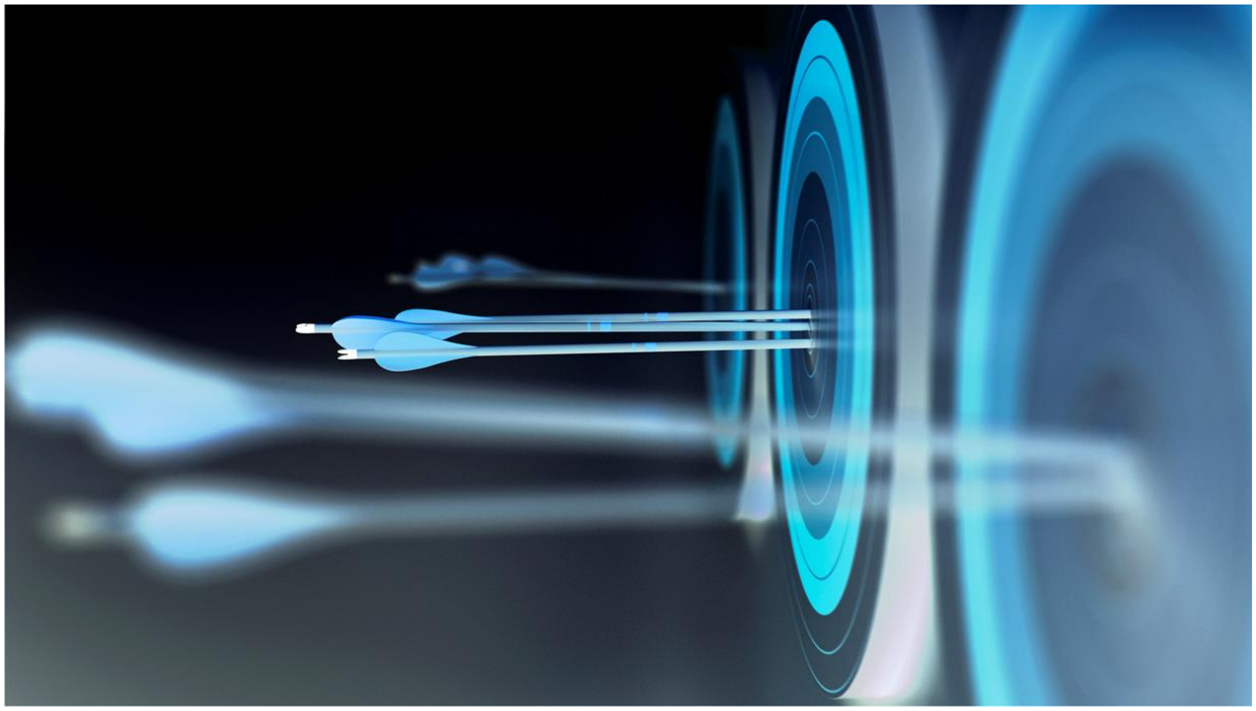
XINCE Language Syntax for XBRL - V1.0

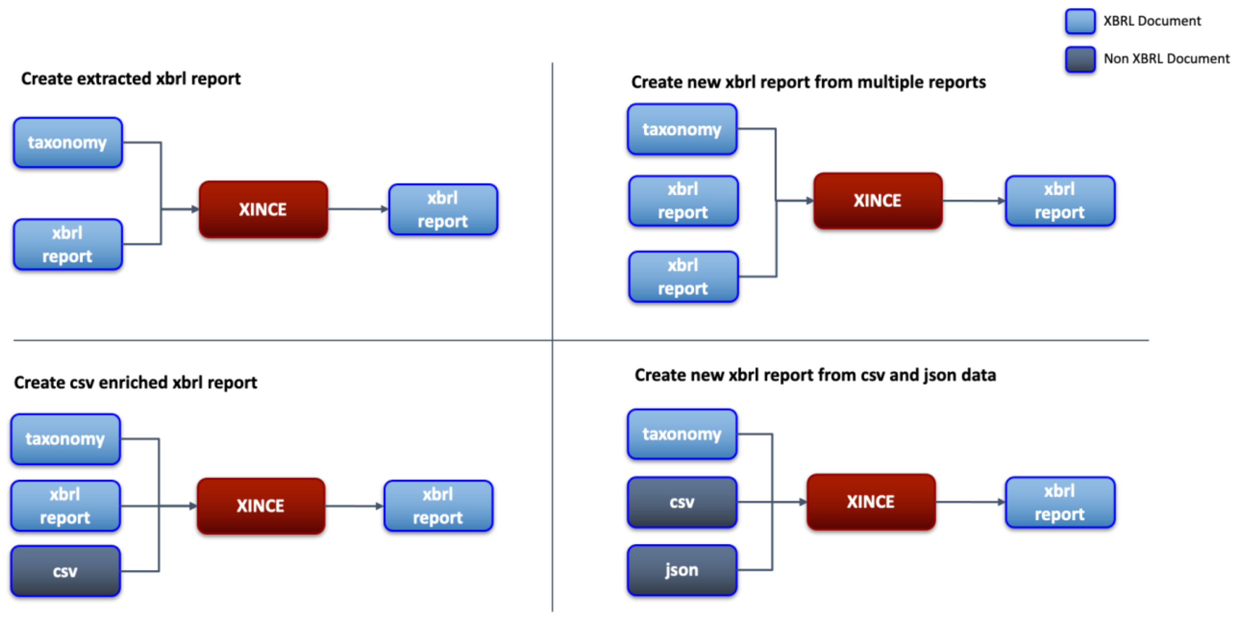
Version 1.0



Overview

The XINCE syntax is a domain specific language used to define and create XBRL instances. The XINCE language uses the XULE syntax to manipulate facts prior to outputting XBRL instances in a JSON or XML format. The diagram below shows how different components can be combined to create instance documents.

Instance Creation Components



XINCE Ruleset Definition

The XINCE ruleset definition is the same as a XULE ruleset. It is a collection of XULE files that defines the expressions that will create the instance document or documents. These files will generally include the following:

- Namespace definitions
- Output-attribute definitions
- Constant definitions
- Output statements

Namespace Definitions

These are regular XULE namespace definitions. The namespaces of any qnames defined in the output statements need to be defined. In addition, namespaces of facts that will be generated by XINCE also need to be defined.

Output Attribute Definitions

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

```
output-attribute instance-name
output-attribute instance-taxonomy
output-attribute fact-value
output-attribute fact-concept
output-attribute fact-unit
output-attribute fact-entity
output-attribute fact-period
output-attribute fact-decimals
output-attribute fact-dimensions
output-attribute fact-add-dimension
output-attribute fact-remove dimension
output-attribute fact-instance
output-attribute fact-alignment
output-attribute fact-footnote
output-attribute fact-id
output-attribute fact-is-nil
```

Output Statements

Output Header

The instance or instances are defined by consolidating a series of output statements. Each output statement has a name that defines the output. All of the output statements are run. At the completion of processing the output statements an instance or instances are produced.

Output Body

The body of an output statement is a XULE expression that returns values or facts that can be passed to the output attributes. The output body has to create an iteration to create a fact. This can be achieved by defining a fact set or a for loop. Each iteration of the expression will generate a fact. The output body can use any XULE expression defined in the XULE syntax.

Output Attributes

The output attributes define the facts that are generated by the iteration. The output attributes can use variables defined as part of the output body, constants or a special function called alignment().

The alignment function returns the alignment of the fact. When used with the fact-alignment attribute the alignment of the iteration is applied to the fact. This means the individual dimensional values of each fact do not have to be defined.

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

The standard output attributes understood by XINCE are as follows:

- instance-name
- instance-taxonomy
- fact-value
- fact-concept
- fact-unit
- fact-entity
- fact-period
- fact-decimals
- fact-dimensions
- fact-add-dimensions
- fact-remove-dimension
- fact-instance
- fact-alignment
- fact-footnote
- fact-id
- fact-is-nil

Output Attribute	Definition	Examples
instance-name	A text string that defines the name of the instance. This can be a variable or a string. It must be unique	`instance-name  'My Instance'`
instance-taxonomy	The taxonomy used by the instance. This can be a single taxonomy or a list of taxonomies. This can only be used with instance-name.	`instance-taxonomy list('https://www.sec.gov/Archives/edgar/data/891166/000089116622000114/uve-20220930.xsd', 'abc.xsd').to-json`
fact-value	Defines the value of the fact.	**`fact-value`** `123000000`

fact-concept	Defines the concept used by the fact. 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.	`fact-concept '\{http://fasb.org/us-gaap/2022\}Assets'`

fact-unit	Defines the unit associated with the fact.	**`fact-unit`** `unit(iso4217:USD).to-xince`
		
		Assigns a unit of USD on the fact
		
		`fact-unit $rule-value.unit.to-xince`
		Assigns the unit of the rule fact

fact-entity	Defines the entity associated with the fact.	**`fact-entity`** `entity('http://some/schema', 'CompanyA').to-xince`
fact-period	Defines the period associated with the fact.	`fact-period date('2022-12-31').to-xince`
fact-decimals	Defines the decimals associated with the fact.	`fact-decimals -6`

fact-dimensions	Define a dictionary of axis member pairs	`fact-dimensions $fact.dimensions.to-xince`
		
		Assigns the dimensions of the $fact variable
		
		`fact-dimensions dict(list("{http://fasb.org/us-gaap/2022}StatementEquityComponentsAxis”,"{ http://fasb.org/us-gaap/2022}CommonStockMember")).to-xince`
		
		Assigns the dimension StatementEquityComponentsAxis and member CommonStockMember to the fact.
		
		`fact-dimensions dict(list(StatementEquityComponentsAxis,CommonStockMember")).to-xince`

fact-add-dimension	Add an additional dimension to the dictionary.	`fact-add-dimension dict(list(StatementEquityComponentsAxis,CommonStockMember")).to-xince` **Adds an additional dimension member pair to the fact.**
fact-remove-dimension	Remove a dimension from an existing dictionary of dimensions	`fact-remove-dimension dict(list(StatementEquityComponentsAxis,CommonStockMember")).to-xince` **Removes the dimension member pair from the fact if it exists.**
fact-instance	Assigns a fact to a specific instance based on the instance name. The fact of the iteration is assigned to the instance.	`fact-instance 'My Instance'`
fact-alignment	Allows the alignment to be set based on the fact iteration. This copies all of the alignment from the current iteration onto the fact. Any alignment can be overwritten by the other output attributes listed above. The value is a dictionary. The value can also be set using the alignment() function, which is a json string.	`fact-alignment alignment().to-xince` **Assigns the iteration alignment to the fact.**

fact-footnote	Defines the footnote associated with the fact. Can be passed as the footnote property or as a dictionary.	`fact-footnote $fact.footnote`
		
		Copies the footnote from the original fact to the new instance fact.
		
		`fact-footnote list(dict(list('lang', 'en-US'),list('arcrole', 'footnote'),list('content', 'hello'))).to-xince`
		
		Assigns the defined footnote with the content 'hello' to the generated fact.

fact-id	Define a fact id for a fact. If a fact id is duplicated, xince will increment that id by adding a number.	`fact-id 'abc'`

		Assigns a fact id of 'abc' to the fact.
fact-is-nil	Defines if the fact has a nil value or not. The value has to resolve to true or false.	`fact-is-nil $rule-value.is-nil` **Assigns a value of nil is true to the fact.**

Output attributes can include expressions. However output attributes cannot create iterations. This means fact set expressions cannot be defined in the output attribute. For loops can, as long as they are included within a set or a list.

Creating Facts

To create a fact the dimensions of the fact have to be passed to the fact using the output attributes. A single output rule can create many facts. Every fact that is created must specify the instance document that it belongs to.

XINCE created facts are sent to the output log. All values sent to the log are sent as a string. This means all XULE objects must be converted to a string representation when used with an output attribute. XINCE includes a property called to-xince, that will convert XULE objects to a string representation of the object that can be included in the output attribute.

Fact Generation Examples

Example 1

The following rule will take every monetary fact in an instance document, multiply the value by 10% and output the result as a new instance called myInstance.

```
output createInstance
true
instance-name  “myInstance”
instance-taxonomy
'https://www.sec.gov/Archives/edgar/data/891166/000089116622000114/uve-20220930.xsd'

output add_fact_values
{@ where $fact.is-monetary}
true
fact-value $rule-value * 1.1
fact-concept $rule-value.concept.to-xince
fact-unit $rule-value.unit.to-xince
fact-entity  $rule-value.entity.to-xince
```

```
fact-period  $rule-value.period.to-xince
fact-decimals  $rule-value.decimals
fact-dimensions  $rule-value.dimensions.to-xince
fact-instance "myInstance"
```

The rule will iterate through each fact value and multiply it by 1.1. All the dimensions have been explicitly stated. Rather than listing out all the dimensions of the fact the fact-alignment output attribute can be used with the alignment function to save listing out all the dimensional attributes. The instance expression above can be expressed as follows using alignment().

```
output add_fact_values
{@ where $fact.is-monetary}

fact-value $rule-value * 1.1
fact-alignment alignment().to-xince
fact-decimals  $rule-value.decimals
fact-instance "myInstance"
```

Decimals have to be defined as they do not comprise the fact alignment. If no fact decimal is provided it defaults to infinite.

Example 2

The following rule will take every monetary fact in an instance document, multiply the value by 20%, and increment all the period values by one year and output the result as a new instance called myInstance.

```
output add_instant_fact_values
{@ where $fact.is-numeric and $fact.concept.period-type == instant}

fact-value $rule-value * 1.2
fact-alignment alignment().to-xince
fact-period  ($rule-value.period.end + time-span('P1Y')).to-xince
fact-decimals  $rule-value.decimals
fact-instance "myInstance"

output add_duration_fact_values
{@ where $fact.concept.period-type == duration}

fact-value if $rule-value.concept.is-numeric  $rule-value * 1.2 else $rule-
value
fact-alignment alignment().to-xince
fact-period duration($rule-value.period.start + time-span('P1Y'),$rule-
value.period.end + time-span('P1Y')).to-xince
```

```
fact-decimals  $rule-value.decimals
fact-instance “myInstance”
```

If the instance taxonomy differs from one output rule to another then the processor will use the one it finds first.

Creating Footnotes

Xince allows the creation of footnotes in a new instance. This can be achieved by using the footnote on a fact in an instance that is being copied or by creating a new footnote. To include the footnote from a reference instance the attribute fact-footnote is used and the footnote property of the fact is assigned.

```
fact-footnote $fact.footnote
```

Alternatively one or more footnotes can be assigned to a fact using a list of dictionaries. Each footnote is defined as a dictionary collection with the following possible keys:

- role
- lang
- arcrole
- content
- id-ref

To define a footnote on a fact say hello the following expression is used:

```
fact-footnote list(dict(list('lang', 'en-US'),
                        list('arcrole', 'footnote'),
                        list('content', 'hello ' ))
                ).to-xince
```

To add multiple footnotes additional dict items are added to the list. In each of these examples the footnote is added at the same time as the fact is created.

Referencing Existing facts

Xince allows the creation of footnotes that are not only textual values, but also footnotes that reference other facts in the instance. This requires knowledge of the reference id of the fact being referenced. The syntax is as follows:

```
fact-footnote list(dict(list('role', 'link/role'),
                        list('arcrole', 'fact-footnote'),
                        list('id-ref', 'fact10'))
                  ).to-xince
```

Arelle Reference Implementation of XULE

Command Line Instructions

Command	Description
`--xince-location=Location of the folder`	Directory or folder where the generated instance file will be created.
`--xince-show-xule-log`	Indicates to output the xule log.
`--xince-file-type=XINCE_FILE_TYPE`	Used to define the type of instance output. Valid values are 'json' or 'xml'. The default format is json.