'

output concepts
   for $concept in $ALL_CONCEPTS
     $concept
package-name $PUBLISH_TAXONOMY
concept $concept
concept-name qname('https://www.zz.org/zz-gaap-2024/', $concept.name.local-
name)
document-uri 'elts/concepts-{$version}.xsd'

output relationships
   for $network in filter taxonomy().networks where $item
              for $relationship in $network.relationships
                     $relationship
package-name $PUBLISH_TAXONOMY
relationship $relationship
relationship-source qname('https://www.zz.org/zz-gaap-2024/',
$relationship.source-name.local-name)
relationship-target qname('https://www.zz.org/zz-gaap-2024/',
$relationship.target-name.local-name)
document-uri 'schedule/{get_sched_name($network)}/schedule-
{get_name($network)}-{get_link_type($network)}-{$version}.xml'

output labels
   for $concept in $ALL_CONCEPTS
     for $lab in $concept.all-labels
              $lab
package-name $PUBLISH_TAXONOMY
label $lab
label-concept-name qname('https://www.zz.org/zz-gaap-2024/',
$relationship.source-name.local-name)
document-uri 'elts/labels.xml'

output refs
   for $concept in $ALL_CONCEPTS
```

```
     for $ref in $concept.all-references
             $ref
package-name $PUBLISH_TAXONOMY
reference $ref
reference-concept-name qname('https://www.zz.org/zz-gaap-2024/',
$relationship.source-name.local-name)
document-uri 'elts/references.xml'
```

Creating a Package from an Existing Taxonomy

This shows how a taxonomy contained in 4 files can be serialized to have a separate schema file for each network defined in the taxonomy. We have a variable called $ALL_CONCEPTS that represents a set of concepts extracted from an existing taxonomy. We can loop through these concepts and put them in a new taxonomy package. In this case we add them to $publish-taxonomy. We define the document schema file as 'elts/concepts.xsd' using the attribute document-uri. We can also indicate that this is an entry point for the taxonomy package using document-package-entry-point. No target namespace is provided as this is picked up automatically from the namespace of the concepts.

```
constant $ALL_CONCEPTS = taxonomy().concepts
constant $PUBLISH_TAXONOMY = 'MyTaxonomy'

output concepts
   for $concept in $ALL_CONCEPTS
     $concept
package-name $PUBLISH_TAXONOMY
concept $concept
document-uri 'elts/concepts.xsd'
```

To add labels we can go through the concepts and pick out the labels we want to add to the new taxonomy.

```
output labels
   for $concept in $ALL_CONCEPTS
     for $lab in $concept.all-labels
             $lab
package-name $PUBLISH_TAXONOMY
label $lab
document-uri 'elts/labels.xml'
```

Now we have created the concepts and the labels we want to copy networks from our existing taxonomy. We want to create a separate schema file and relationship file for each network. First we create the schema files.

We create a schema document for each network to import the concepts and labels. To do this we add import statements to the schema file. The schema file name is a derivative of the network name.

```
output schedule_schema_files
   for $network in taxonomy().networks
            $network
package-name $PUBLISH_TAXONOMY
document-uri 'schedule/{get_sched_name($network)}/schedule-
{get_name($network.role)}.xsd'
Document-namespace
'http://taxonomies.xbrl.us/xodel/schedule/{get_sched_name($network)}
document-import list('elts/concepts.xsd', 'elts/lables.xml')
```

Next we create the presentation, calculation and definition linkbase files, by reading the networks.

```
output network_files
   for $network in taxonomy().networks
            $network
package-name $PUBLISH_TAXONOMY
network $network
document-uri 'schedule/{get_sched_name($network)}/schedule-
{get_name($network)}-{get_link_type($network)}.xml'
document-imported-in 'schedule/{get_sched_name($network)}/schedule-
{get_name($network.role)}.xsd'
```

At this point we still do not have an entry point for the package. To add two entry points, such as the elements and labels and one with all the networks we can do the following.

```
output entry-point1
   True
package-name $PUBLISH_TAXONOMY
document-uri 'all-elts-entry-point.xsd'
document-namespace 'http://taxonomies.xbrl.us/xodel/all-elts-entry-point/'
document-import list('elts/concepts.xsd', 'elts/lables.xml')
document-package-entry-point true
```

To bring all the components together the `all-entry-point.xsd` entry point file is defined as follows:.

```
output entry-point2
```

```
   $import-schemas = set(for $role in taxonomy().roles
       'schedule/' + get_sched_name($role) + '/schedule-' + get_name($role)' +
'.xsd')
       true
package-name $PUBLISH_TAXONOMY
document-uri 'all-entry-point.xsd'
document-namespace 'http://taxonomies.xbrl.us/xodel/all-entry-point/'
document-import $import-schemas
document-package-entry-point true
```

The rule builds up a set of the import schemas that are passed to the document-import attribute. The import schemas are generated by looping through all the taxonomy roles. A set is used so that duplicate schema files are not created.

Create An Extension Taxonomy

To create an extension taxonomy for abc company we perform the following steps:

1. Import the us-gaap, srt, country and dei elements taxonomies
2. Define company specific elements
3. Define company specific labels for the elements used
4. Define labels for base concepts
5. Define roles
6. Define the presentation linkbase
7. Define the calculation linkbase
8. Define hypercubes based on the presentation

Import the Base Taxonomies

```
constant $EXTENSION_TAXONOMY = 'MyTaxonomy'

output import_base_taxonomy
true
package-name $EXTENSION_TAXONOMY
package-url 'https://taxonomies.xbrl.us/xodel/MyTaxonomy'
document-uri abc.xsd'
document-import set('https://fasb.org/elts/us-gaap-2023.xsd',
'https://fasb.org/elts/us-srt-2023.xsd', 'https://sec.gov/elts/dei-2023.xsd')
```

Define Extension Concepts

```
constant $EXTENSION_CONCEPTS = list(
list('CarpetLoans',label,'Carpet Loans'),
list('CarpetLoans','documentation','Loans to customers to buy carpet'),
list('LoomEquipment','label','Loom Equipment'),
list('LoomEquipment','documentation','Equipment that makes carpet'),
list('WoolStock','label','Wool Stock'),
list('WoolStock','documentation','Wool in inventory to make carpet'))

output create_extension_concept
$base_asset = taxonomy().concept(Assets)
$UniqueAssets = set(filter $EXTENSION_CONCEPTS returns $item[1])
for $asset in $UniqueAssets
      $asset

package-name $EXTENSION_TAXONOMY
document-uri 'abc.xsd'
concept $base_asset.to-xodel
concept-namespace 'https://abc.com/2024'
concept-local-name $rule-value
```

Define Company Specific Labels

```
output create_extension_labels
for $label in $EXTENSION_CONCEPTS
            $label
package-name $EXTENSION_TAXONOMY
document-uri 'abc_lab.xml'
label-concept-name qname('https://abc.com/2024', $label[1]).to-xodel
label-text $label[3]
label-role 'http://www.xbrl.org/2003/role/' + $label[2]
```

Define Base Concept Labels

```
constant $BASE_LABELS = list(
list(Assets,'standard','Assets'),
list(Liabilities,'standard','Liabilities'),
list(LiabilitiesCurrent,'standard','Current Liabilities'))

output create_base_labels
for $base in $BASE_LABELS
       $base
package-name $EXTENSION_TAXONOMY
document-uri 'abc_lab.xml'
label-concept-name $base[1].to-xodel
```

```
label-text $base[3]
label-role 'http://www.xbrl.org/2003/role/' + $base[2]
```

Define Balance Sheet Role

```
output BalanceSheet_Role
      true
package-name $EXTENSION_TAXONOMY
document-uri 'abc.xsd'
role-uri 'http://www.abc.com/role/CONSOLIDATEDBALANCESHEET'
role-description '100 - Statement - Balance Sheet (Consolidated)'
role-used-on list('Presentation', 'Calculation','Definition')
role-name '100-Balance'
```

Define Presentation Balance Sheet

```
constant $BALANCE_SHEET_NETWORK = first(FILTER taxonomy().networks(parent-
child) where $item.role.description.contains('104000 - Statement -'))

constant $BASE_ELEMENTS = FILTER $BASE_LABELS returns $item[1]

output BalanceSheet_Presentation
$relationships = navigate parent-child ancestors from $BASE_ELEMENTS role
$BALANCE_SHEET_NETWORK.role returns relationship
for $pres_rel in $relationships.to-set
            $pres_rel
package-name $EXTENSION_TAXONOMY
document-uri 'abc_pre.xml'
relationship $pres_rel.to-xodel
relationship-role 'http://www.abc.com/role/CONSOLIDATEDBALANCESHEET'

output BalanceSheet_Presentation_Extensions
$extensionConcepts = set(for $item in $EXTENSION_CONCEPTS
            $item[1])
for $concept in $extensionConcepts
      $concept
package-name $EXTENSION_TAXONOMY
document-uri 'abc_pre.xml'
relationship-target qname('https://abc.com/2024, $concept).to-xodel
relationship-source AssetsAbstract
relationship-role 'http://www.abc.com/role/CONSOLIDATEDBALANCESHEET'
relationship-type 'presentation'
```

```
relationship-arcrole 'parent-child'
relationship-order $concept.length
```

Define Calculation Balance Sheet

```
output BalanceSheet_Calculation

$extensionConcepts = set(for $item in $EXTENSION_CONCEPTS
            $item[1])
for $concept in $extensionConcepts
      $concept
package-name $EXTENSION_TAXONOMY
document-uri 'abc_cal.xml'
relationship-target qname(‘https://abc.com/2024, $concept).to-xodel
relationship-source Assets.to-xodel
relationship-role 'http://www.abc.com/role/CONSOLIDATEDBALANCESHEET'
relationship-type 'calculation'
relationship-weight 1
```