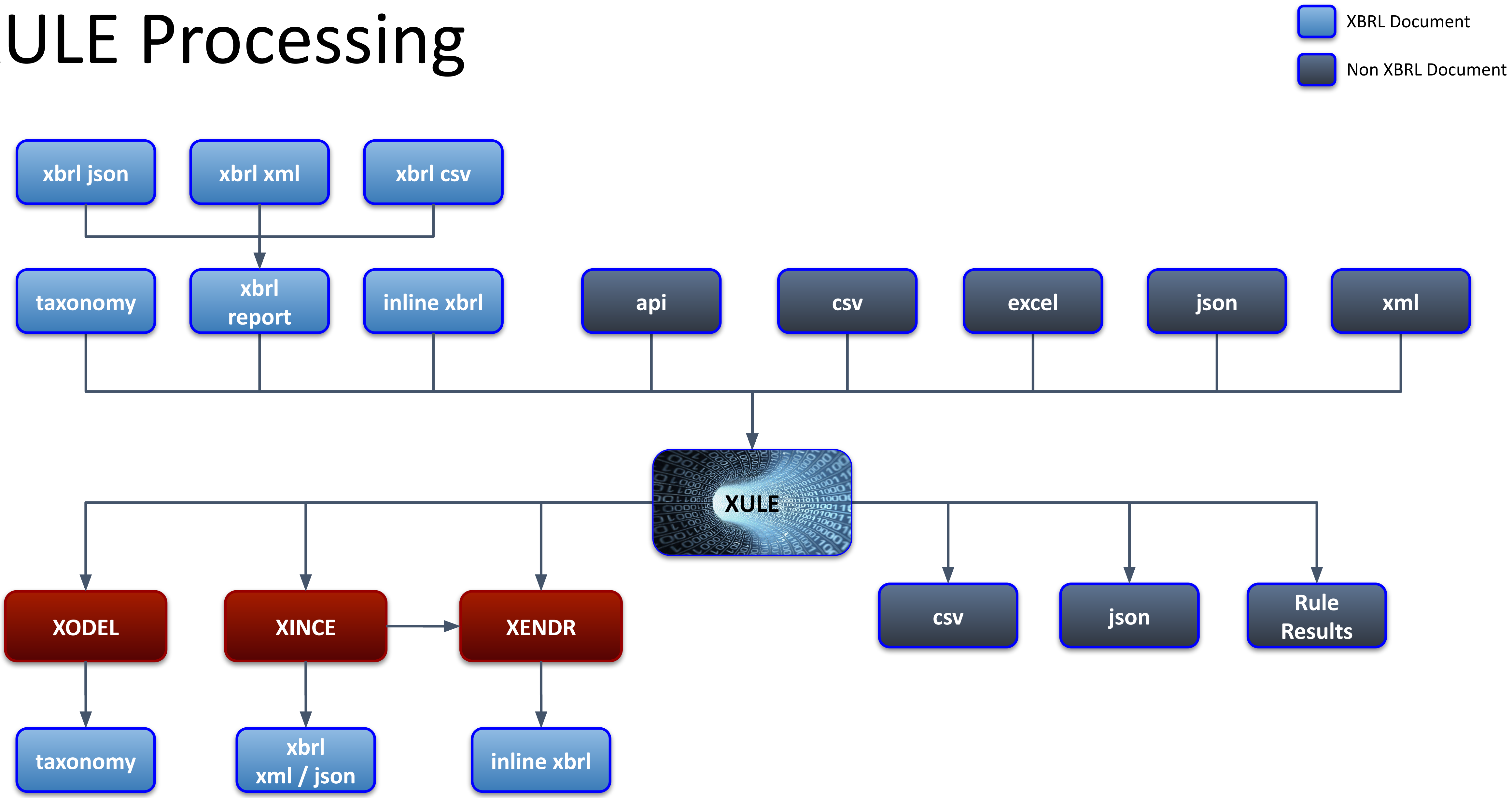
XULE Family

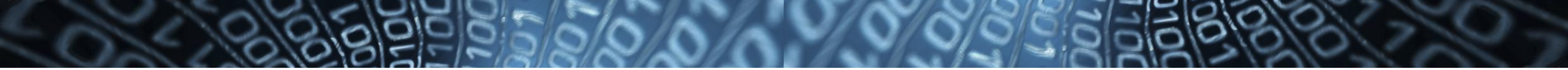
October 16, 2023

XULE Family

- Formula Expressions
  - Syntax to query and process XBRL taxonomies and instance documents. (XULE)
- Rendering of Inline Instance Documents
  - Uses XULE syntax within an html template to define and create inline XBRL instance documents. (XENDR)
- Instance Creation
  - Uses XULE syntax to create instance documents. (XINCE)
- Taxonomy Creation
  - Uses XULE syntax to create taxonomies. (XODEL)

XULE Processing





Simple NonNeg example

Concept	Period	Value	eg:Segment
eg:Assets	2023	6,560,000	
eg:Assets	2022	-5,460,000	
eg:Assets	2023	3,230,000	eg:WidgetDivision
eg:Assets	2022	-2,130,000	eg:Widget Division

```
namespace eg = 'https://example.com/taxonomy'
assert rule1 satisfied

$nonNeg = {@concept = eg:Assets};

$nonNeg < 0

message

'The concept {$nonNeg.name} for the period {$nonNeg.period}has a value of
{$nonNeg}. This concept cannot be negative.

Dimension: {$nonNeg.dimensions.join(',','=')}'
```

Simple NonNeg example

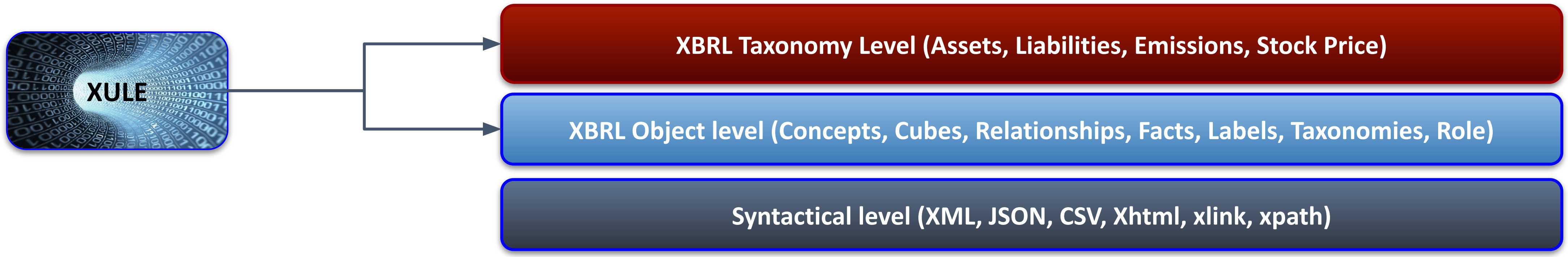
```
The concept Assets for the period 2022 has a value of -5,460,000. This
concept cannot be negative.

Dimension: none
```

```
The concept Assets for the period 2022 has a value of -2,130,000. This
concept cannot be negative.

Dimension: eg:Segment=eg:WidgetDivision
```

What is XULE?

1. XULE is a language that allows users to quickly and easily pull data from XBRL instances or taxonomies.
2. XULE is available as an Arelle plugin or part of the Altova Raptor product
3. XULE is used today to:
   - Define rules for validation of SEC, FERC and IFRS filings
   - Normalize XBRL data
   - Generate inline XBRL documents for the FERC
   - Create Instance documents
   - Converting data reported in XML, JSON, CSV and spreadsheet formats into XBRL instances

Abstraction Levels & XULE


XULE
XBRL Taxonomy Level (Assets, Liabilities, Emissions, Stock Price)
XBRL Object level (Concepts, Cubes, Relationships, Facts, Labels, Taxonomies, Role)
Syntactical level (XML, JSON, CSV, Xhtml, xlink, xpath)

Can
Write a rule to check if Assets are positive
Write a rule to check **if** all tables have dimensions
Write a rule to check **if** a role has a calculation

Cannot
Write a rule to check the document is XML valid
Write a rule to check invalid characters are used
Write a rule to check if the label linkbase file has an xml suffix.

How does XULE Work?

XULE supports the following operations:

- Query Taxonomy and Instance Properties
- Fact Filtering
- Taxonomy Navigation
- Dimensional Alignment and Iterations
- Mathematical and Text Operations
- Date Operations
- Custom Functions
- Defining Variables
- Management of Collections (Sets, Lists and Dictionaries)

Query object properties

Using XULE, all taxonomy and instance objects have properties that can be retrieved and manipulated to evaluate and create output.

XULE can get:

- concepts in a taxonomy
- references and labels for a concept
- roles in a taxonomy (sections of a report)
- properties for any fact
- cube information (fact data + corresponding role details)
- the taxonomy entry point
- the taxonomy used for an instance document
- the files making up a taxonomy

XBRL Taxonomy Objects

Taxonomy (DTS) Object

- concepts
- cubes
- concept
- cube
- dimensions
- dimensions-explicit
- dimensions-typed
- dts-document-locations
- entry-point-namespace
- entry-point
- networks
- namespaces
- roles
- arc-roles

Cube Object

- cube-concept
- closed
- drs-role
- dimensions
- primary-concepts
- facts

Dimension Object

- dimension-type
- default
- concept

Concept Object

- attribute
- balance
- base-type
- data-type
- enumerations
- has-enumerations
- is-abstract
- is-monetary
- is-numeric
- is-type
- label
- local-name
- name
- namespace-uri
- nillable
- period-type
- references
- relationships
- source-relationships
- substitutions

Role Object

- uri
- description
- used-on

Reference Object

- part-by-name
- parts
- reference-role

Parts Object

- part-value
- name
- Namespace-uri
- local-name
- order

Label Object

- text
- role
- lang

Network Object

- arcrole
- concept-names
- concepts
- source-concepts
- target-concepts
- relationships
- role
- roots

Relationship Object

- source
- source-name
- target
- target-name
- order
- weight
- preferred-label
- role
- arcrole
- arcrole-uri
- arcrole-description
- link-name
- arc-name
- network

Type Object

- name
- parent-type
- has-enumerations
- enumerations
- base-type
- ancestors

XBRL Instance Objects

Instance Object

- document-location
- taxonomy
- facts

Fact Object

- decimal
- concept
- period
- unit
- entity
- dimensions
- dimensions-explicit
- dimensions-typed
- id
- footnotes
- instance
- cube
- is-fact
- is-nil

Fact Object (Inline)

- inline-is-hidden
- inline-scale
- inline-format
- inline-display-value
- inline-negated
- inline-parents
- inline-children
- inline-ancestors
- inline-descendants

Unit Object

- numerator
- denominator
- id

Period Object

- days
- end
- start

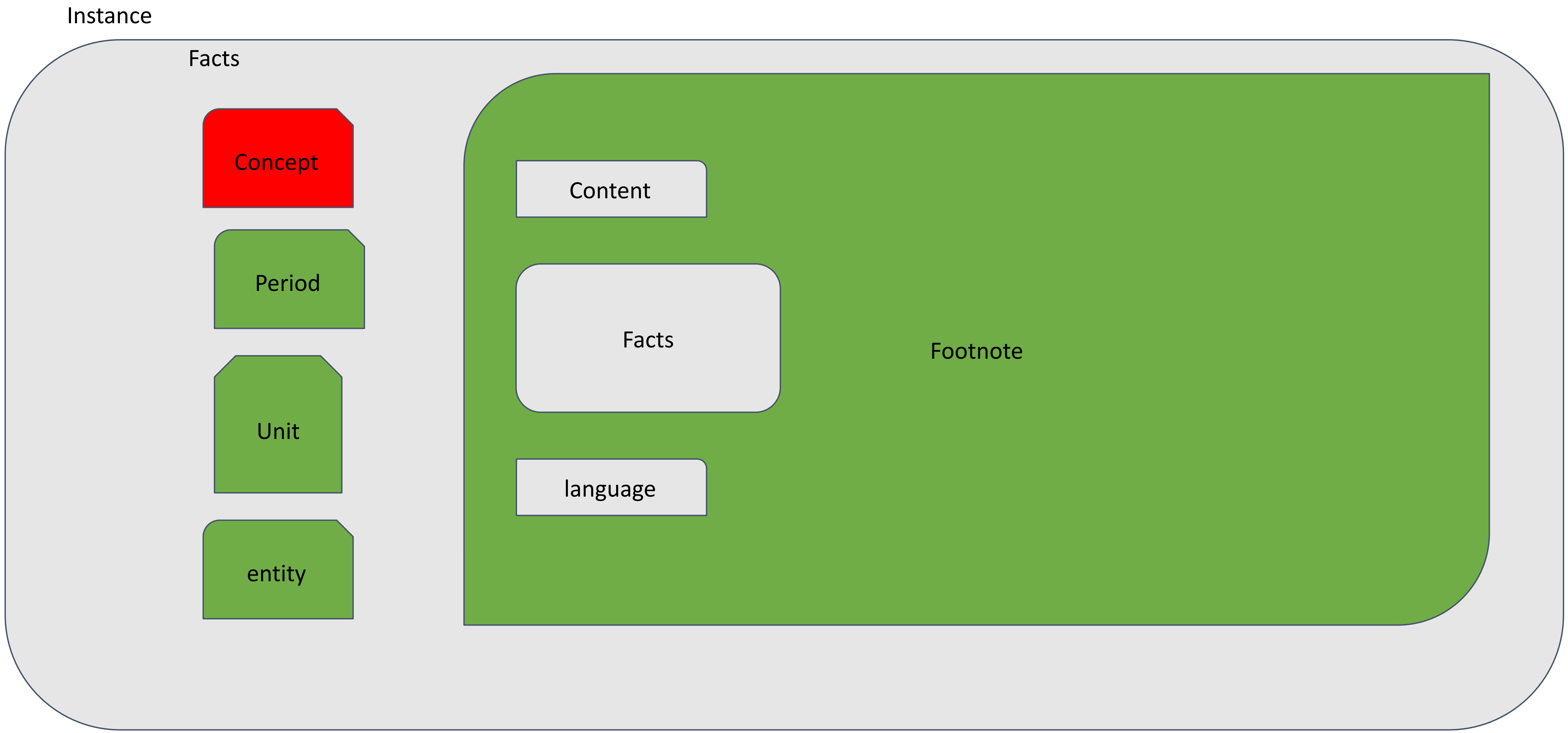
Footnote Object

- content
- arcrole
- role
- lang

entity Object

- id
- scheme

Object Relationship



What is Fact Filtering?

Fact filtering allows a users to pull factual information from one or more XBRL instance documents.

1. Get all the facts from an instance document
2. Get all the facts with a given concept name
3. Get all the facts with a given unit
4. Get all the facts with a given period i.e. All values at the end of 2022.
5. Get all the facts with a taxonomy defined dimension
6. Get all facts for a given entity
7. Get all facts in a dimensional cube

These filters can also be combined to allow granular querying of data in an instance document.

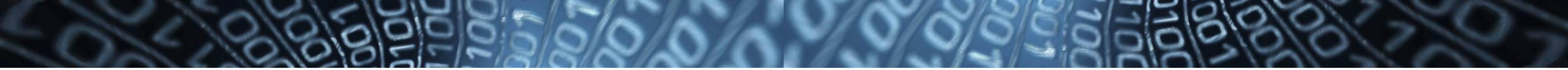
Refining Fact Filtering?

The **WHERE** filter allows the user to refine the search on additional properties.

Examples of properties not included in the filter include:

- Facts with a given decimal
- Facts with a value in a given range

These are properties of the fact itself. The fact filter queries fact data based on the dimensions of the fact, not the facts properties.



What is Fact Filtering?

The following example shows a factset filter that will return values in the instance document that are Assets

```
{@concept = Assets}
```

The following example shows a factset filter that will return values in the instance document where the values are recorded at 2021-12-31

```
{@period = date('2021-12-31')}
```

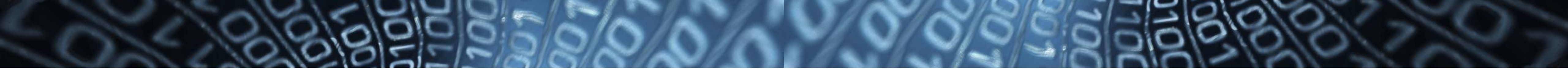
What is Fact Filtering?

To get the value of Assets for 2021-12-31 the previous filters can be combined:

```
{@concept = Assets @period =
date('2021-12-31')}
```

To get the value of Assets for 2021-12-31 where the value is greater than 100 the where clause is used:

```
{@concept = Assets @period =
date('2021-12-31') where $fact > 100}
```



What is Taxonomy Navigation?

- XULE allows a user to query one or more taxonomies. The primary mechanism to query taxonomies is taxonomy navigation.
- Taxonomy Navigation allows the following operations:
  - Return concepts in a specific tree
  - Allows navigation up and down a tree
- Taxonomy navigation can be used to get a list of all concepts appearing in the Balance Sheet.
- Alternatively it can be used to get a list of monetary instance items that appear in the notes and not on the balance sheet.

What is Taxonomy Navigation?

Taxonomy navigation allows a complete understanding of what a concept means from the taxonomy.

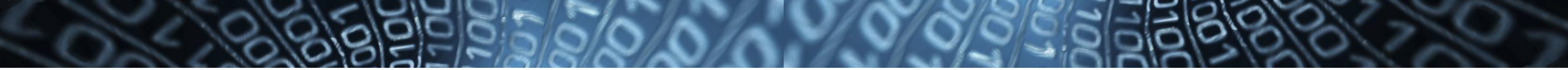
Taxonomy navigation also allows the user to capture the details of how an element was retrieved through the navigation process.

What is Taxonomy Navigation?

The following is an example of a navigate expression

```
navigate parent-child descendants from
IncomeStatementAbstract stop when
$relationship.target.name ==
RevenuesAbstract
```

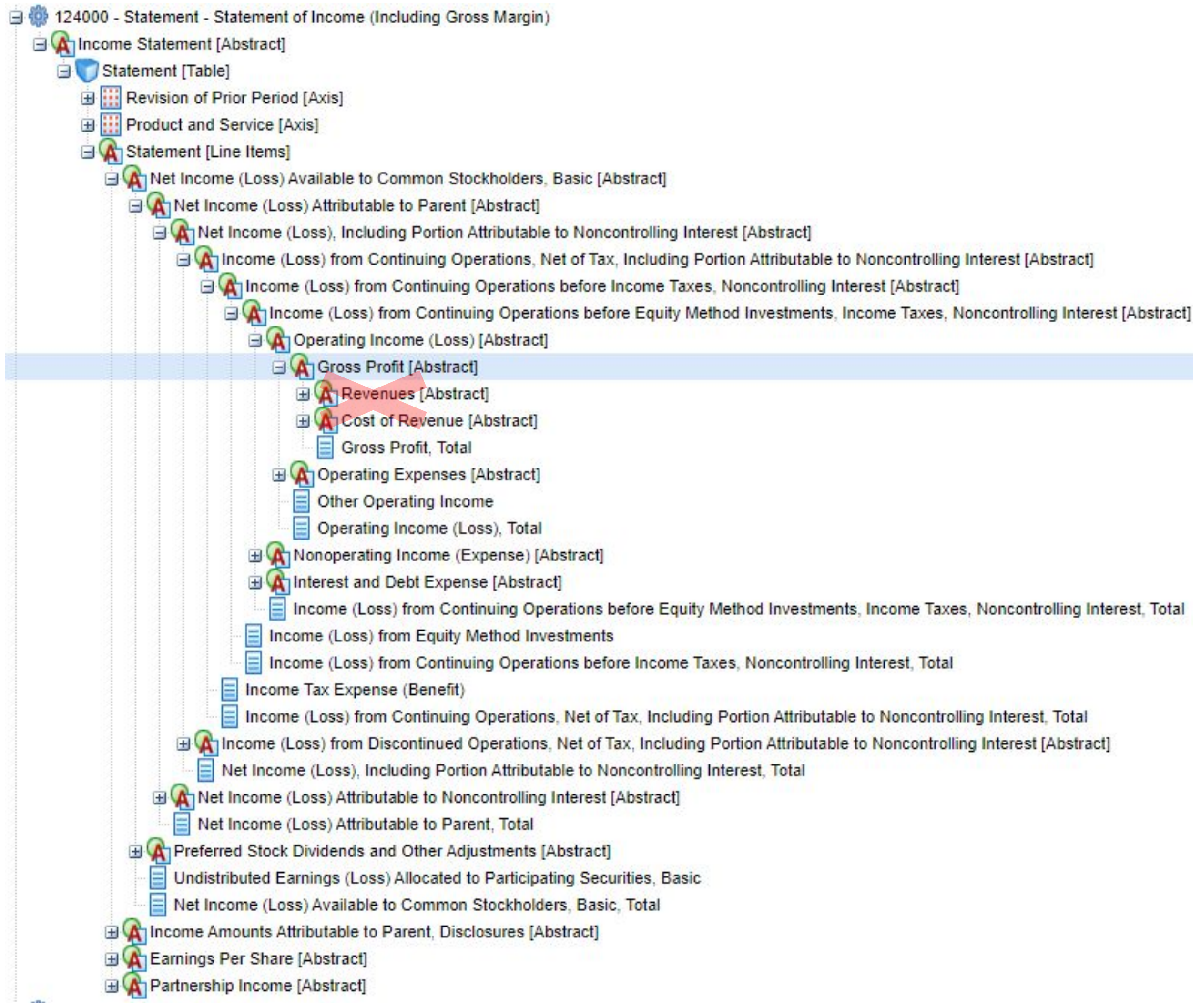
Returns descendant concepts of IncomeStatementAbstract in the presentation linkbase, but will exclude any children of RevenuesAbstract.



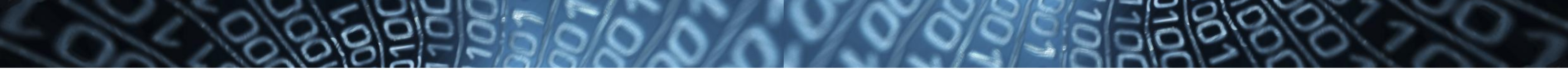
Taxonomy Navigation

```
navigate parent-child
descendants from
IncomeStatementAbstract stop
when $relationship.target.name
== RevenuesAbstract
```

Returns all the descendant concepts of IncomeStatementAbstract in the presentation linkbase of the filing, but will exclude any children of RevenuesAbstract.



```
⊟ 124000 - Statement - Statement of Income (Including Gross Margin)
  ⊟ Income Statement [Abstract]
    ⊟ Statement [Table]
      ⊞ Revision of Prior Period [Axis]
      ⊞ Product and Service [Axis]
      ⊟ Statement [Line Items]
        ⊟ Net Income (Loss) Available to Common Stockholders, Basic [Abstract]
          ⊟ Net Income (Loss) Attributable to Parent [Abstract]
            ⊟ Net Income (Loss), Including Portion Attributable to Noncontrolling Interest [Abstract]
              ⊟ Income (Loss) from Continuing Operations, Net of Tax, Including Portion Attributable to Noncontrolling Interest [Abstract]
                ⊟ Income (Loss) from Continuing Operations before Income Taxes, Noncontrolling Interest [Abstract]
                  ⊟ Income (Loss) from Continuing Operations before Equity Method Investments, Income Taxes, Noncontrolling Interest [Abstract]
                    ⊟ Operating Income (Loss) [Abstract]
                      ⊟ Gross Profit [Abstract]
                        ⊞ Revenues [Abstract]
                        ⊞ Cost of Revenue [Abstract]
                          Gross Profit, Total
                      ⊞ Operating Expenses [Abstract]
                        Other Operating Income
                        Operating Income (Loss), Total
                    ⊞ Nonoperating Income (Expense) [Abstract]
                    ⊞ Interest and Debt Expense [Abstract]
                      Income (Loss) from Continuing Operations before Equity Method Investments, Income Taxes, Noncontrolling Interest, Total
                    Income (Loss) from Equity Method Investments
                    Income (Loss) from Continuing Operations before Income Taxes, Noncontrolling Interest, Total
                  Income Tax Expense (Benefit)
                  Income (Loss) from Continuing Operations, Net of Tax, Including Portion Attributable to Noncontrolling Interest, Total
              ⊞ Income (Loss) from Discontinued Operations, Net of Tax, Including Portion Attributable to Noncontrolling Interest [Abstract]
                Net Income (Loss), Including Portion Attributable to Noncontrolling Interest, Total
            ⊞ Net Income (Loss) Attributable to Noncontrolling Interest [Abstract]
              Net Income (Loss) Attributable to Parent, Total
          ⊞ Preferred Stock Dividends and Other Adjustments [Abstract]
            Undistributed Earnings (Loss) Allocated to Participating Securities, Basic
            Net Income (Loss) Available to Common Stockholders, Basic, Total
        ⊞ Income Amounts Attributable to Parent, Disclosures [Abstract]
        ⊞ Earnings Per Share [Abstract]
        ⊞ Partnership Income [Abstract]
```



What is Dimensional Alignment and Iterations?

Alignment refers to the process of aligning facts so that consistent calculations are applied.

For example, adding Current Assets and Noncurrent Assets to derive the value of Assets. It’s expected that Current Assets for 2021 is added to Noncurrent Assets for 2021. The user expects the date periods for the calculation to be aligned.

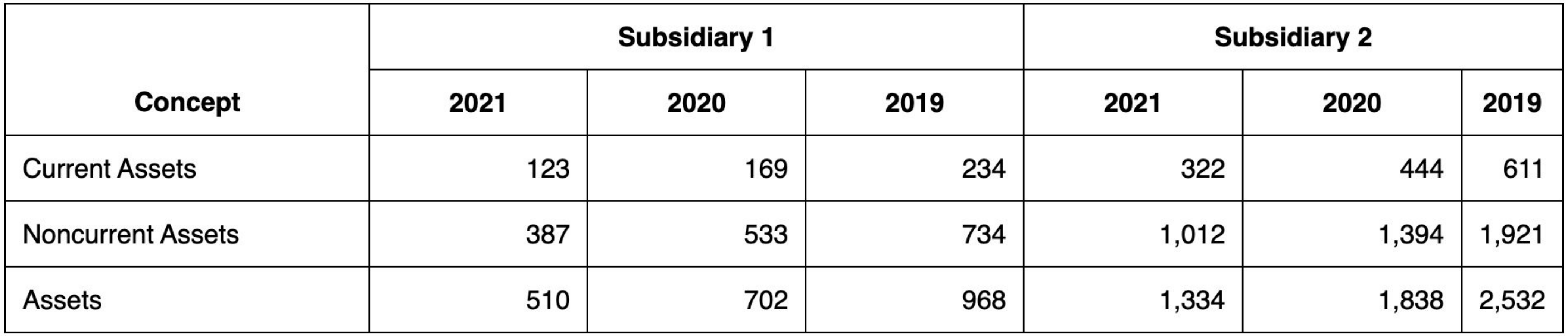
If there is three years of data for each of these elements the calculation would be expected to run three times. Once for 2019, once for 2020 and once for 2021. The running of each of these calculations is called an iteration.

The same calculation can be used for all 3 years without the need to define a separate formula for each year.

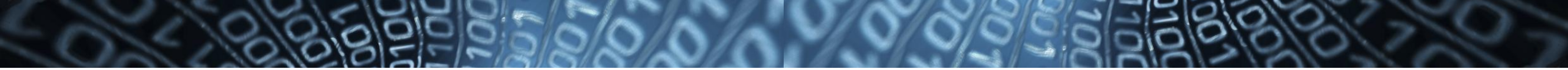
What is Dimensional Alignment and Iterations?

The expression below will create 6 boolean results, that correspond to each column of the table. With a value of true or false.

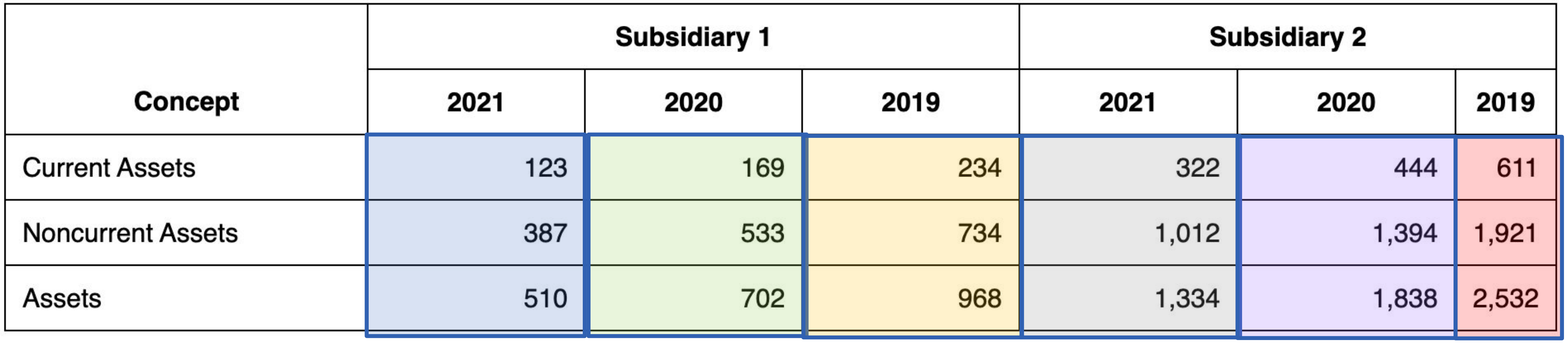
```
{@Assets} != {@CurrentAssets} + {@NoncurrentAssets};
```



Concept	Subsidiary 1			Subsidiary 2		
	2021	2020	2019	2021	2020	2019
Current Assets	123	169	234	322	444	611
Noncurrent Assets	387	533	734	1,012	1,394	1,921
Assets	510	702	968	1,334	1,838	2,532



Dimensional Alignment and Iterations



Concept	Subsidiary 1			Subsidiary 2		
	2021	2020	2019	2021	2020	2019
Current Assets	123	169	234	322	444	611
Noncurrent Assets	387	533	734	1,012	1,394	1,921
Assets	510	702	968	1,334	1,838	2,532

```
{@Assets} != {@CurrentAssets} +
    {@NoncurrentAssets}
```

What is Dimensional Alignment and Iterations?

The use of iterations in XULE is extremely powerful as a single rule can perform many calculations.

In a tool such as excel the user would need to define 6 calculations for each year of each subsidiary. If an additional subsidiary is added then additional calculations have to be added. If expressed in XULE the calculation only needs to be defined once and is independent of the actual data reported.

What is Dimensional Alignment and Iterations?

Alignment and iterations work across all dimensions. To calculate the consolidated value of all subsidiaries the following expression could be used:

```
$ConsolidatedEntity = {@SubDimension = Sub1} + {@SubDimension = Sub2};
$ConsolidatedEntity != {@SubDimension = none}
```

This generates 9 iterations using the previous example, and would check that the consolidated total for CurrentAssets, NoncurrentAssets, and Assets for each of the three years is equal to the consolidated total. i.e. 3 concepts multiplied by 3 years.

What are nested filters?

Allows the calculation of an interim value that can then be checked. For example calculate a total for a record and then sum the derived value of $270.

ContractAxis	RatePerDay	NumberDays	CalcValue
1	5	30	150
2	6	10	60
3	3	20	60
			270

What are nested filters?

```
sum({@ContractAxis = * {@RatePerDay @unit} *
{@NumberDays @unit}}) != {@CalcValue @unit
@ContractAxis = none}
```

The calc between rate per day * Number of days is done for each record but not between records. The value is then taken out of alignment for each contract and summed and compared to the calcValue total.

(This is not possible in XBRL formula)

Mathematical and Text Operations

Xule supports many common mathematical operations. Xule has a number of aggregation functions to aggregate sets and lists of data that are returned.

- Aggregation Functions
  - Sum
  - Average
  - Count
  - Product
  - Max
  - Min
  - Standard Deviation
  - Any
  - All

Xule also supports standard operators such as multiplication, addition, subtraction, division, power, log10, abs, round, trunc, mod, random etc.

Mathematical and Text Operations

Xule supports many common textual operations.

- Text Functions
    - Regex-match
    - Length
    - String
    - Number
    - Split
    - Contains
    - Upper and Lower Case
    - Substring
    - trim

Date Operations

Xule supports the following date operations.

- Date Functions
    - Convert string to date
    - Return year, month and day from a date
    - Calculate days in a duration
    - Add and subtract a time-span to a date
    - Define a durational period
    - Define a forever period

Custom Functions

Xule allows the definition of user defined functions. This allows a user to define a function that can be used across multiple rulesets and in multiple rules.

Functions can also reference global constants.

Functions cannot be recursive.

The following function to add two variables is defined as follows:

```
function add_two_numbers($a , $b)
  $a + $b
```

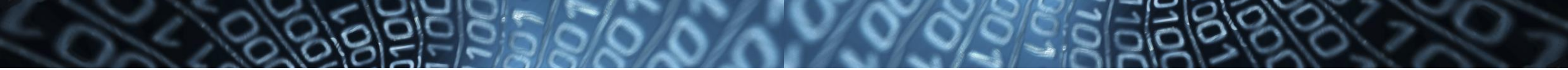
Defining Variables

- Variables can be defined within a rule or as a global constant.
- Global constants are defined once and can be used by any rule.
- Local variables in a rule are updated for each iteration of the rule.
- Local variables are defined as follows:

```
$MyInteger = 23
```

- Global variables are prefixed with the keyword constant

```
constant $MyInteger = 23
```



Management of Collections

Xule supports the use of list, sets and dictionaries.

These allow the storing and manipulation of values together in one of these collections.

- Sets are unique and unordered
- lists are an ordered list of items that can be referenced by an index
- Dictionaries are an array of values with a key to index the values.

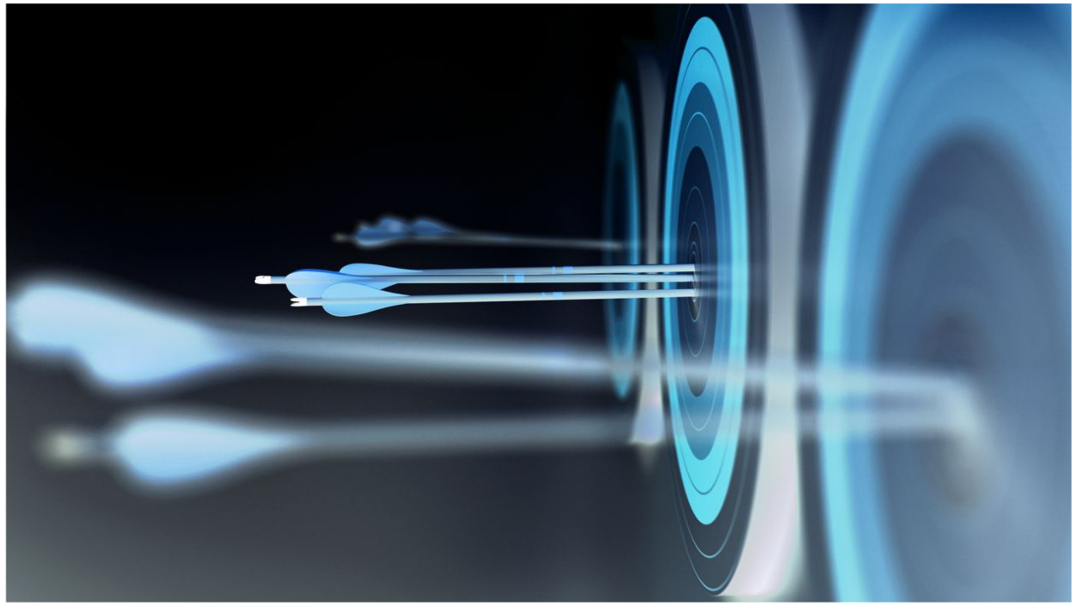
Xule allows values to be put into these collections and provides functions to manipulate them.

Management of Collections

Xule supports the following operators to manipulate data in collections:

- Get the intersection of two or more sets
- Get the union of two or more sets of data
- Get the difference of two sets
- Reference the keys or index of a dictionary or list to get the values in the collection
- Get the count of items in the set or a list
- Test properties of the collection
- Convert a collection to a different type i.e. convert a list to a set.
- Turn a dictionary into JSON
- Collections can contain sets, lists, objects, strings, dictionaries, variables etc.

XINCE



What is XINCE?

1. XINCE uses the xule syntax to create instance documents.
2. XINCE is available as an Arelle plugin
3. XINCE is used today to:
   - Generated 60,000 instance documents for standardized XBRL data.
   - Generate instance documents test cases for DQC rules (NonNegs)

What is XINCE?

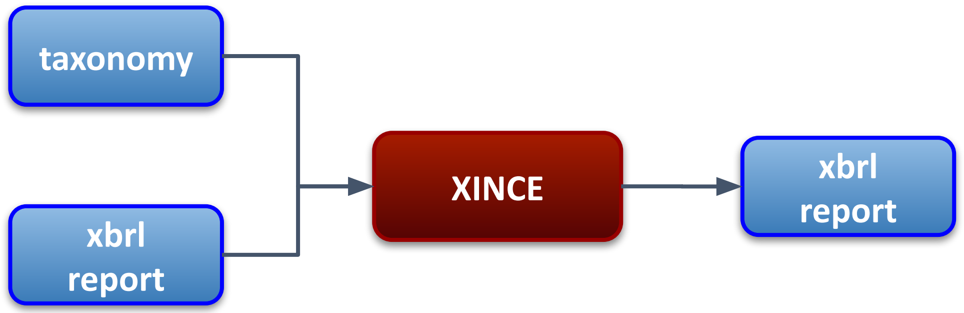
1. XINCE uses the XULE expression syntax to create XBRL reports using a XULE processor.
2. XINCE can create XBRL instances in either a JSON or XML format.
3. The XINCE syntax supports associating dimensions with a fact. These include the period the concept, the entity the unit and any other taxonomy defined dimensions.
4. XINCE can be used to create as many facts as required. Every fact that is created must specify the instance document that it belongs too.
5. XINCE created facts can then be written to a single instance or multiple instances.

XINCE Uses

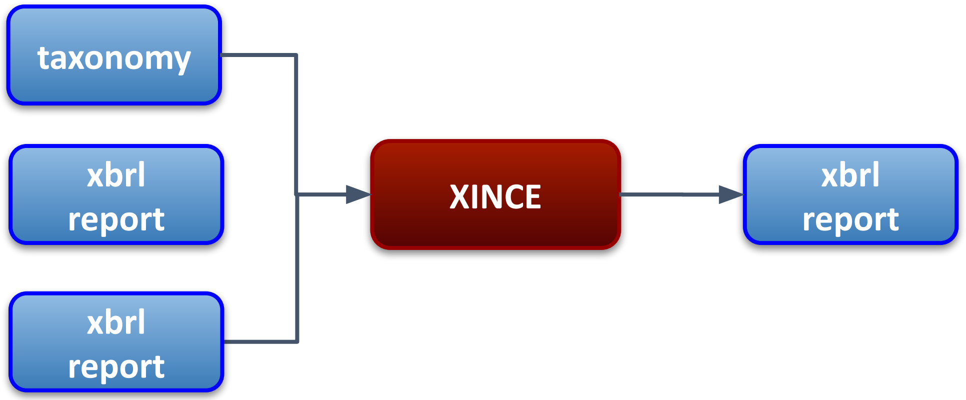
🟦 XBRL Document

⬛ Non XBRL Document

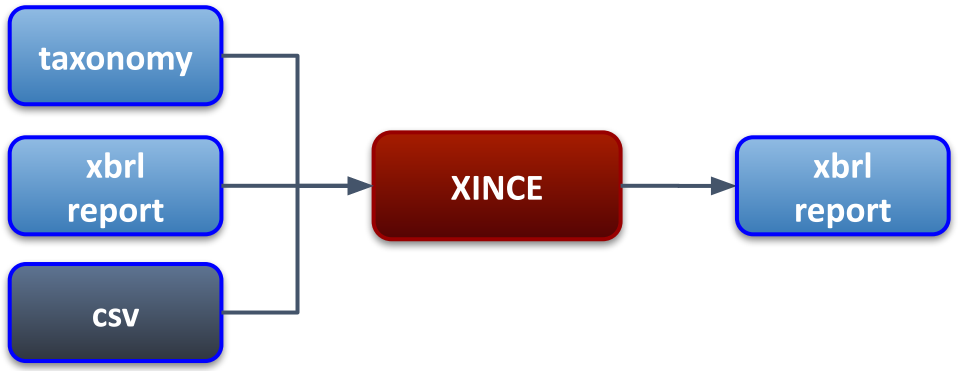
Create extracted xbrl report



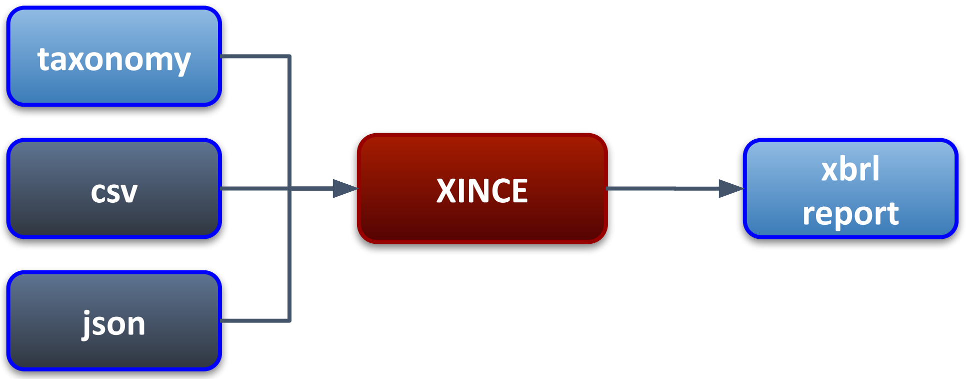
Create new xbrl report from multiple reports



Create csv enriched xbrl report



Create new xbrl report from csv and json data



Create a Forecast

Create an example instance that takes an existing instance document and creates new values for the 2024 year that increase by 10%.

Concept	Period	Value	eg:SegmentAxis
eg:Assets	2023	6,560,000	
eg:Assets	2022	5,460,000	
eg:Assets	2023	3,230,000	eg:WidgetDivision
eg:Assets	2022	2,130,000	eg:Widget Division
eg:Revenues	2023	13,230,000	
eg:Revenues	2022	12,130,000	
eg:Revenues	2023	7,230,000	eg:WidgetDivision
eg:Revenues	2022	5,130,000	eg:Widget Division

Example - Create an Empty Instance

First we define a new instance document based on the extension taxonomy used by the filer. We call the new instance document myInstance

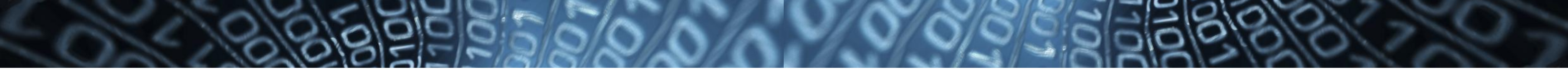
```
output createInstance
true
instance-name  "myInstance"

instance-taxonomy
'https://www.sec.gov/Archives/edgar/data/891166/000089116622000114/u
ve-20220930.xsd'
```

Example - Add Instant Facts

Next we select the numeric instant facts in a filing and multiply the values by 10% and increment the period by 1 year.

These values are added to the instance myInstance

```
output add_instant_fact_values
{where $fact.is-numeric and $fact.concept.period-type == instant and
$fact.period.end.year == '2023'}

fact-value $rule-value * 1.1
fact-alignment alignment().to-xince
fact-add-dimension dict(list(ScenarioAxis,ForecastMember)).to-xince
fact-period ($rule-value.period.end + time-span('P1Y')).to-xince
fact-decimals $rule-value.decimals
fact-instance "myInstance"
```



Example - Add Duration Facts

Finally we select the numeric duration facts in the filing and multiply the values by 10% and increment the period by 1 year.

These values are added to the instance myInstance

```
output add_duration_fact_values
{where   $fact.is-numeric   and   $fact.concept.period-type   ==     duration     and
$fact.period.end.year == '2023'}

fact-value $rule-value * 1.1
fact-alignment alignment().to-xince
fact-add-dimension dict(list(ScenarioAxis,ForecastMember)).to-xince
fact-period  duration($rule-value.period.start +
time-span('P1Y'),$rule-value.period.end + time-span('P1Y'))to-xince
fact-decimals  $rule-value.decimals
fact-instance "myInstance"
```

Resulting Instance

Creates new instance with forecast data.

Concept	Period	Value	eg:SegmentAxis	eg:ScenarioAxis
eg:Assets	2024	7,216,000		eg:ForecastMember
eg:Assets	2024	6,006,000	eg:WidgetDivision	eg:ForecastMember
eg:Revenues	2024	14,533,000		eg:ForecastMember
eg:Revenues	2024	13,343,000	eg:WidgetDivision	eg:ForecastMember

Example - Add Original Instance Facts

Original values are added to the instance myInstance

```
output add_original_values
{}

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

Resulting Instance

Concept	Period	Value	eg:SegmentAxis	eg:ScenarioAxis
eg:Assets	2024	7,216,000		eg:ForecastMember
eg:Assets	2024	6,006,000	eg:WidgetDivision	eg:ForecastMember
eg:Revenues	2024	14,533,000		eg:ForecastMember
eg:Revenues	2024	13,343,000	eg:WidgetDivision	eg:ForecastMember
eg:Assets	2023	6,560,000		
eg:Assets	2022	5,460,000		
eg:Assets	2023	3,230,000	eg:WidgetDivision	
eg:Assets	2022	2,130,000	eg:Widget Division	
eg:Revenues	2023	13,230,000		
eg:Revenues	2022	12,130,000		
eg:Revenues	2023	7,230,000	eg:WidgetDivision	
eg:Revenues	2022	5,130,000	eg:Widget Division	

XINCE Functionality

- Allows creation and modification of XBRL facts from existing Instance documents.
- Allows creation of new facts from data files such as CSV, excel, json and XML.
- XINCE supports the creation of footnotes including fact to fact footnotes and any extended footnote types.
- Allows the creation of multiple instance documents in a single run.
- Can save as json and xml formats.
- XENDR can be used to create inline documents after XINCE has created the instance document.

XINCE Functionality

Can define the following:

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

XENDR

What is XENDR?

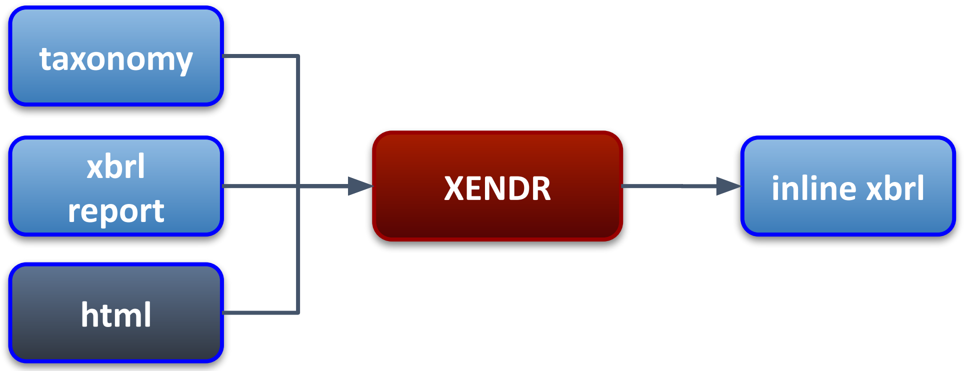
1. XENDR uses the xule syntax to create inline xbrl instance documents.
2. XENDR is available as an Arelle plugin
3. XENDR is used today to:
   - Generate inline XBRL documents for the FERC
   - Generate test cases of inline XBRL for DQC rules

XENDR Uses

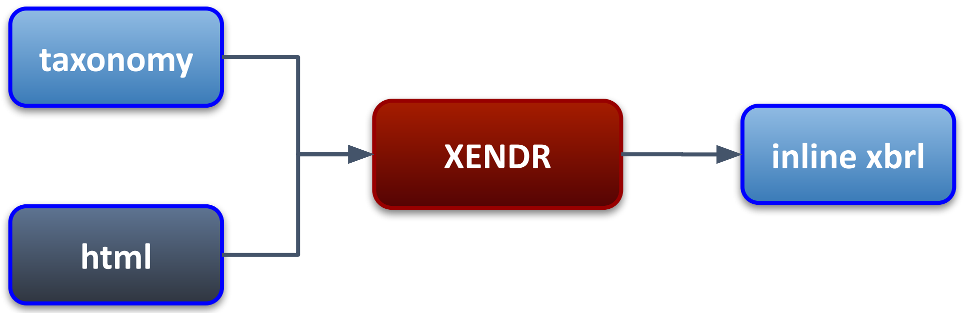
🟦 XBRL Document

⬛ Non XBRL Document

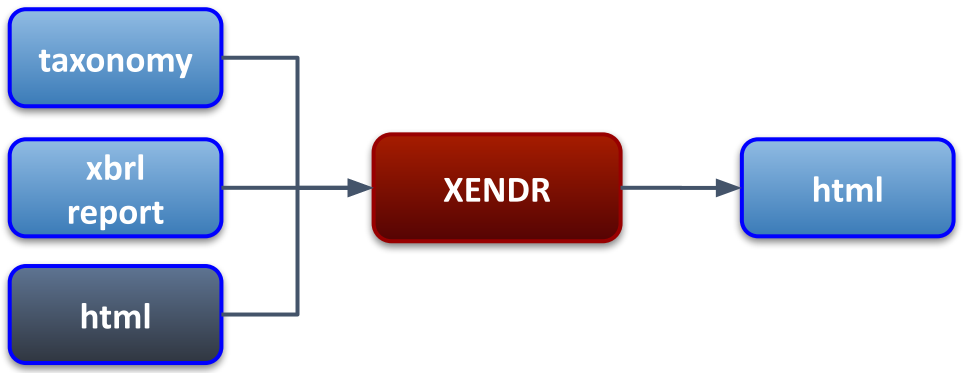
Create iXBRL Document as a Form



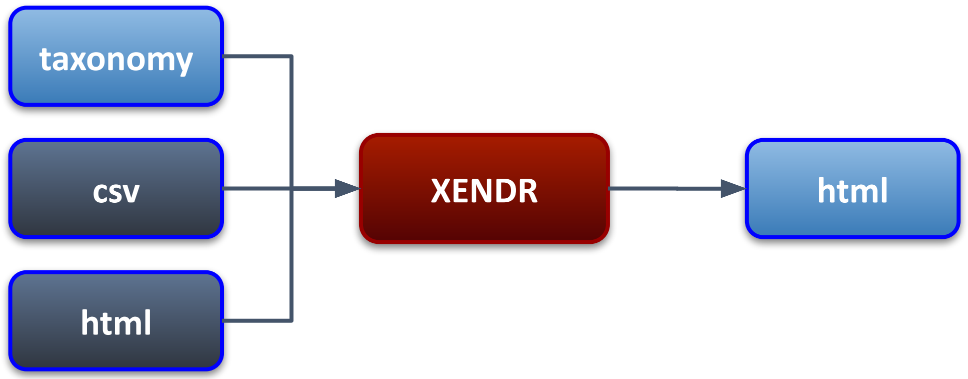
Create iXBRL Document as a blank Form



Create HTML Page(s)

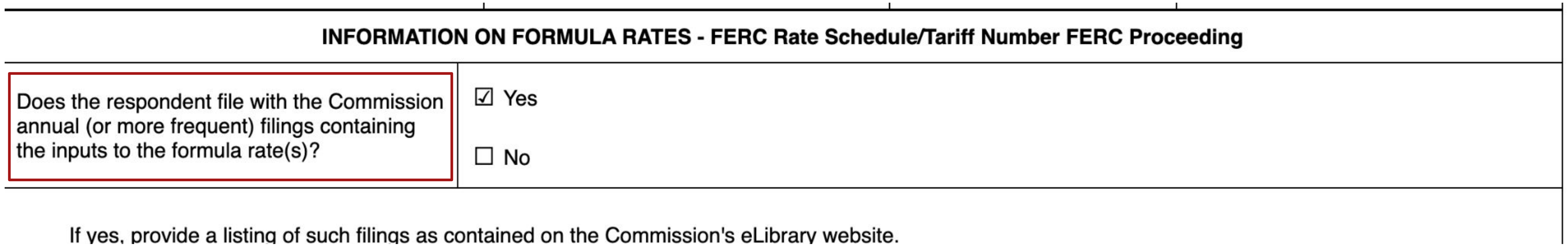


Create HTML Page(s) with taxonomy Information



Insert XULE Expressions into HTML

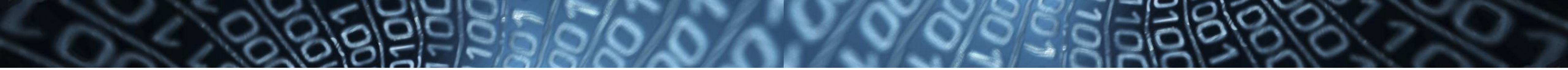
```
<html>
    .....
    <xendr:replace>
        <xendr:expression>
    taxonomy().concept(ferc:UsesFormulaRates).label("http://ferc.gov/form/2020-01-01/roles/la
    bel/F1FormulaRates").text
        </xendr:expression>
    </xendr:replace>
    ....
</html>
```



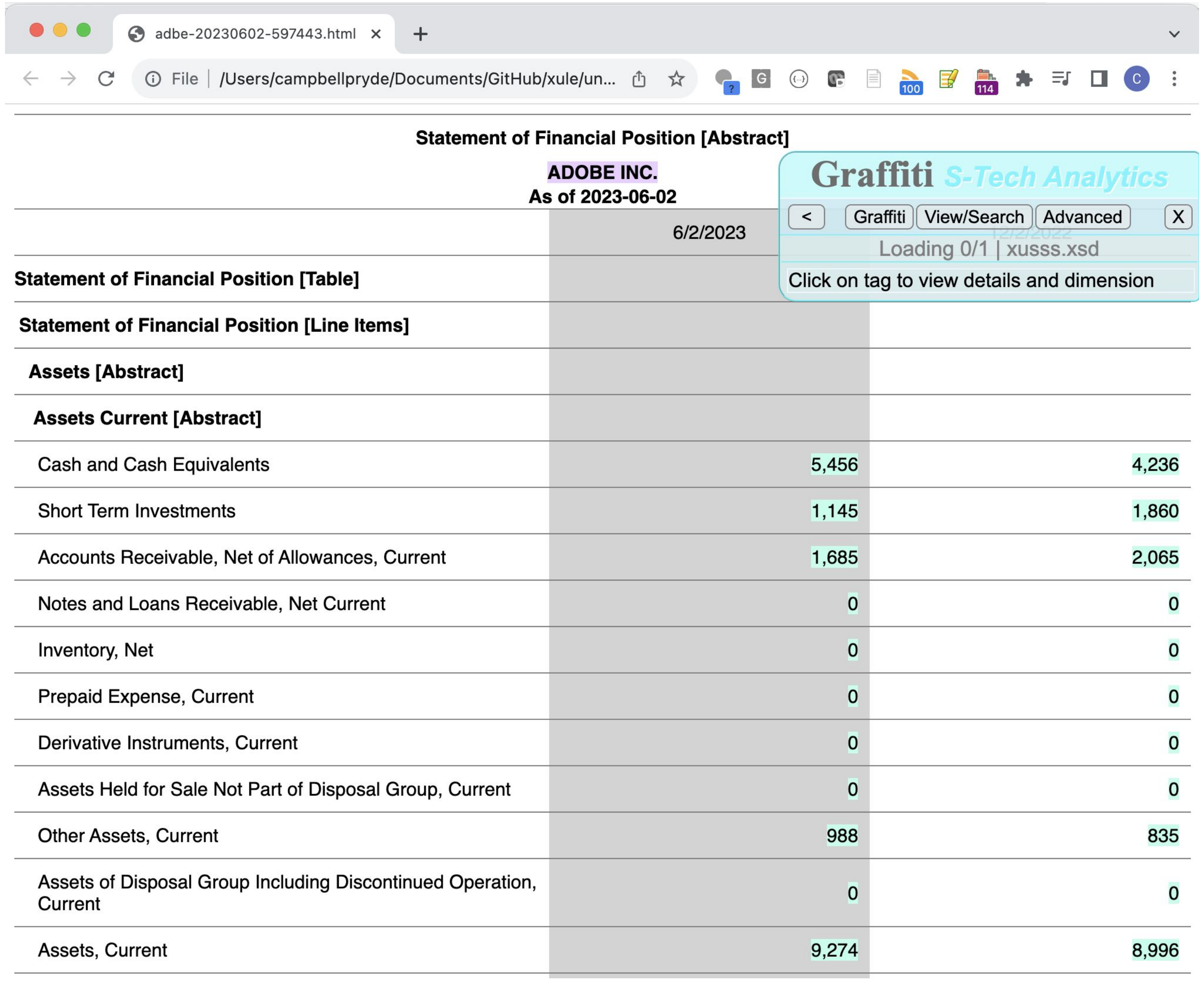
INFORMATION ON FORMULA RATES - FERC Rate Schedule/Tariff Number FERC Proceeding

Does the respondent file with the Commission annual (or more frequent) filings containing the inputs to the formula rate(s)?	[✓ Yes] No

If yes, provide a listing of such filings as contained on the Commission's eLibrary website.



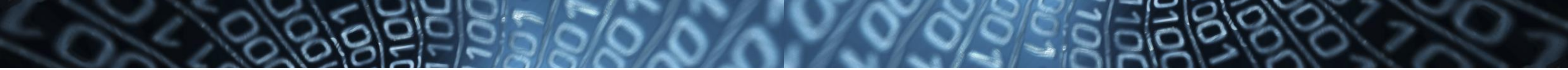
Generate Balance Sheet


adbe-20230602-597443.html
File | /Users/campbellpryde/Documents/GitHub/xule/un...
Graffiti S-Tech Analytics
<
Graffiti
View/Search
Advanced
X
Loading 0/1 | xusss.xsd
Click on tag to view details and dimension

Statement of Financial Position [Abstract]

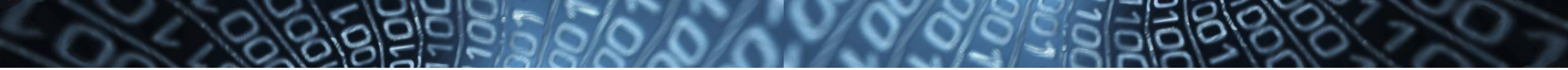
ADOBE INC.
As of 2023-06-02

	6/2/2023	12/2/2022
Statement of Financial Position [Table]		
Statement of Financial Position [Line Items]		
Assets [Abstract]		
Assets Current [Abstract]		
Cash and Cash Equivalents	5,456	4,236
Short Term Investments	1,145	1,860
Accounts Receivable, Net of Allowances, Current	1,685	2,065
Notes and Loans Receivable, Net Current	0	0
Inventory, Net	0	0
Prepaid Expense, Current	0	0
Derivative Instruments, Current	0	0
Assets Held for Sale Not Part of Disposal Group, Current	0	0
Other Assets, Current	988	835
Assets of Disposal Group Including Discontinued Operation, Current	0	0
Assets, Current	9,274	8,996



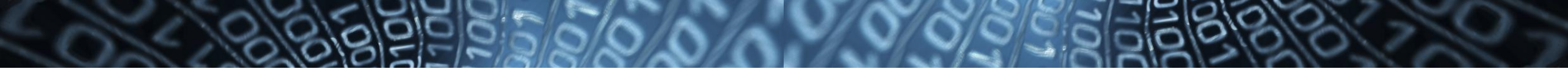
Balance Sheet

```
<tr class="schedule-row" xendr:repeat="BSLineItems">
     <xendr:expression name="BSLineItems">
     $Balance_Sheet = navigate parent-child descendants from xusss:StatementOfFinancialPositionAbstract role
     "https://taxonomies.xbrl.us/standardized-statements/roles/StatementOfFinancialPositionClassified" returns list (target-name,
     preferred-label, navigation-depth);  // Get the balance sheet structure from the taxonomy
     $data-labels = list(for $line in $Balance_Sheet
        $concept = taxonomy().concept($line[1]);  // Get the taxonomy concept for the label
        $current = first-value-or-none(first(list([covered @concept = $line[1] @period = $currentInstant]))); // Current balance
        $prior = first-value-or-none(first(list([covered @concept = $line[1] @period = $priorInstant]))); // Prior balance

       list($concept,                                      // Concept                [1]
            $concept.label($line[2].role.uri).text,        // Row caption            [2]
            $current,                                      // Current fact value     [3]
            $prior,                                        // Prior fact value       [4]
            $line[2].role.uri,                             // Preferred label        [5]
            $line[3]                                       // Navigation depth       [6]
            )
       );
      for $row in $data-labels
     </xendr:expression>
```



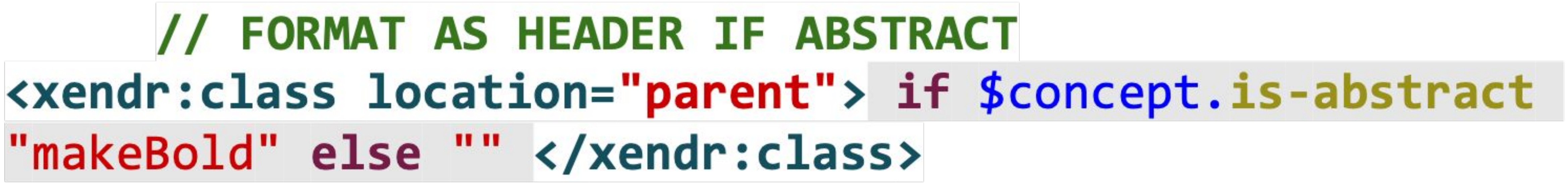
$dataLabels

$concept	$concept.label($line[2].role.uri).text	$current	$prior	$Line[2]	$Line[3]
StatementOfFinancialPositionTable	Statement of Financial Position [Table]	None	None	label	1
StatementOfFinancialPositionLineItems	Statement of Financial Position [Line Items]	None	None	label	2
AssetsAbstract	Assets [Abstract]	None	None	label	3
AssetsCurrentAbstract	Assets Current [Abstract]	None	None	label	4
CashAndCashEquivalents	Cash and Cash Equivalents	5,456,000,000	4,236,000,000	label	5
ShortTermInvestments	Short Term Investments	1,145,000,000	1,860,000,000	label	5
AccountsReceivableNetOfAllowancesCurrent	Accounts Receivable, Net of Allowances, Current	1,685,000,000	2,065,000,000	label	5
………					
1	2	3	4	5	6



Rendering Uses the Class Element

- XENDR allows control of the inline presentation by setting classes.
- Classes can be set as the result of an expression

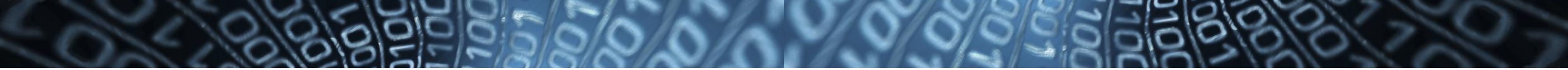


```
        // FORMAT AS HEADER IF ABSTRACT
<xendr:class location="parent"> if $concept.is-abstract
"makeBold" else "" </xendr:class>
```

- This assigns a class to the parent tag of “makeBold”.
- The CSS can then be defined to make this class bold text.

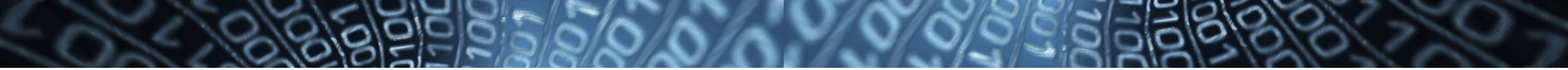
Create Column 1

```
// CREATE COL1
<td class="fs-statement description_items">

     <xendr:replace>

           // GET ROW CAPTION
           <xendr:expression name="BSLineItems" part="1" > $row[2] </xendr:expression>

           // FORMAT AS TOTAL IF USES TOTAL LABEL
           <xendr:class location="parent"> if $row[5] == "http://www.xbrl.org/2003/role/totalLabel" "total-row" else "" </xendr:class>

           // FORMAT AS HEADER IF ABSTRACT
           <xendr:class location="parent"> if $row[1].is-abstract "sch-title" else "" </xendr:class>

           // INDENT HEADER BASED ON TREE DEPTH IN TAXONOMY
           <xendr:class location="parent"> "padding-" + $row[6].string </xendr:class>

     </xendr:replace>

</td>
```



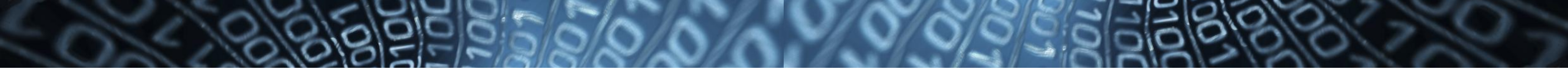
Create Row 1 Column 1

```
 // CREATE COL1
   <td class=" sch-title padding-1">

        Statement of Financial Position [Table]

   </td>
```

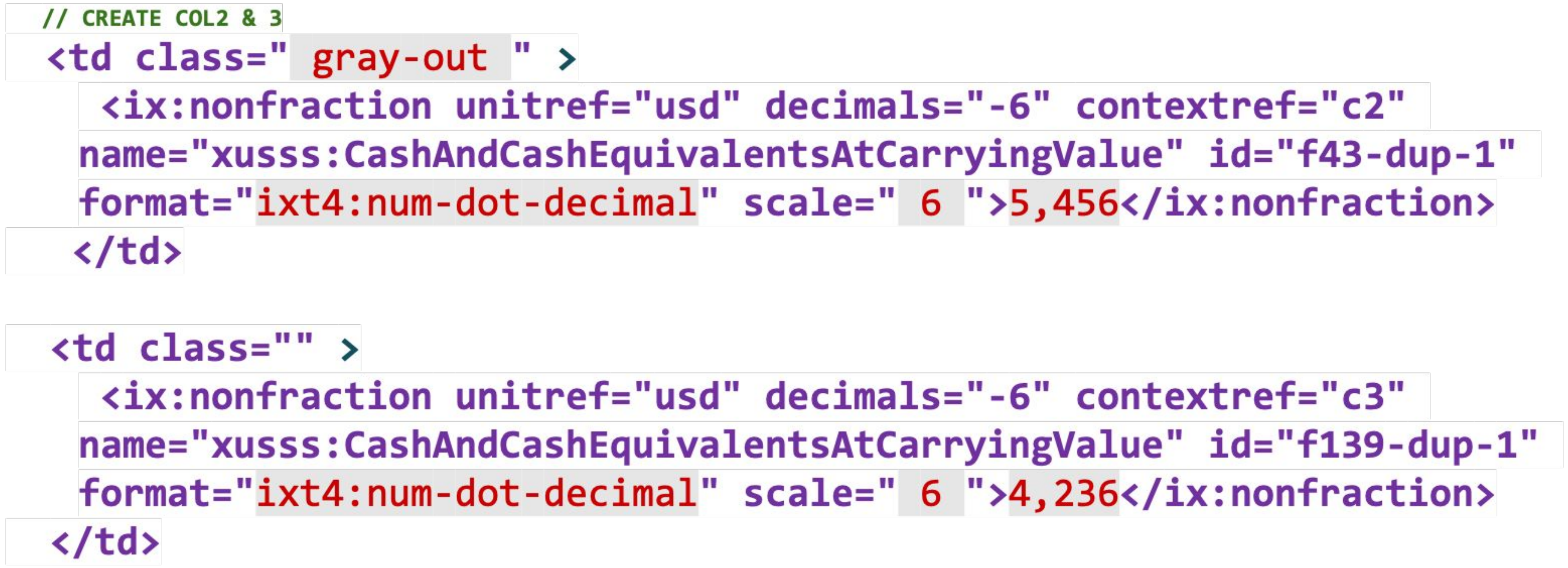


Create Column 2 & 3

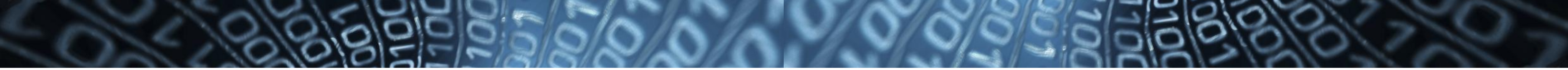
```
// CREATE COL2 & 3
<td class="fs-statement monetary_items" xendr:repeatWithin="BSLineItems" xendr:repeat="cols">
    <xendr:expression name="cols">
      for $col in list(3,4)  // LOOP THROUGH THE CURRENT AND PRIOR VALUES
    </xendr:expression>
       <xendr:replace>

               // GET VALUE FOR CURRENT AND PRIOR PERIOD AND RENDER AS XBRL FACT
          <xendr:expression name="cols" part="2" fact="true"> $row[$col] </xendr:expression>

               // MAKE FIRST COL NUMBERS GRAY BY SETTING CLASS TO gray-out
          <xendr:class location="parent"> if $col == 3 "gray-out" else "" </xendr:class>

              // FORMAT AS TOTAL IF USES TOTAL LABEL
          <xendr:class location="parent"> if $row[5] == "http://www.xbrl.org/2003/role/totalLabel" "total-row" else "" </xendr:class>

               // FORMAT NUMBERS USING XBRL TRANSFORMS IN REVERSE
           <xendr:format> if $row[$col].is-numeric "ixt4:num-dot-decimal" else none </xendr:format>

               // SCALE NUMBERS TO 6 (MILLIONS) IF FACT IS MONETARY
           <xendr:scale> if $row[$col].is-monetary "6" else none </xendr:scale>

       </xendr:replace>

</td>
```



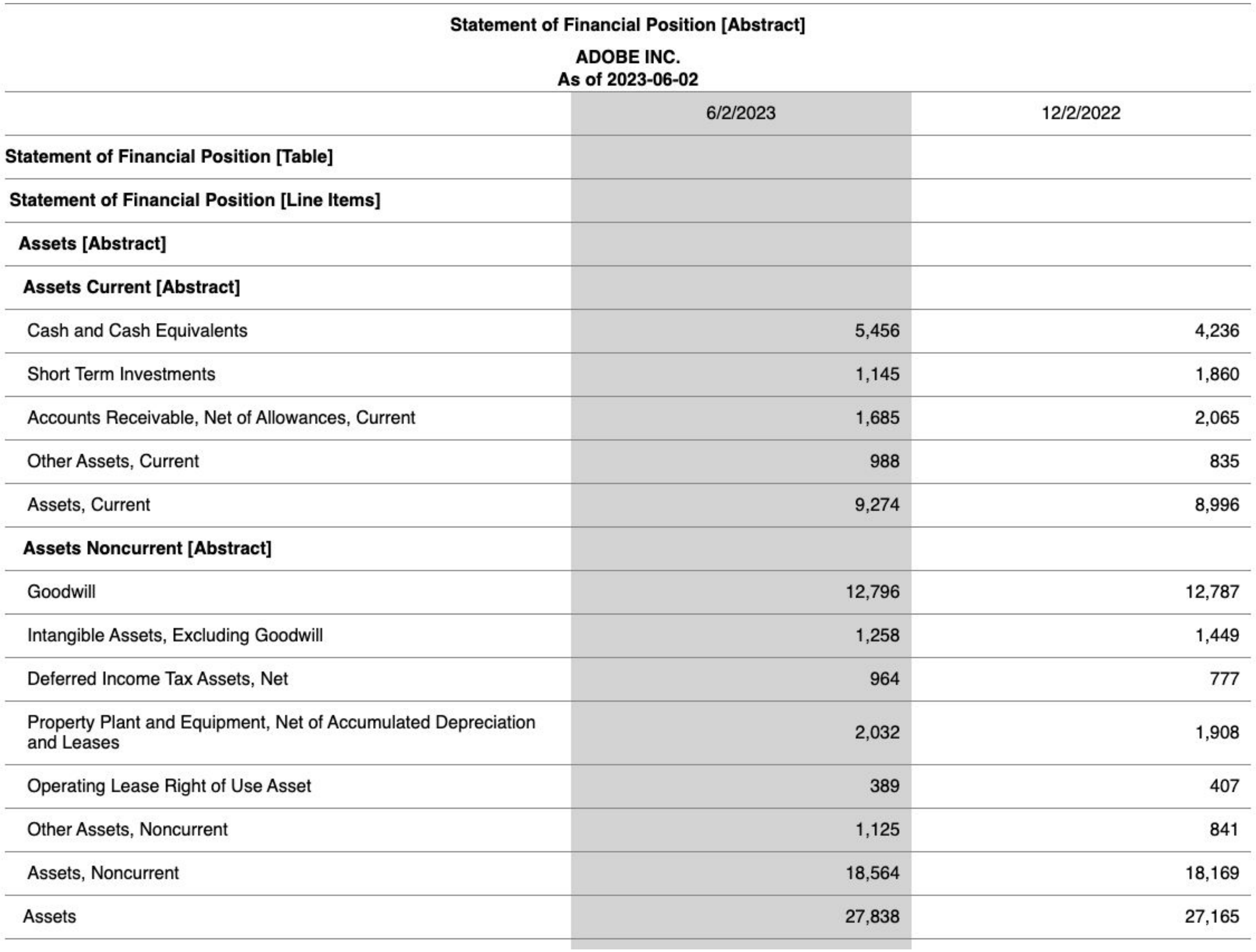
Create Column 2 & 3



```
 // CREATE COL2 & 3
  <td class=" gray-out " >
     <ix:nonfraction unitref="usd" decimals="-6" contextref="c2"
    name="xusss:CashAndCashEquivalentsAtCarryingValue" id="f43-dup-1"
    format="ixt4:num-dot-decimal" scale=" 6 ">5,456</ix:nonfraction>
    </td>

  <td class="" >
     <ix:nonfraction unitref="usd" decimals="-6" contextref="c3"
    name="xusss:CashAndCashEquivalentsAtCarryingValue" id="f139-dup-1"
    format="ixt4:num-dot-decimal" scale=" 6 ">4,236</ix:nonfraction>
  </td>
```



Exclude Empty Rows



Statement of Financial Position [Abstract]

ADOBE INC.
As of 2023-06-02

	6/2/2023	12/2/2022
Statement of Financial Position [Table]		
Statement of Financial Position [Line Items]		
Assets [Abstract]		
Assets Current [Abstract]		
Cash and Cash Equivalents	5,456	4,236
Short Term Investments	1,145	1,860
Accounts Receivable, Net of Allowances, Current	1,685	2,065
Other Assets, Current	988	835
Assets, Current	9,274	8,996
Assets Noncurrent [Abstract]		
Goodwill	12,796	12,787
Intangible Assets, Excluding Goodwill	1,258	1,449
Deferred Income Tax Assets, Net	964	777
Property Plant and Equipment, Net of Accumulated Depreciation and Leases	2,032	1,908
Operating Lease Right of Use Asset	389	407
Other Assets, Noncurrent	1,125	841
Assets, Noncurrent	18,564	18,169
Assets	27,838	27,165

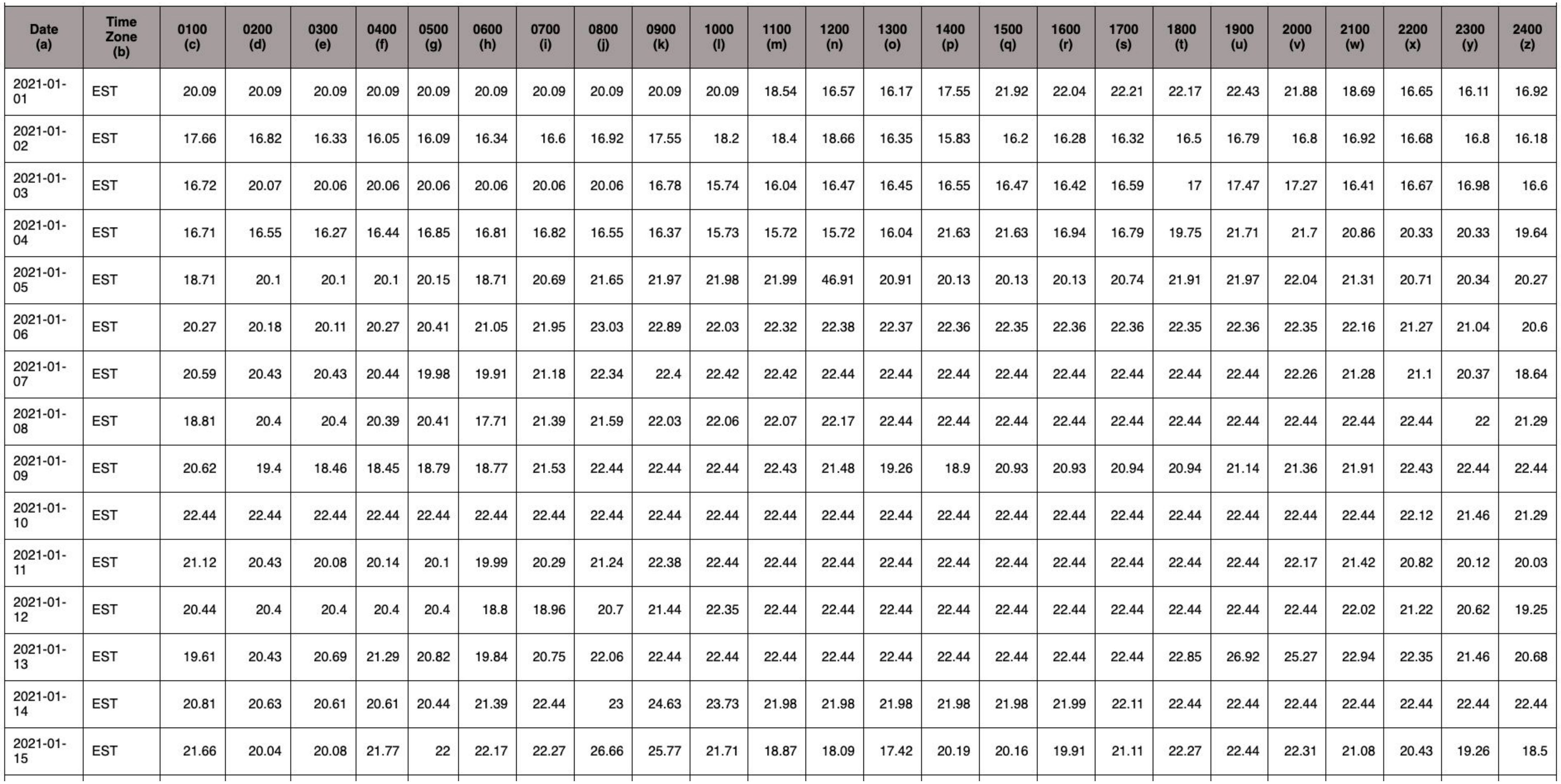
Remove Zero and None Values

```
for $row in $data-labels
```

Becomes

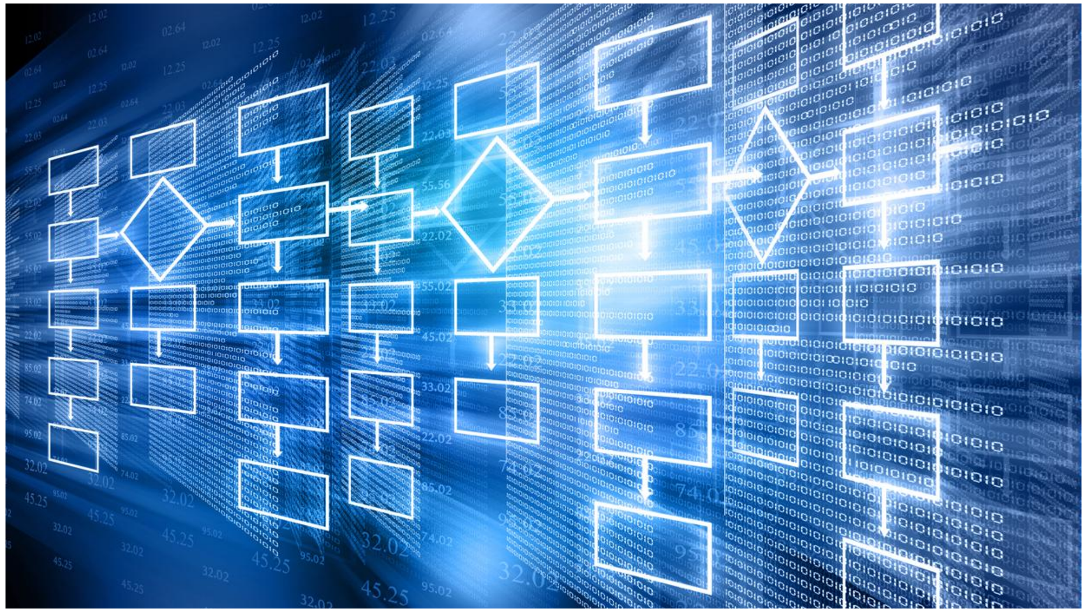
```
for $row in filter $data-labels
            where (($item[3] != none // CHECK IF PRIOR & CURRENT VALUES ARE BOTH ZERO OR NONE AND REMOVE IF TRUE
                   and $item[3] != 0
                   and $item[4] != none
                   and $item[4] != 0)
                 or $item[1].is-abstract) // KEEP THE HEADER
```

Can Render Data Beyond Table Linkbase



Date (a)	Time Zone (b)	0100 (c)	0200 (d)	0300 (e)	0400 (f)	0500 (g)	0600 (h)	0700 (i)	0800 (j)	0900 (k)	1000 (l)	1100 (m)	1200 (n)	1300 (o)	1400 (p)	1500 (q)	1600 (r)	1700 (s)	1800 (t)	1900 (u)	2000 (v)	2100 (w)	2200 (x)	2300 (y)	2400 (z)
2021-01-01	EST	20.09	20.09	20.09	20.09	20.09	20.09	20.09	20.09	20.09	20.09	18.54	16.57	16.17	17.55	21.92	22.04	22.21	22.17	22.43	21.88	18.69	16.65	16.11	16.92
2021-01-02	EST	17.66	16.82	16.33	16.05	16.09	16.34	16.6	16.92	17.55	18.2	18.4	18.66	16.35	15.83	16.2	16.28	16.32	16.5	16.79	16.8	16.92	16.68	16.8	16.18
2021-01-03	EST	16.72	20.07	20.06	20.06	20.06	20.06	20.06	20.06	16.78	15.74	16.04	16.47	16.45	16.55	16.47	16.42	16.59	17	17.47	17.27	16.41	16.67	16.98	16.6
2021-01-04	EST	16.71	16.55	16.27	16.44	16.85	16.81	16.82	16.55	16.37	15.73	15.72	15.72	16.04	21.63	21.63	16.94	16.79	19.75	21.71	21.7	20.86	20.33	20.33	19.64
2021-01-05	EST	18.71	20.1	20.1	20.1	20.15	18.71	20.69	21.65	21.97	21.98	21.99	46.91	20.91	20.13	20.13	20.13	20.74	21.91	21.97	22.04	21.31	20.71	20.34	20.27
2021-01-06	EST	20.27	20.18	20.11	20.27	20.41	21.05	21.95	23.03	22.89	22.03	22.32	22.38	22.37	22.36	22.35	22.36	22.36	22.35	22.36	22.35	22.16	21.27	21.04	20.6
2021-01-07	EST	20.59	20.43	20.43	20.44	19.98	19.91	21.18	22.34	22.4	22.42	22.42	22.44	22.44	22.44	22.44	22.44	22.44	22.44	22.44	22.26	21.28	21.1	20.37	18.64
2021-01-08	EST	18.81	20.4	20.4	20.39	20.41	17.71	21.39	21.59	22.03	22.06	22.07	22.17	22.44	22.44	22.44	22.44	22.44	22.44	22.44	22.44	22.44	22.44	22	21.29
2021-01-09	EST	20.62	19.4	18.46	18.45	18.79	18.77	21.53	22.44	22.44	22.44	22.43	21.48	19.26	18.9	20.93	20.93	20.94	20.94	21.14	21.36	21.91	22.43	22.44	22.44
2021-01-10	EST	22.44	22.44	22.44	22.44	22.44	22.44	22.44	22.44	22.44	22.44	22.44	22.44	22.44	22.44	22.44	22.44	22.44	22.44	22.44	22.44	22.44	22.12	21.46	21.29
2021-01-11	EST	21.12	20.43	20.08	20.14	20.1	19.99	20.29	21.24	22.38	22.44	22.44	22.44	22.44	22.44	22.44	22.44	22.44	22.44	22.44	22.17	21.42	20.82	20.12	20.03
2021-01-12	EST	20.44	20.4	20.4	20.4	20.4	18.8	18.96	20.7	21.44	22.35	22.44	22.44	22.44	22.44	22.44	22.44	22.44	22.44	22.44	22.44	22.02	21.22	20.62	19.25
2021-01-13	EST	19.61	20.43	20.69	21.29	20.82	19.84	20.75	22.06	22.44	22.44	22.44	22.44	22.44	22.44	22.44	22.44	22.44	22.85	26.92	25.27	22.94	22.35	21.46	20.68
2021-01-14	EST	20.81	20.63	20.61	20.61	20.44	21.39	22.44	23	24.63	23.73	21.98	21.98	21.98	21.98	21.98	21.99	22.11	22.44	22.44	22.44	22.44	22.44	22.44	22.44
2021-01-15	EST	21.66	20.04	20.08	21.77	22	22.17	22.27	26.66	25.77	21.71	18.87	18.09	17.42	20.19	20.16	19.91	21.11	22.27	22.44	22.31	21.08	20.43	19.26	18.5

XODEL



What is XODEL?

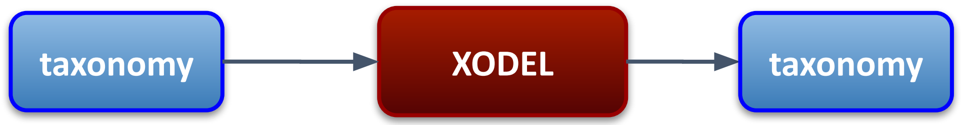
1. XODEL uses the xule syntax to create xbrl taxonomies.
2. XODEL is available as an Arelle plugin
3. XODEL creates a taxonomy as a package
4. XODEL is used to:
   - Publish base taxonomies
   - Generate test cases easily and quickly

XODEL Uses

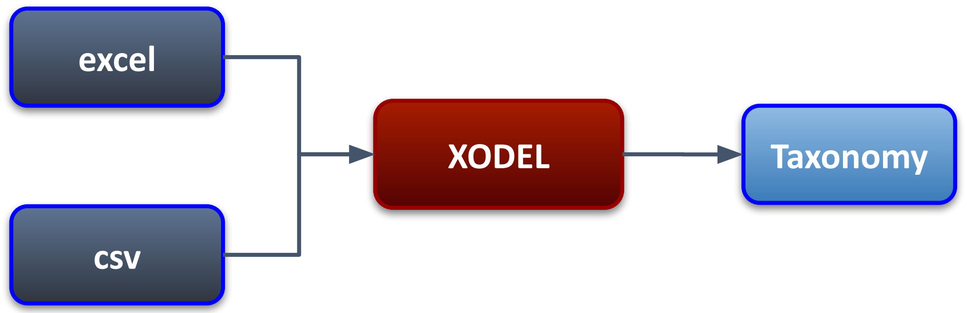
🟦 XBRL Document

⬛ Non XBRL Document

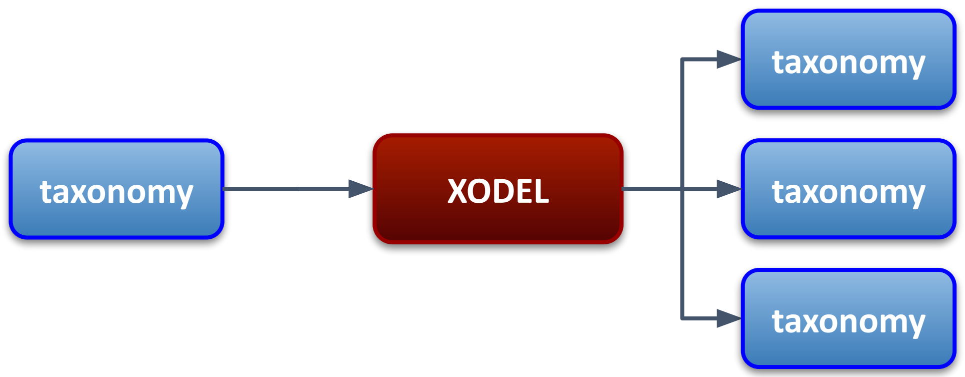
Create publish taxonomy from working taxonomy



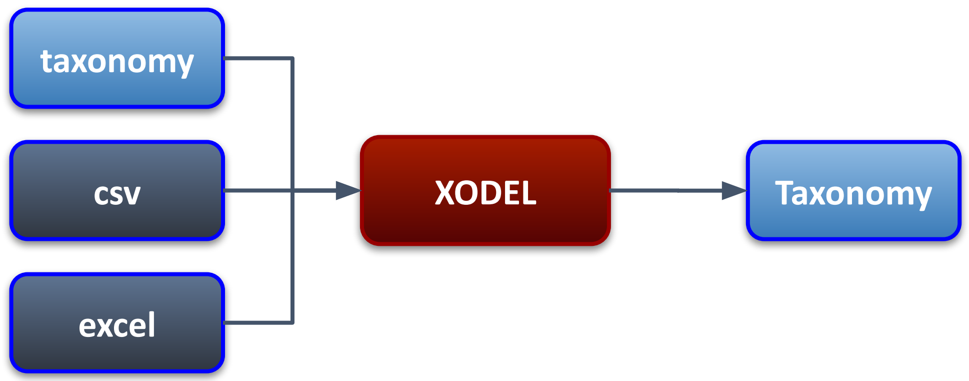
Create taxonomy from other data sources



Change Serialization of Taxonomy

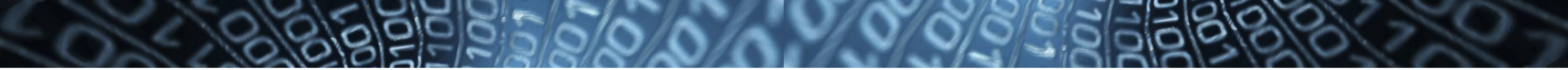


Enrich existing taxonomy with meta-data



Allows the creation of taxonomy objects

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



Example - Create extension concepts

Create an extension taxonomy with the following extension elements, these are defined in extensions.csv

Concept	Label Type	Label
CarpetLoans	label	Carpet Loans
CarpetLoans	documentation	Loans to customers to buy carpet
LoomEquipment	label	Loom Equipment
LoomEquipment	documentation	Equipment that makes carpet
WoolStock	label	Wool Stock
WoolStock	documentation	Wool in inventory to make carpet

Example - Create extension concepts

Add extensions.csv to a variable called $EXTENSION_CONCEPTS

```
constant $EXTENSION_CONCEPTS = csv-data('extensions.csv',true, list(
'string', 'string', 'string'))
```

function

csv-data(file name, has headers, list of field types)

Example - Create extension concepts

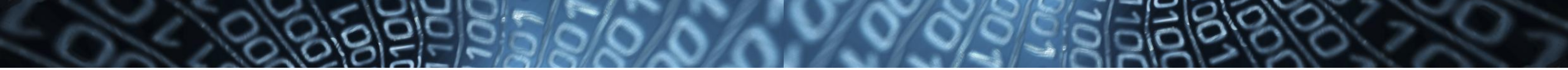
First, we define a new taxonomy package We call the new taxonomy package ‘MyTaxonomy’. This taxonomy imports the FASB taxonomy, the dei taxonomy and the srt taxonomy.

```
constant $EXTENSION_TAXONOMY = 'MyTaxonomy'

output create_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')
document-package-entry-point true
document-package-entry-point-description 'main entry point'
```

Example - Create extension concepts

Next we create the extension concepts based on the Assets concept in the us-gaap taxonomy.

```
constant $US-GAAP = taxonomy('https://fasb.org/elts/us-gaap-2023.xsd')

output create_extension_concepts
$baseAsset = $US-GAAP.concept(Assets)
$UniqueAssets = set(filter $EXTENSION_CONCEPTS returns $item[1])
for $asset in $UniqueAssets
      $asset

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



Example - Create extension labels

Finally, we create the labels based on the data in the csv file.

```
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]
```

Output

MyTaxonomy.zip

abc.xsd

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:link="http://www.xbrl.org/2003/linkbase"
xmlns:abc="http://www.abc.com/2024"
xmlns:xbrli="http://www.xbrl.org/2003/instance"
xmlns:dtr-types="http://www.xbrl.org/dtr/type/2020-01-21"
xmlns:xbrldt="http://xbrl.org/2005/xbrldt"
xmlns:num="http://www.xbrl.org/dtr/type/numeric"
attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="http://www.abc.com/2024">
  <xs:import namespace="http://fasb.org/srt/2023"
schemaLocation="https://fasb.org/elts/us-srt-2023.xsd"/>
  <xs:import namespace="http://fasb.org/us-gaap/2023"
schemaLocation="https://fasb.org/elts/us-gaap-2023.xsd"/>
    <xs:import namespace="http://xbrl.sec.gov/dei/2023"
schemaLocation="ttps://sec.gov/elts/dei-2023.xsd"/>
  <xs:annotation>
    <xs:appinfo>
      <link:linkbaseRef xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:arcrole="http://www.w3.org/1999/xlink/properties/linkbase"
xlink:href="abc_lab.xml"
xlink:role="http://www.xbrl.org/2003/role/labelLinkbaseRef"
xlink:type="simple"/>
                        </xs:appinfo>
      </xs:annotation>

<xs:element id="CarpetLoans" abstract="false" name="CarpetLoans"
nillable="true" xbrli:periodType="instant" xbrli:balance="debit"
substitutionGroup="xbrli:item" type="xbrli:monetaryItemType"/>
<xs:element id="LoomEquipment" abstract="false"
name="LoomEquipment" nillable="true" xbrli:periodType="instant"
xbrli:balance="debit" substitutionGroup="xbrli:item"
type="xbrli:monetaryItemType"/>
<xs:element id="WoolStock" abstract="false" name="WoolStock"
nillable="true" xbrli:periodType="instant" xbrli:balance="debit"
substitutionGroup="xbrli:item" type="xbrli:monetaryItemType"/>
</xs:schema>
```

abc_lab.xml

```
<link:linkbase
xmlns:link="http://www.xbrl.org/2003/linkbase"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-in
stance"
xsi:schemaLocation="http://www.xbrl.org/2003/link
base
http://www.xbrl.org/2003/xbrl-linkbase-2003-12-31.
xsd">
<link:labelLink
xlink:role="http://www.xbrl.org/2003/role/link"
xlink:type="extended">
  <link:loc xlink:type="locator"
xlink:label="CarpetLoans"
xlink:href="abc.xsd#CarpetLoans"/>
  <link:label id="CaprpetLoans"
xlink:label="lab_CarpetLoans"
xlink:role="http://www.xbrl.org/2003/role/label"
xlink:type="resource"
xmlns:xml="http://www.w3.org/XML/1998/namesp
ace" xml:lang="en-US">Carpet Loans</link:label>
 <link:labelArc
xlink:arcrole="http://www.xbrl.org/2003/arcrole/co
ncept-label" xlink:from="CarpetLoans"
xlink:to="lab_CarpetLoans" xlink:type="arc"
order="1"/>
```

catalog.xml

taxonomyPackage.xml

Advantages of XODEL

1. Can document taxonomy creation steps.
2. Can easily recreate a taxonomy with updated information
3. Reduce manual entry
4. Automatically generate definition linkbase
5. Can use with multiple tools
6. Uses XULE expressions
7. Intricate control over taxonomy serialization

Large Data Files

Aggregators

Windows

Resources

Documentation

- https://xule.org
- https://xince.org
- https://xendr.org
- https://xodel.org

Code Repository

- https://github.com/xbrlus/xule

Unit Tests

- https://github.com/xbrlus/xule/tree/main/unitTests/