Getting Started

- The name of the ISO ERC file must be a string that starts with two uppercase letters representing the line of business code, followed by a space, then two more uppercase letters representing the layer code, another space, then eight digits representing the effective date, another space, then a capital letter V followed by two digits representing the version number. The last part of the string is optional and consists of a space followed by one to eight uppercase letters or digits representing the program code.

eg. AU CW 20220901 V01

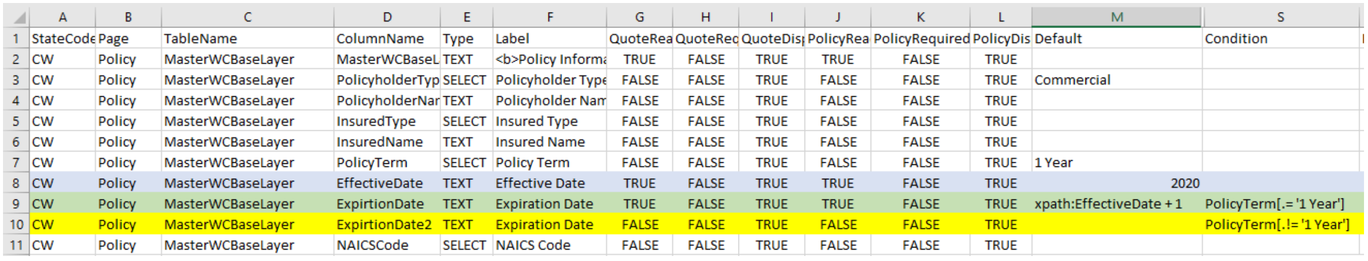
- Generally, UI is filled from top to bottom. When you fill a field, the following fields which have their default value dependent on the field will have their value set automatically. However, if you want a field to be infillable and store calculated result of the other field, it needs to be done in rules. Default column can be a value or start with xpath; hence dependent on other fields.

- The program code needs to be consistent with file name.

- Follow normal variable name conventions for table name, column name, domain table name and rate table name and rule name. For every names that are not displayed on the UI, follow this rule: start with upper case letter, followed by only letters (upper or lower case) and numbers. Avoid space, Apostrophe symbol('). They could potentially cause issues.

- Field value dependency relationship between drop-downs is set via domain table key columns and related fields table.

- The read-only field shows when the value of Policy Term is ‘1 Year’, and use xpath to set its value. When the value of Policy Term is not '1 Year", we display the editable field. We are using **XPath 3.1**.



	A	B	C	D	E	F	G	H	I	J	K	L	M	S
1	StateCode	Page	TableName	ColumnName	Type	Label	QuoteRea	QuoteReq	QuoteDis	PolicyRea	PolicyRequired	PolicyDis	Default	Condition
2	CW	Policy	MasterWCBaseLayer	MasterWCBaseL	TEXT	<b>Policy Informa	TRUE	FALSE	TRUE	TRUE	FALSE	TRUE		
3	CW	Policy	MasterWCBaseLayer	PolicyholderTyp	SELECT	Policyholder Type	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	Commercial	
4	CW	Policy	MasterWCBaseLayer	PolicyholderNar	TEXT	Policyholder Nam	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE		
5	CW	Policy	MasterWCBaseLayer	InsuredType	SELECT	Insured Type	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE		
6	CW	Policy	MasterWCBaseLayer	InsuredName	TEXT	Insured Name	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE		
7	CW	Policy	MasterWCBaseLayer	PolicyTerm	SELECT	Policy Term	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	1 Year	
8	CW	Policy	MasterWCBaseLayer	EffectiveDate	TEXT	Effective Date	TRUE	FALSE	TRUE	TRUE	FALSE	TRUE	2020	
9	CW	Policy	MasterWCBaseLayer	ExpirtionDate	TEXT	Expiration Date	TRUE	FALSE	TRUE	TRUE	FALSE	TRUE	xpath:EffectiveDate + 1	PolicyTerm[.= '1 Year']
10	CW	Policy	MasterWCBaseLayer	ExpirtionDate2	TEXT	Expiration Date	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE		PolicyTerm[.!= '1 Year']
11	CW	Policy	MasterWCBaseLayer	NAICSCode	SELECT	NAICS Code	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE		

- The sequence in the field list is not defined. It shall be assigned value based on, how it is to be enabled in UI. The convention is to add 10- to each. Here, the intension is to leave some buffer to insert fields in future. All fields in the field list should have sequence.

- We shall always add premium fields into each level - *Premium, PolicyTermPremium*. Field *Premium* saves final annual premium. Field *PolicyTermPremium* is displayed on UI but not set in rules. There is logic to set them after rules. Rating worksheet can give better insights. All the fields in premium summary shall be readonly. They are used to display calculation process and result, not for user to input.

Premium	Premium	decimal	👤
PolicyTermPremium	PolicyTermPremium	decimal	👤

- Avoid field names ending with premium. In ISO, this conveys real premium and some code rely on this convention.
- In ISO rules, for each level, normally at the end you will assign a value to the Premium field of that level. Then in some upper level, you sum up all the Premium in lower levels.
- Do not include attribute of tag which does not contain any value.

eg. *FromParam* below:

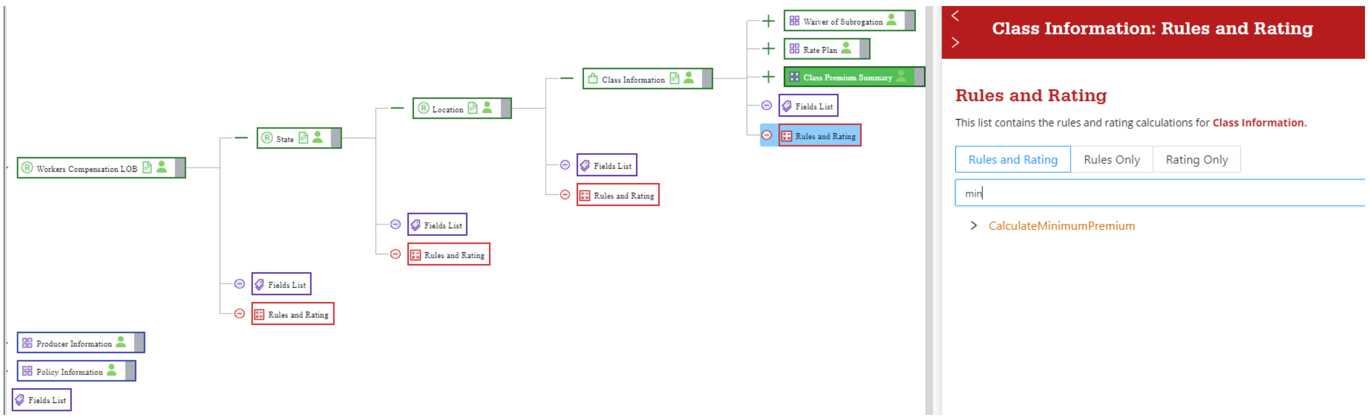
```
<rul:Value ToDataDef="YearsOfOperation" FromDataDef="YearsOfOperation"
Type="integer" AllowNullReturn="true" FromParam=""/>
```

- The *Type* attribute of a rule is the type of data it returns. It shall be *None*, if it does not return anything. But, if it does return, it could not be empty.

eg. Since below rul:RunRule runs inside rul:Sum, it should return something for the sum rule to sum. It cannot be empty.

```
<rul:Sum ToDataDef="ErcCalculatedTotalPremium">
    <rul:RunRule Type=""
FileName="MasterWCBaseLayerLocationClassificationWCLiabilityRules"
Rule="WCLiabilityPremium" ClearCache="true"/>
</rul:Sum>
```

- To get value for fields at current level for calculation, we use xpath. eg. Rule *CalculateMinimumPremium* in written at level *Class Information* coverage group.



Class Information: Rules and Rating

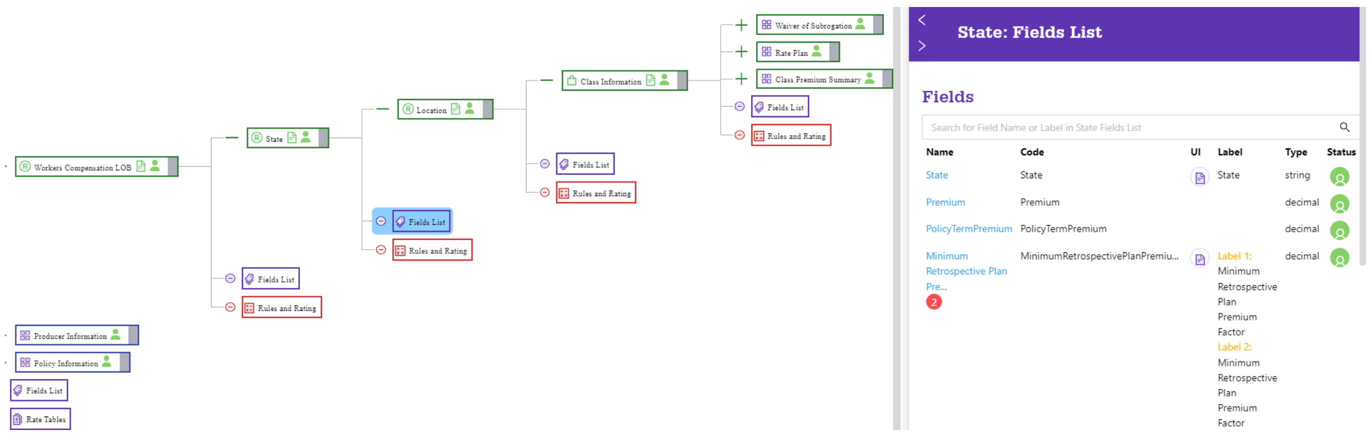
Rules and Rating

This list contains the rules and rating calculations for **Class Information.**

CalculateMinimumPremium

Here, *MinimumRetrospectivePlanPremiumFactor* field is present at level *State* risk. So, we refer it with xpath *../../../../MinimumRetrospectivePlanPremiumFactor*.

../../ denotes one level up.

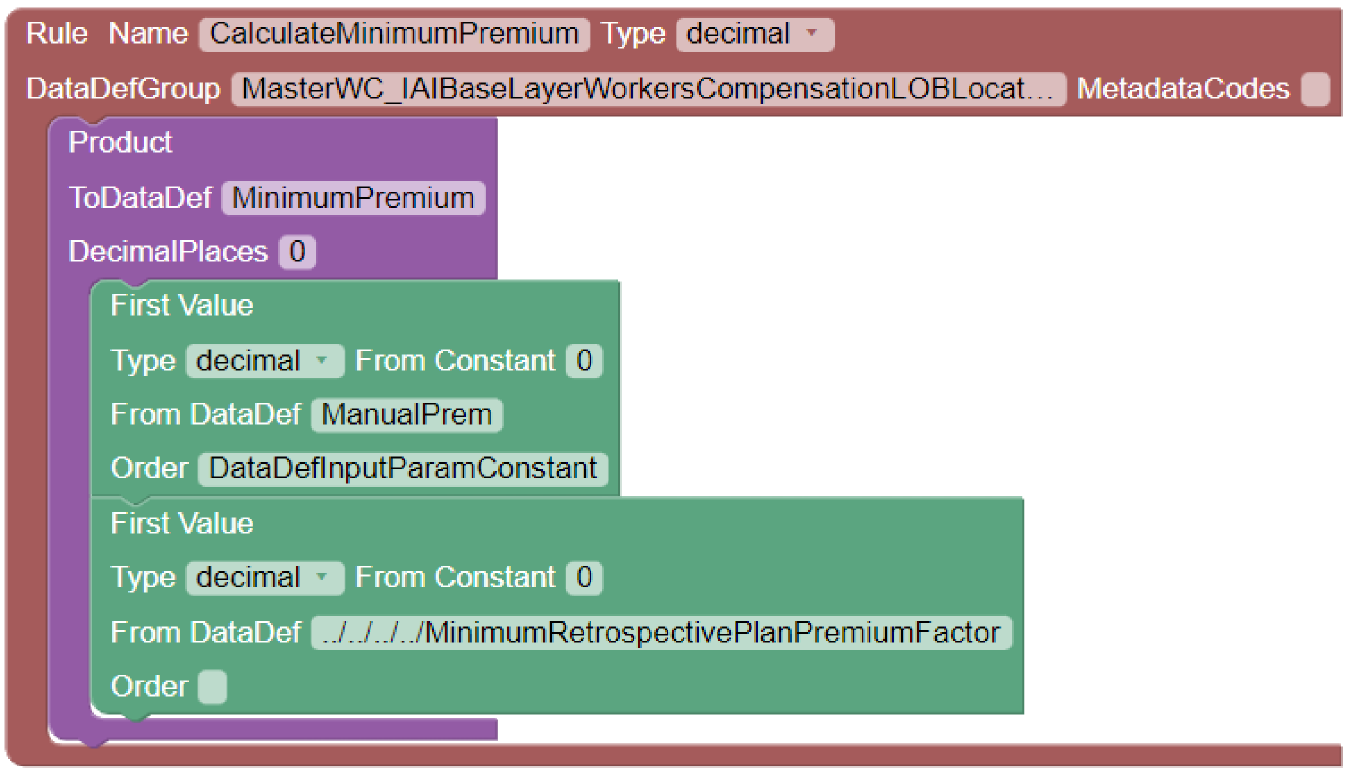


State: Fields List

Fields

Search for Field Name or Label in State Fields List

Name	Code	UI	Label	Type	Status
State	State	🗎	State	string	●
Premium	Premium			decimal	●
PolicyTermPremium	PolicyTermPremium			decimal	●
Minimum Retrospective Plan Pre... 2	MinimumRetrospectivePlanPremiu...	🗎	Label 1: Minimum Retrospective Plan Premium Factor Label 2: Minimum Retrospective Plan Premium Factor	decimal	●



```
Rule Name CalculateMinimumPremium Type decimal
DataDefGroup MasterWC_IAIBaseLayerWorkersCompensationLOBLocat... MetadataCodes
    Product
    ToDataDef MinimumPremium
    DecimalPlaces 0
        First Value
        Type decimal From Constant 0
        From DataDef ManualPrem
        Order DataDefInputParamConstant
        First Value
        Type decimal From Constant 0
        From DataDef ../../../../MinimumRetrospectivePlanPremiumFactor
        Order
```

- In attribute *ToDataDef*, xpath should not contain predicates([,]). Xpath should point to current level, or child elements located within current element. Locations beyond(upper level) should not be given.

The Rating Structure

Policy data :

The rules will execute against policy data file.

We can upload policy data using *Sample Policies* button.

Similarly, policy data file can be downloaded with *InsureMO JSON*, *DIPL JSON*, *DIPL XML* buttons.

Below are contents from policy data file(using *DIPL XML* button) :

```
<Policy>
    <State>
        <StateCode Type="string">US</StateCode>
    </State>
    <MasterWCBaseLayerTable>
        <MasterWCBaseLayer>
            <__ProductCode Type="string">ISO_WO_US_WCBASELA</__ProductCode>
            <__ProductElementCode
Type="string">ISO_WO_US_WCBASELA</__ProductElementCode>
            <PolicyholderType>Commercial</PolicyholderType>
            <PolicyTerm>1 Year</PolicyTerm>
            <EffectiveDate>2023-03-23</EffectiveDate>
            <ExpirtionDate>2024-03-23</ExpirtionDate>
            <InsuredType>LLC</InsuredType>
            <State>AS</State>
            <City>Jber</City>
            <Zip>99508</Zip>
            <MasterWCBaseLayerWorkerCompLOBTable>
                <MasterWCBaseLayerWorkerCompLOB>
                    <CoverageType>2</CoverageType>
            ...
            ...
```

Policy data file begins with *Policy* node. Wherever requirement is to have one-to-many relation, file has following structure : *nameLayerTable*(MasterWCBaseLayerTable), followed by *nameLayer*(MasterWCBaseLayer).

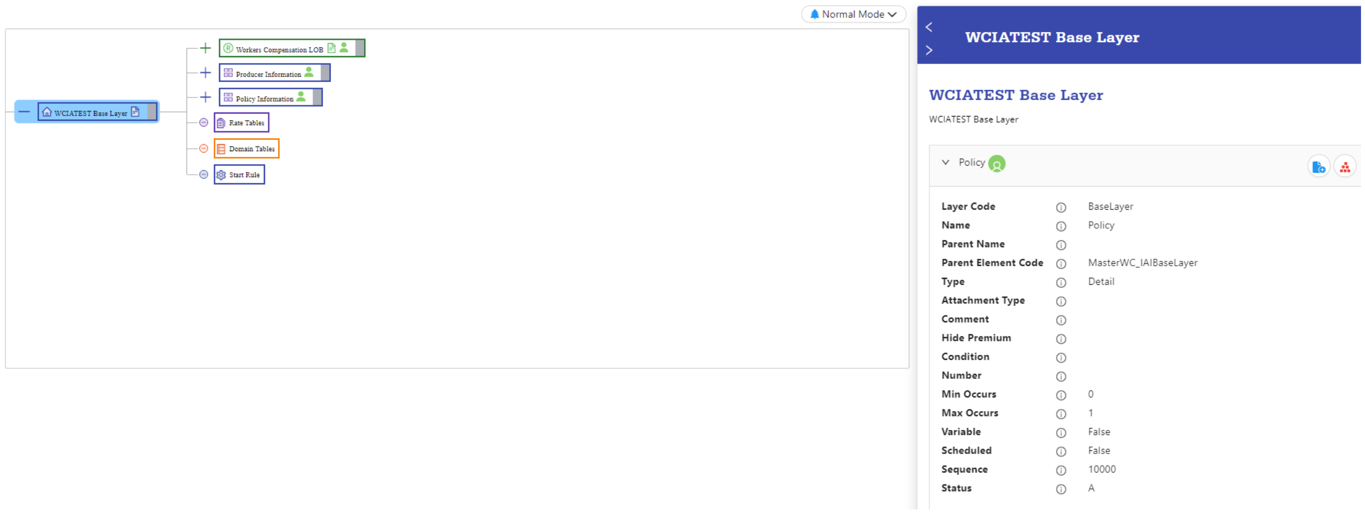
```
<MasterWCBaseLayerTable>
        <MasterWCBaseLayer>
```

We could have many *nameLayer* under *nameLayerTable*.

```
            <MasterWCBaseLayerWorkerCompLOBTable>
                <MasterWCBaseLayerWorkerCompLOB>
                ...
                ...
                </MasterWCBaseLayerWorkerCompLOB>
                <MasterWCBaseLayerWorkerCompLOB>
                ...
                ...
            </MasterWCBaseLayerWorkerCompLOB>
 </MasterWCBaseLayerWorkerCompLOBTable>
```

BaseLayer

For a new project, tree structure already has *Base Layer* node created.



WCIATEST Base Layer

WCIATEST Base Layer

WCIATEST Base Layer

Policy

Layer Code	BaseLayer
Name	Policy
Parent Name	
Parent Element Code	MasterWC_IAIBaseLayer
Type	Detail
Attachment Type	
Comment	
Hide Premium	
Condition	
Number	
Min Occurs	0
Max Occurs	1
Variable	False
Scheduled	False
Sequence	10000
Status	A

Below few lines are standardised for every rule file (eg. MasterWCBaseLayerRules.Rule.xml),

```
<rul:Rules xmlns:rul="http://www.verisk.com/iso/erc/Rule">
<rul:MetaData>
  <rul:MetaDataCode>RulesetTypeCountrywide</rul:MetaDataCode>
</rul:MetaData>
```

In below script, *Name*(Name="CalculateTotalPremium") denotes name of the rule. If the rule has return type, it can be specified using *Type*(Type="decimal"). *DataDefGroup*(DataDefGroup="MasterWCBaseLayer") specifies the current level at which we are executing the rule.

```
<rul:Rule Name="CalculateTotalPremium" Type="decimal"
DataDefGroup="MasterWCBaseLayer" MetadataCodes="">
```

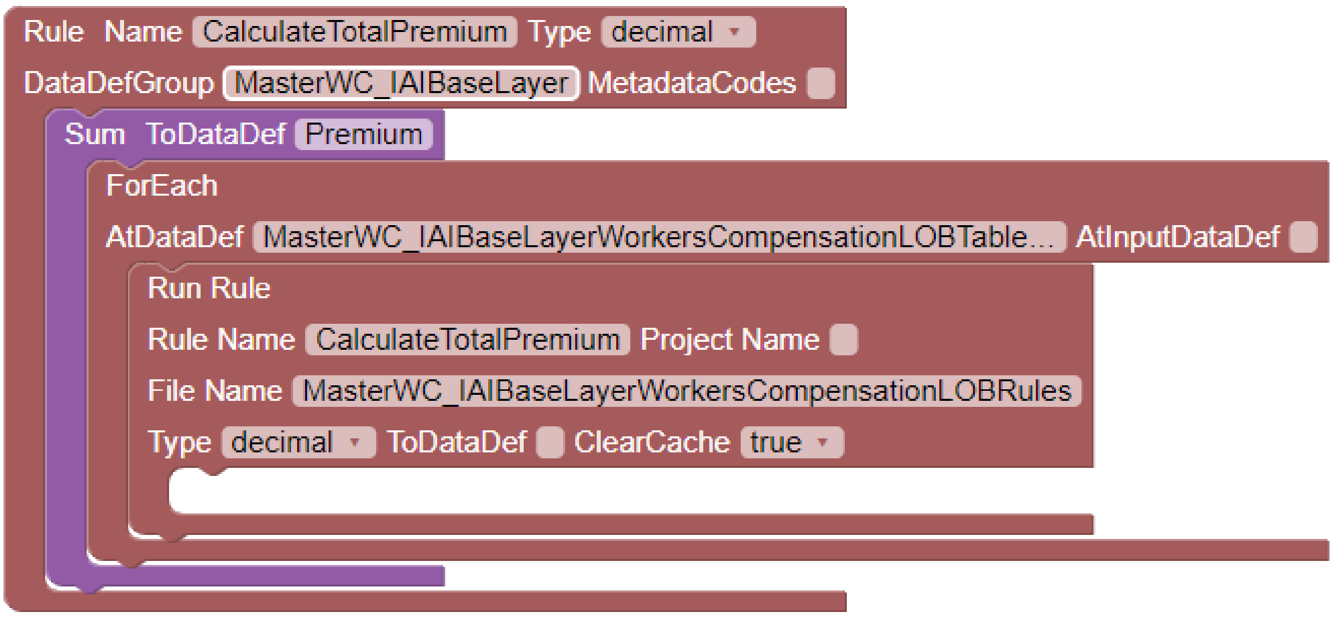
Then, we go to the next level *AtDataDef*(AtDataDef="MasterWCBaseLayerWorkerCompLOBTable/MasterWCBaseLayerWorkerCompLOB") and run rule *Rule*(Rule="CalculateTotalPremium") in file *FileName*(FileName="MasterWCBaseLayerWorkerCompLOBRules")

```
    <rul:Sum ToDataDef="Premium">
      <rul:ForEach
AtDataDef="MasterWCBaseLayerWorkerCompLOBTable/MasterWCBaseLayerWorkerCompLOB">
        <rul:RunRule Type="decimal" FileName="MasterWCBaseLayerWorkerCompLOBRules"
Rule="CalculateTotalPremium" ClearCache="true"/>
      </rul:ForEach>
    </rul:Sum>
```

In this file, we are having a rule(Name="CalculateTotalPremium"), which is making call to rule in another file(FileName="MasterWCBaseLayerWorkerCompLOBRules" Rule="CalculateTotalPremium"). We can make call to multiple rules from a same file.

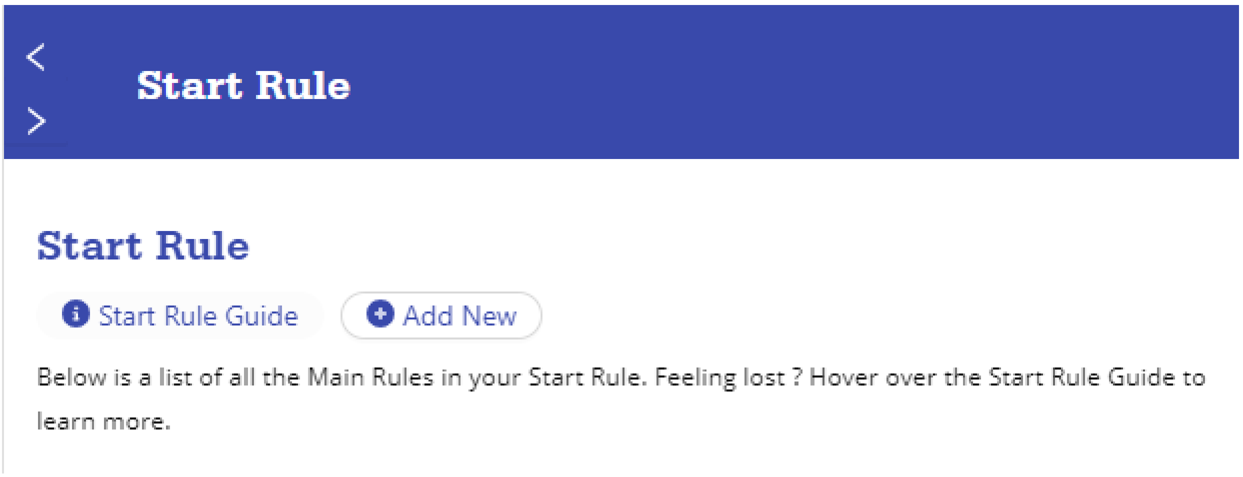
The last rule statement(rul:Sum) within rule(rul:Rule) specifies the operation. The calculated result will be stored in *ToDataDef*(ToDataDef="Premium") field of *DataDefGroup*(DataDefGroup="MasterWCBaseLayer").

```
  <rul:Rule Name="CalculateTotalPremium" Type="decimal"
DataDefGroup="MasterWCBaseLayer" MetadataCodes="">
    <rul:Sum ToDataDef="Premium">
      <rul:ForEach
AtDataDef="MasterWCBaseLayerWorkerCompLOBTable/MasterWCBaseLayerWorkerCompLOB">
        <rul:RunRule Type="decimal" FileName="MasterWCBaseLayerWorkerCompLOBRules"
Rule="CalculateTotalPremium" ClearCache="true"/>
      </rul:ForEach>
    </rul:Sum>
  </rul:Rule>
```


Rule Name CalculateTotalPremium Type decimal
DataDefGroup MasterWC_IAIBaseLayer MetadataCodes
Sum ToDataDef Premium
ForEach
AtDataDef MasterWC_IAIBaseLayerWorkersCompensationLOBTable... AtInputDataDef
Run Rule
Rule Name CalculateTotalPremium Project Name
File Name MasterWC_IAIBaseLayerWorkersCompensationLOBRules
Type decimal ToDataDef ClearCache true

Start Rule

Rule execution always starts from *Overall Rating.Rule.xml file*. This part is included in **Start Rule**.


Start Rule

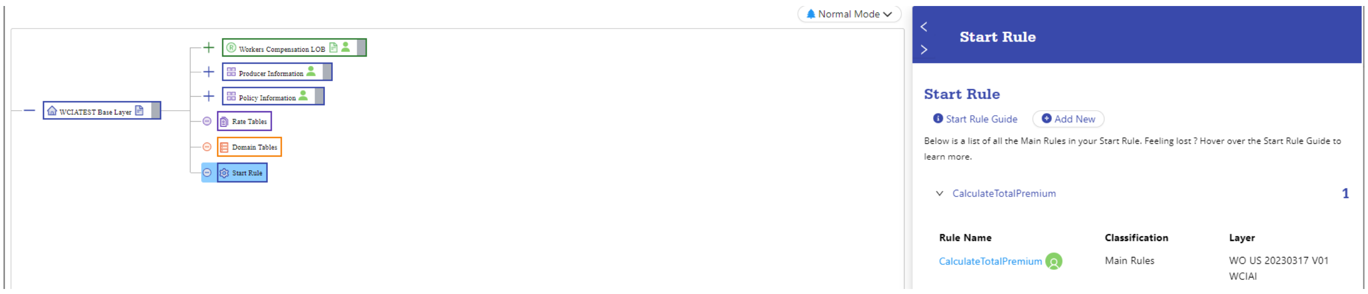
Start Rule

Start Rule Guide Add New

Below is a list of all the Main Rules in your Start Rule. Feeling lost ? Hover over the Start Rule Guide to learn more.

Below contents of file are standardised :

```
 <rul:Rules xmlns:rul="http://www.verisk.com/iso/erc/Rule">
 <rul:MetaData>
   <rul:MetaDataCode>RulesetTypeSystem</rul:MetaDataCode>
 </rul:MetaData>
 <rul:Default>
   <rul:Sequence>
     <rul:Constant Type="integer" ToDataDef="Renewal">0</rul:Constant>
     <rul:Constant Type="string" ToDataDef="State/Code">CW</rul:Constant>
     <rul:Constant Type="string" ToDataDef="State/Name">Workmen's Compensation
Base Layer</rul:Constant>
     <rul:DateAdd ToDataDef="ExpDate" UnitType="Years">
       <rul:Value Type="dateTime" FromDataDef="EffDate"/>
       <rul:Constant Type="integer">1</rul:Constant>
     </rul:DateAdd>
     <rul:Locate AtOutputDataDef="Policy" OutputAction="Append">
     <rul:Sequence/>
     </rul:Locate>
     </rul:Sequence>
 </rul:Default>
</rul:Rules>
```



Start Rule

Start Rule

Below is a list of all the Main Rules in your Start Rule. Feeling lost ? Hover over the Start Rule Guide to learn more.

CalculateTotalPremium 1

Rule Name	Classification	Layer
CalculateTotalPremium	Main Rules	WO US 20230317 V01 WCIAI

The only variable content is as follows :

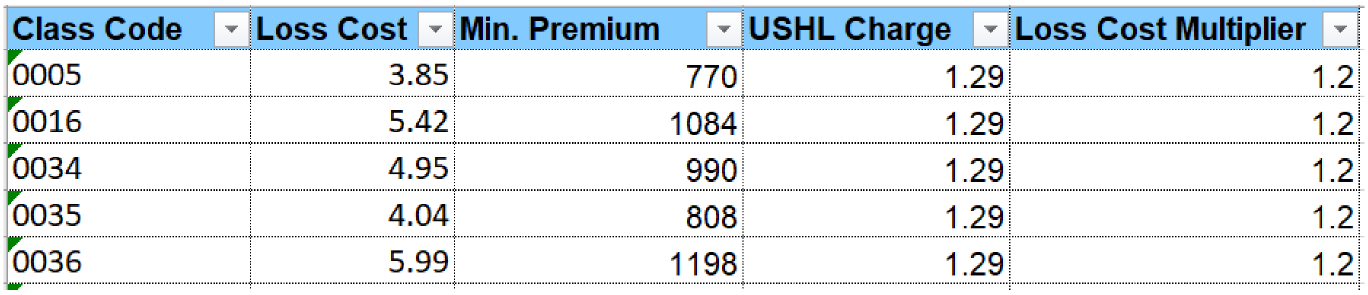
```
 <rul:ForEach AtDataDef="MasterWCBaseLayerTable/MasterWCBaseLayer">
  <rul:RunRule Type="none" FileName="MasterWCBaseLayerRules"
Rule="CalculateTotalPremium" ClearCache="true"/>
</rul:ForEach>
```

In above example, it is used to call *CalculateTotalPremium* rule.

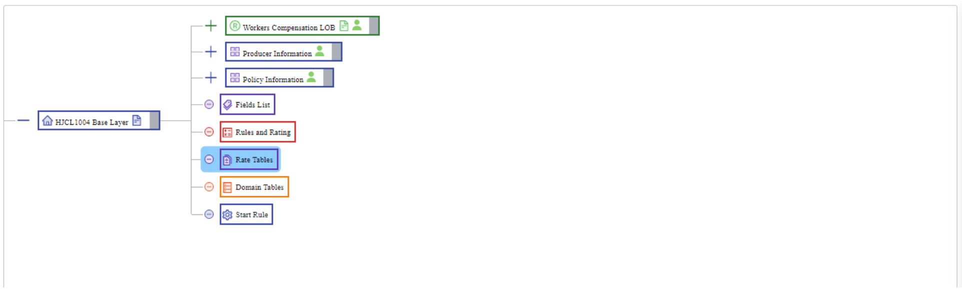
Rate Tables

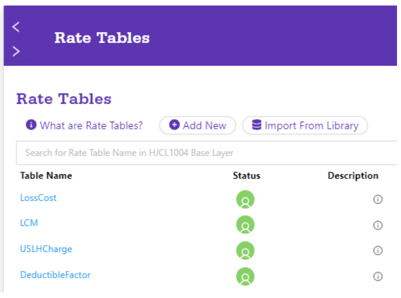
It is used to map a value, based on corresponding key. Product engine does not support one key column mapped to mutiple value columns.

eg. Here, 4 different rate tables have to be created.



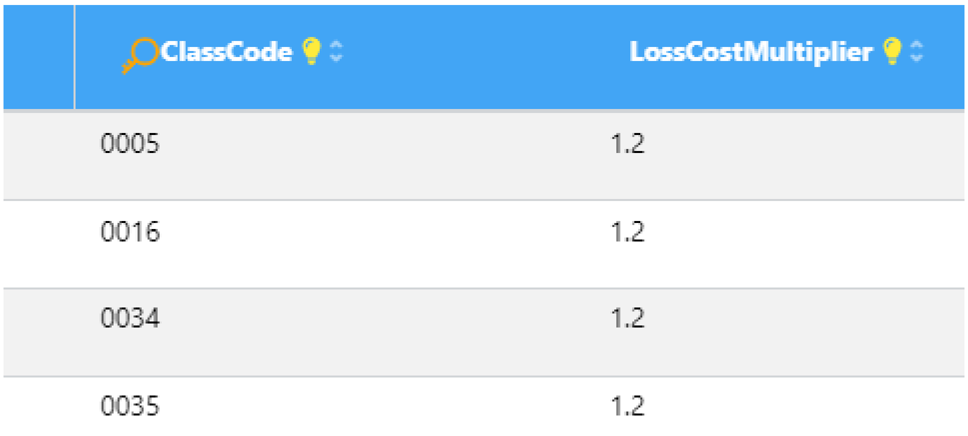
Class Code	Loss Cost	Min. Premium	USHL Charge	Loss Cost Multiplier
0005	3.85	770	1.29	1.2
0016	5.42	1084	1.29	1.2
0034	4.95	990	1.29	1.2
0035	4.04	808	1.29	1.2
0036	5.99	1198	1.29	1.2





Rate Tables

Table Name	Status	Description
LossCost		
LCM		
USLHCharge		
DeductibleFactor		

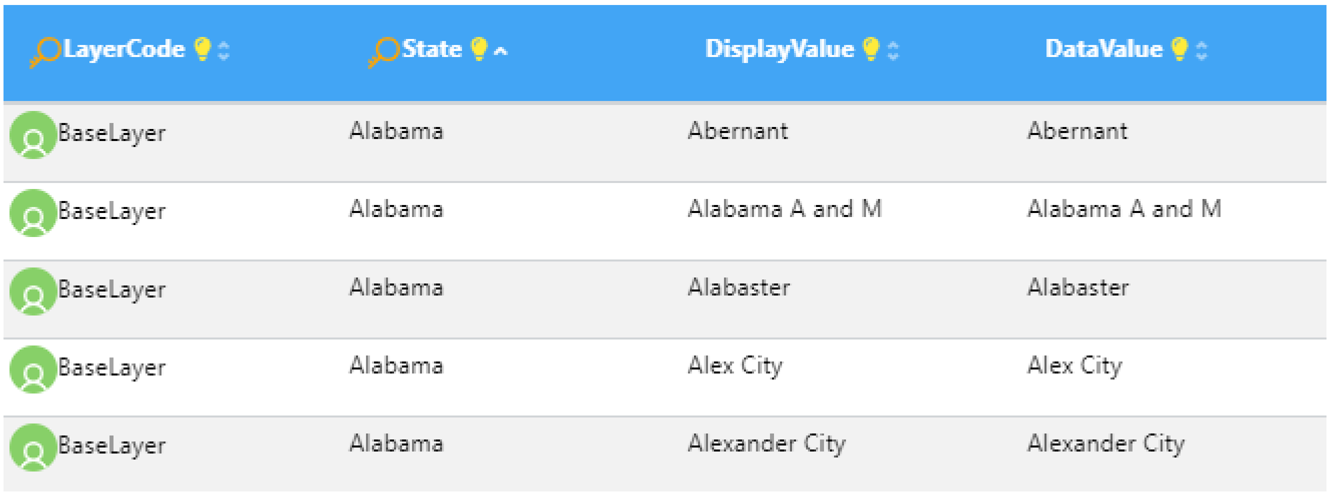


	ClassCode	LossCostMultiplier
	0005	1.2
	0016	1.2
	0034	1.2
	0035	1.2

Domain Tables

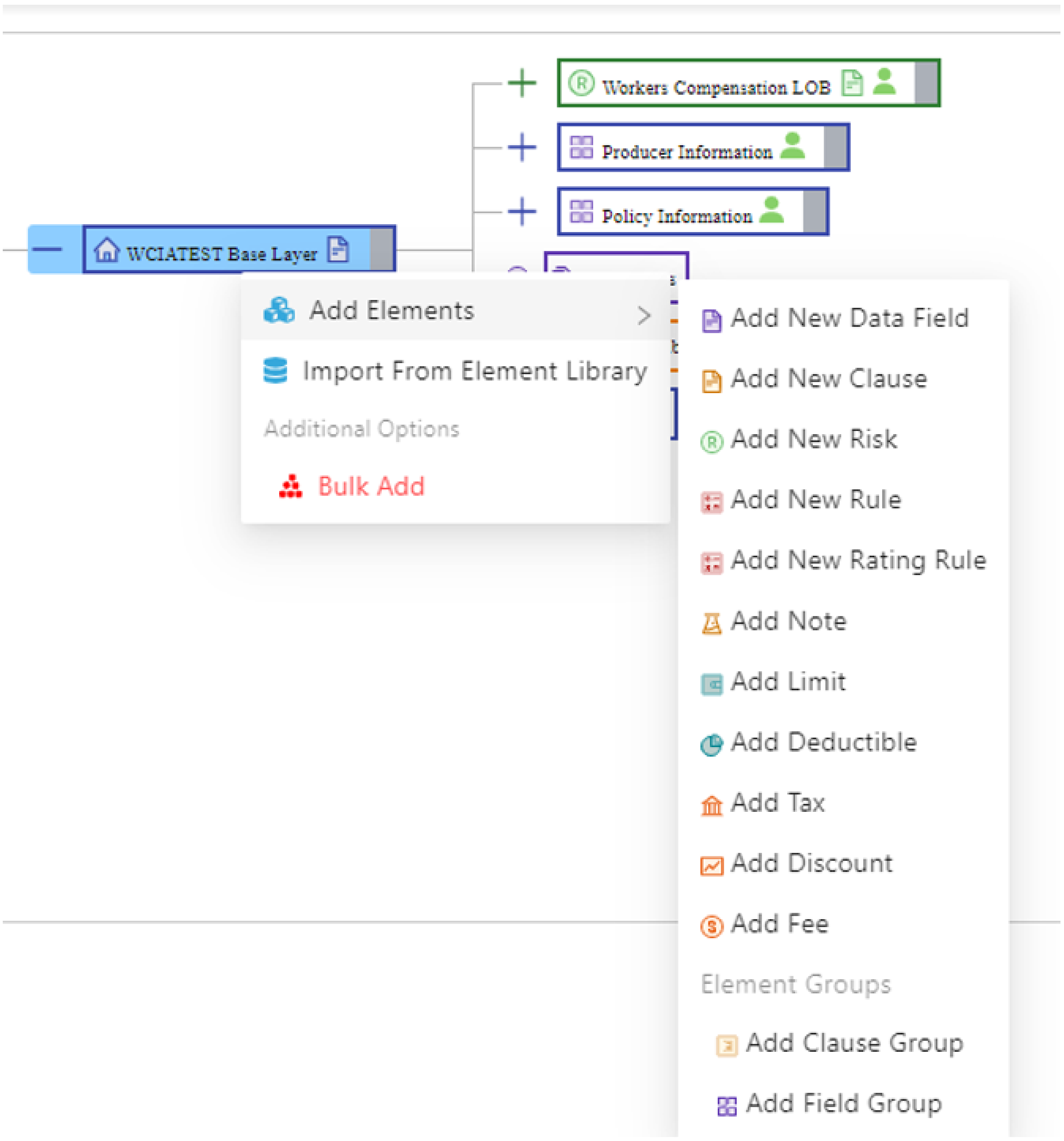
With domain tables, from a drop-down list user can select a value from multiple values.

eg. Below, user has to select a City. So, based on State selection, a drop-down of cities included within that State is displayed to user. Based on user's selection(*display value*), corresponding mapped *data value* is passed to script in backend.



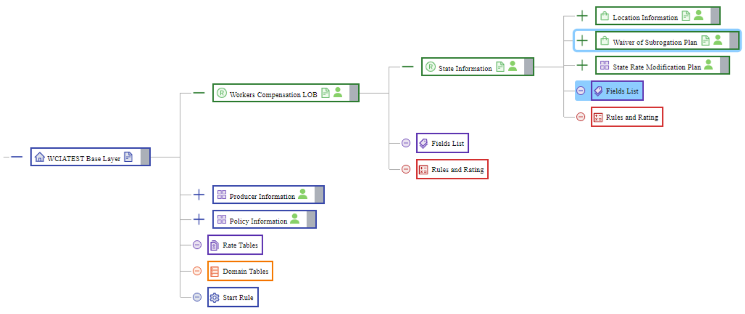
LayerCode	State	DisplayValue	DataValue
BaseLayer	Alabama	Abernant	Abernant
BaseLayer	Alabama	Alabama A and M	Alabama A and M
BaseLayer	Alabama	Alabaster	Alabaster
BaseLayer	Alabama	Alex City	Alex City
BaseLayer	Alabama	Alexander City	Alexander City

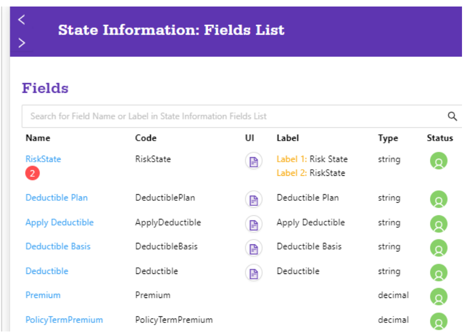
To every node, new Risk, Coverage, Data Field, Rule and Rating can be added using,



Data Field

It is used to add label field(variable). This can be used to input value by user, store output result or reuse variable for calculation.




State Information: Fields List

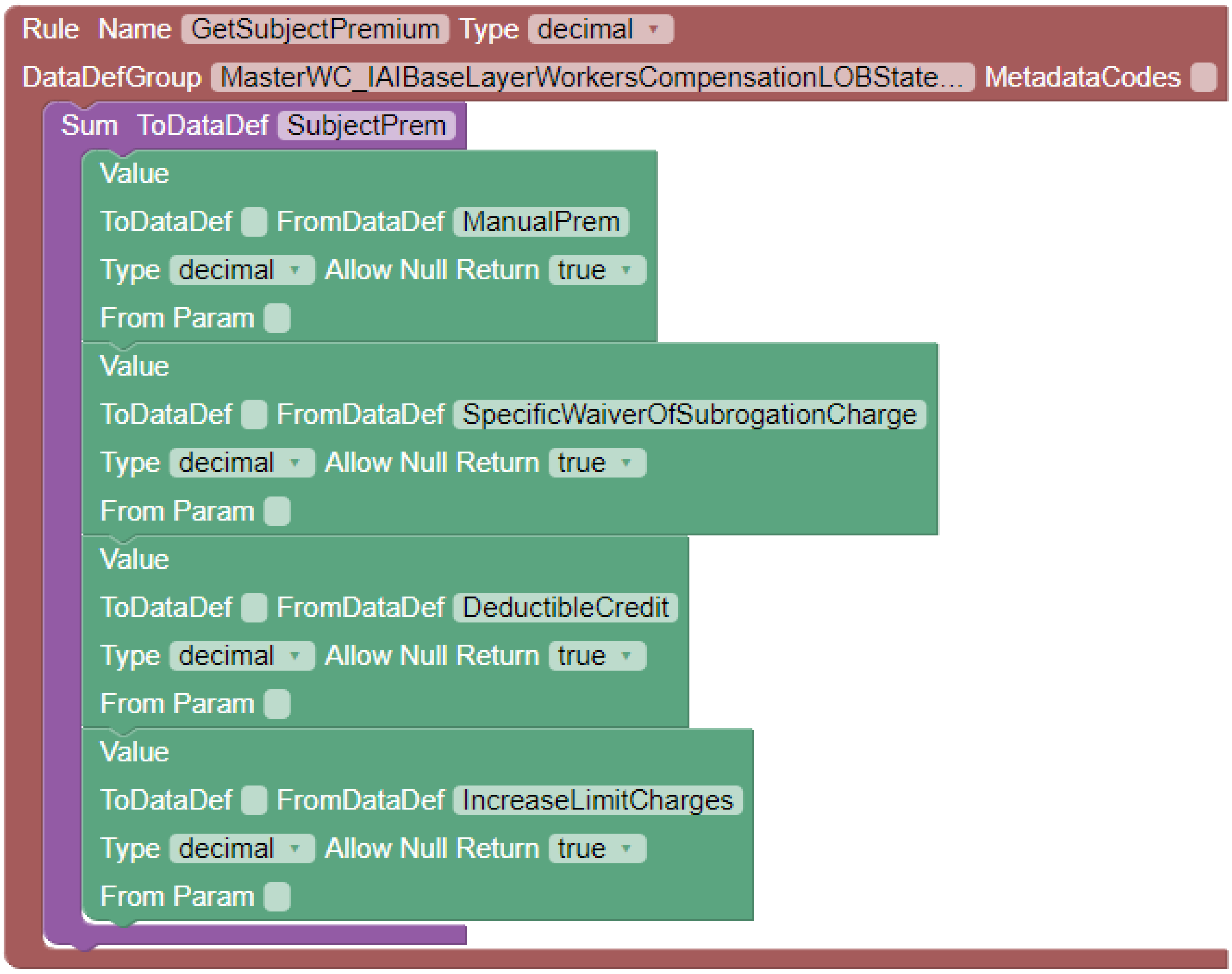
Fields

Search for Field Name or Label in State Information Fields List

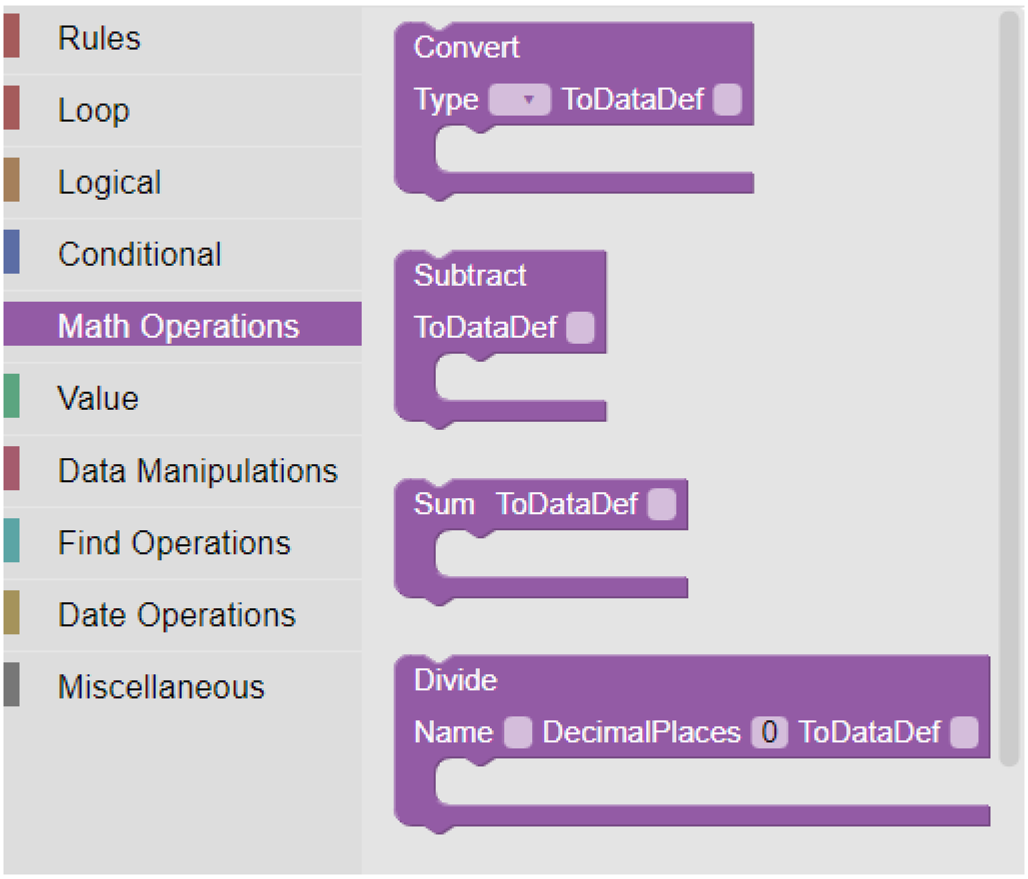
Name	Code	UI	Label	Type	Status
RiskState 2	RiskState		Label 1: Risk State Label 2: RiskState	string	
Deductible Plan	DeductiblePlan		Deductible Plan	string	
Apply Deductible	ApplyDeductible		Apply Deductible	string	
Deductible Basis	DeductibleBasis		Deductible Basis	string	
Deductible	Deductible		Deductible	string	
Premium	Premium			decimal	
PolicyTermPremium	PolicyTermPremium			decimal	

Rules and Rating

It is used to write rules and logic. A set of operators is included. All calculation part is implemented here.

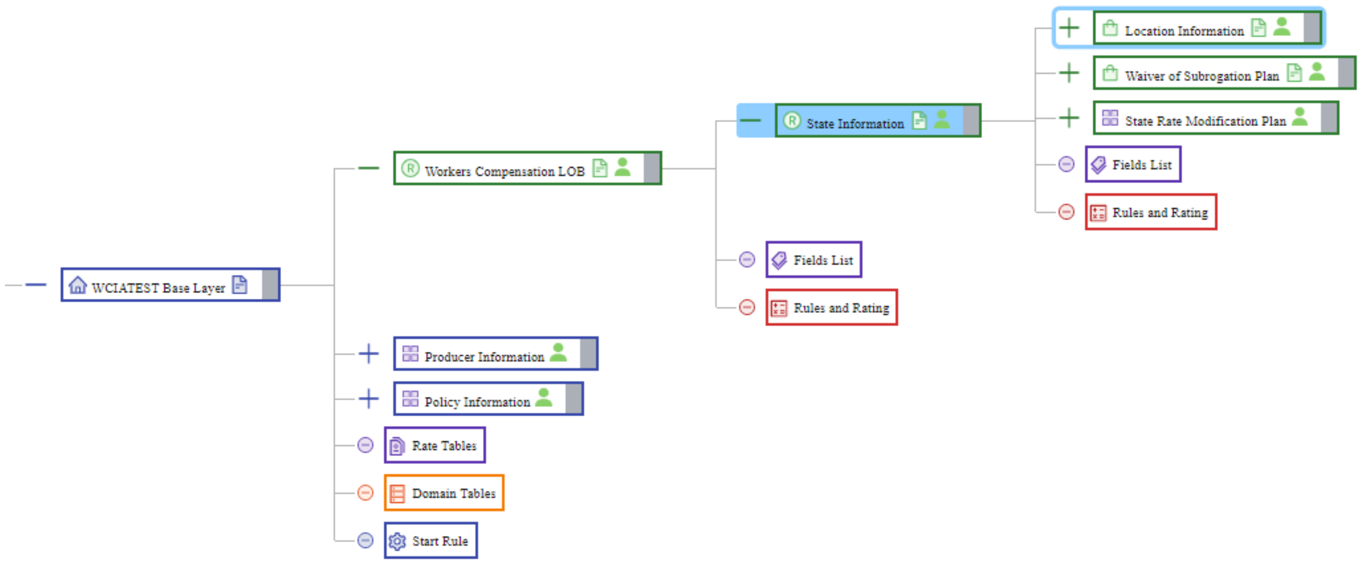


```
Rule  Name GetSubjectPremium Type decimal
DataDefGroup MasterWC_IAIBaseLayerWorkersCompensationLOBState... MetadataCodes
  Sum  ToDataDef SubjectPrem
    Value
    ToDataDef  FromDataDef ManualPrem
    Type decimal  Allow Null Return true
    From Param
    Value
    ToDataDef  FromDataDef SpecificWaiverOfSubrogationCharge
    Type decimal  Allow Null Return true
    From Param
    Value
    ToDataDef  FromDataDef DeductibleCredit
    Type decimal  Allow Null Return true
    From Param
    Value
    ToDataDef  FromDataDef IncreaseLimitCharges
    Type decimal  Allow Null Return true
    From Param
```



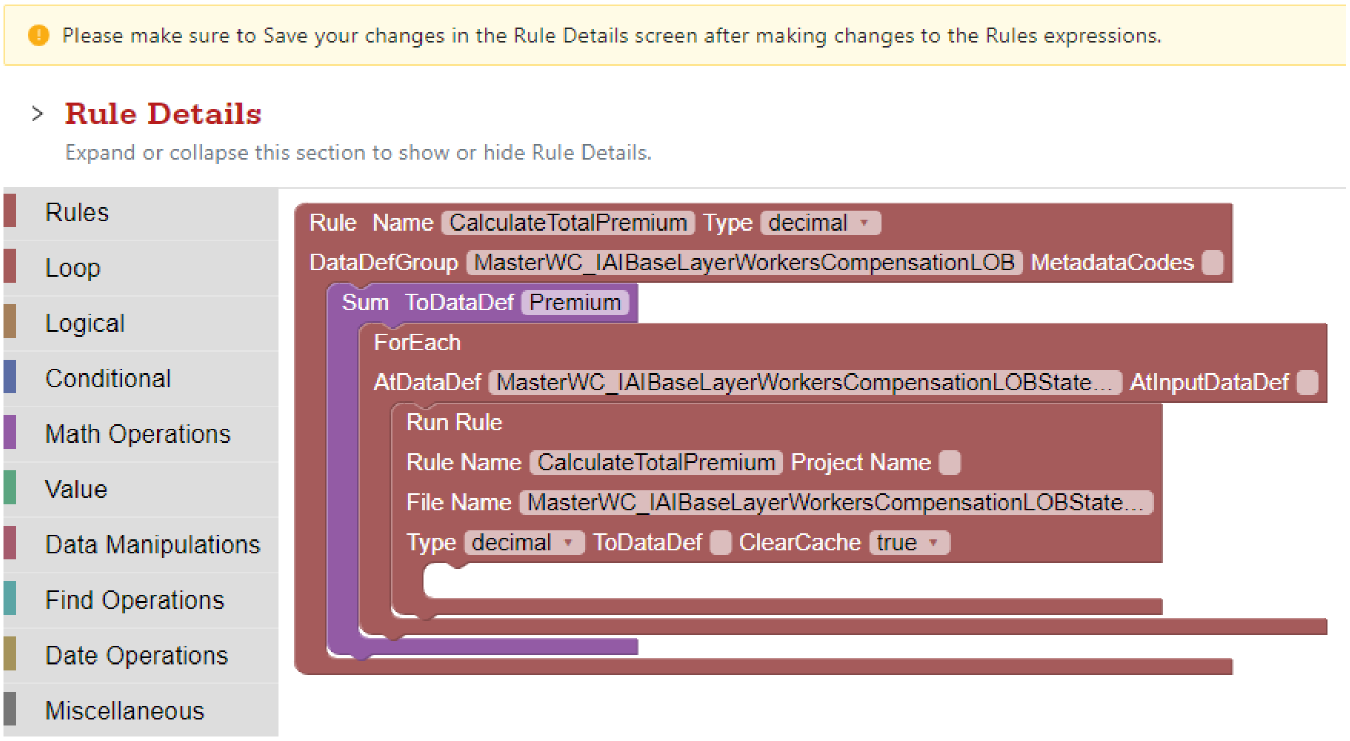
Risk

Based on business strucutre we can add a Risk node.



Here, we create below rule :

CalculateTotalPremium Blockly



Please make sure to Save your changes in the Rule Details screen after making changes to the Rules expressions.

> Rule Details

Expand or collapse this section to show or hide Rule Details.

Rule Name CalculateTotalPremium Type decimal
DataDefGroup MasterWC_IAIBaseLayerWorkersCompensationLOB MetadataCodes
Sum ToDataDef Premium
ForEach
AtDataDef MasterWC_IAIBaseLayerWorkersCompensationLOBState... AtInputDataDef
Run Rule
Rule Name CalculateTotalPremium Project Name
File Name MasterWC_IAIBaseLayerWorkersCompensationLOBState...
Type decimal ToDataDef ClearCache true

eg. MasterWCBaseLayerWorkerCompLOBRules.Rule.xml

In below script, *Name*(Name="CalculateTotalPremium") denotes name of the rule. If the rule has return type, it can be specified using *Type*(Type="decimal"). *DataDefGroup*(DataDefGroup="MasterWCBaseLayerWorkerCompLOB") specifies the current level at which we are executing the rule.

```
  <rul:Rule Name="CalculateTotalPremium" Type="decimal"
DataDefGroup="MasterWCBaseLayerWorkerCompLOB" MetadataCodes="RuleTypeCountrywide">
```

Then, we go to the next level *AtDataDef*(AtDataDef="MasterWCBaseLayerLocationTable/MasterWCBaseLayerLocation") and run rule *Rule*(Rule="CalculateTotalPremium") in file *FileName*(FileName="MasterWCBaseLayerLocationRules")

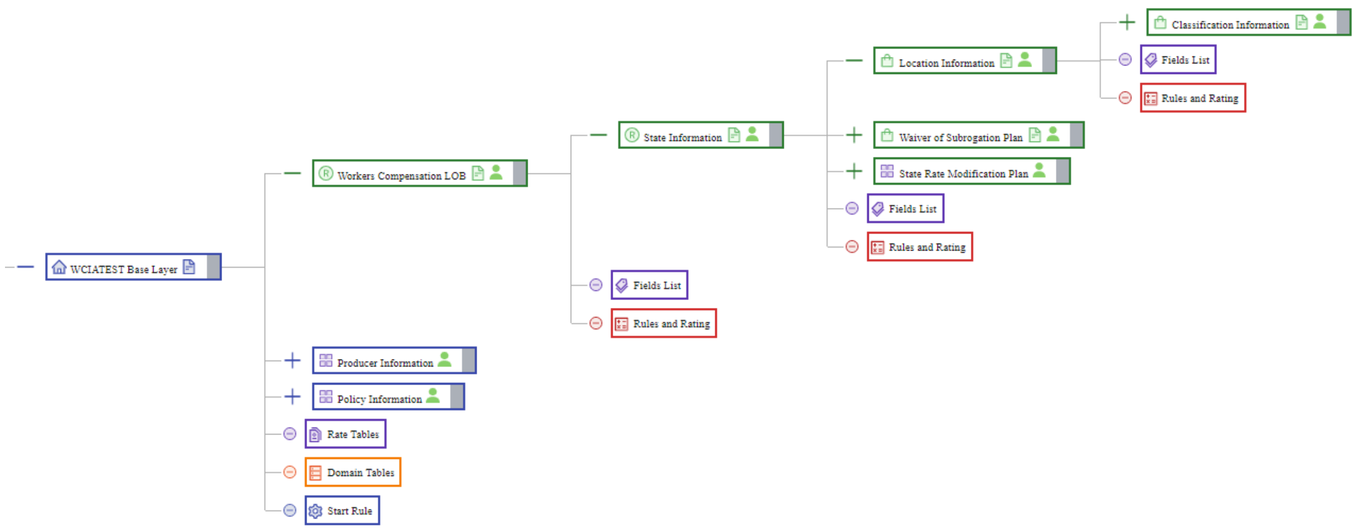
```
    <rul:Sum ToDataDef="Premium">
      <rul:ForEach
AtDataDef="MasterWCBaseLayerLocationTable/MasterWCBaseLayerLocation">
        <rul:RunRule Type="decimal" FileName="MasterWCBaseLayerLocationRules"
Rule="CalculateTotalPremium" ClearCache="true"/>
      </rul:ForEach>
    </rul:Sum>
```

The calculated result will be stored in *ToDataDef*(ToDataDef="Premium") field of *DataDefGroup*(DataDefGroup="MasterWCBaseLayerWorkerCompLOB").

```
  <rul:Rule Name="CalculateTotalPremium" Type="decimal"
DataDefGroup="MasterWCBaseLayerWorkerCompLOB" MetadataCodes="RuleTypeCountrywide">
    <rul:Sum ToDataDef="Premium">
      <rul:ForEach
AtDataDef="MasterWCBaseLayerLocationTable/MasterWCBaseLayerLocation">
        <rul:RunRule Type="decimal" FileName="MasterWCBaseLayerLocationRules"
Rule="CalculateTotalPremium" ClearCache="true"/>
      </rul:ForEach>
    </rul:Sum>
  </rul:Rule>
```

Coverage

Based on business strucutre we can add a coverage node.



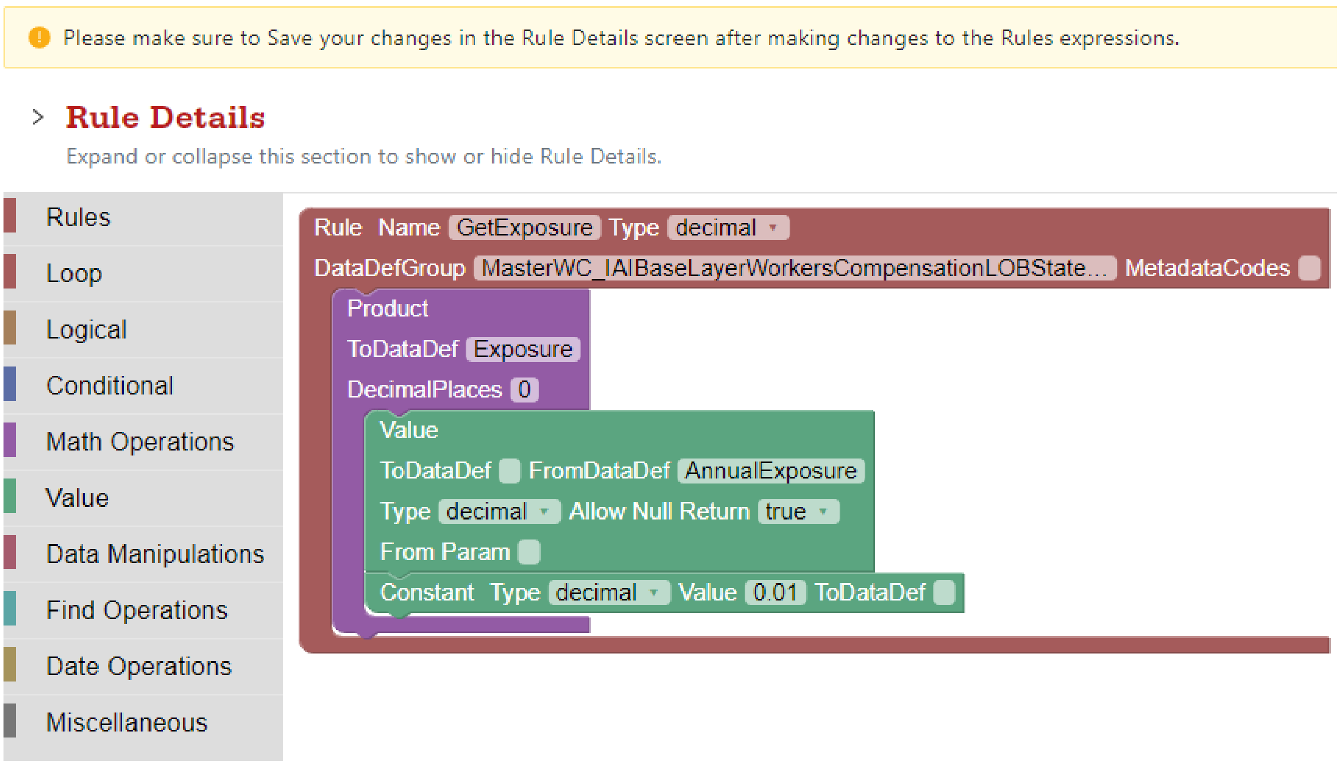
eg. MasterWCBaseLayerLocationClassificationRules.Rule.xml

In rules and rating, we write below rules,

SetRatesAndFactors

It is used to lookup rate tables and setup initial factors required for calculation.

GetExposure Blockly


Please make sure to Save your changes in the Rule Details screen after making changes to the Rules expressions.

Rule Details

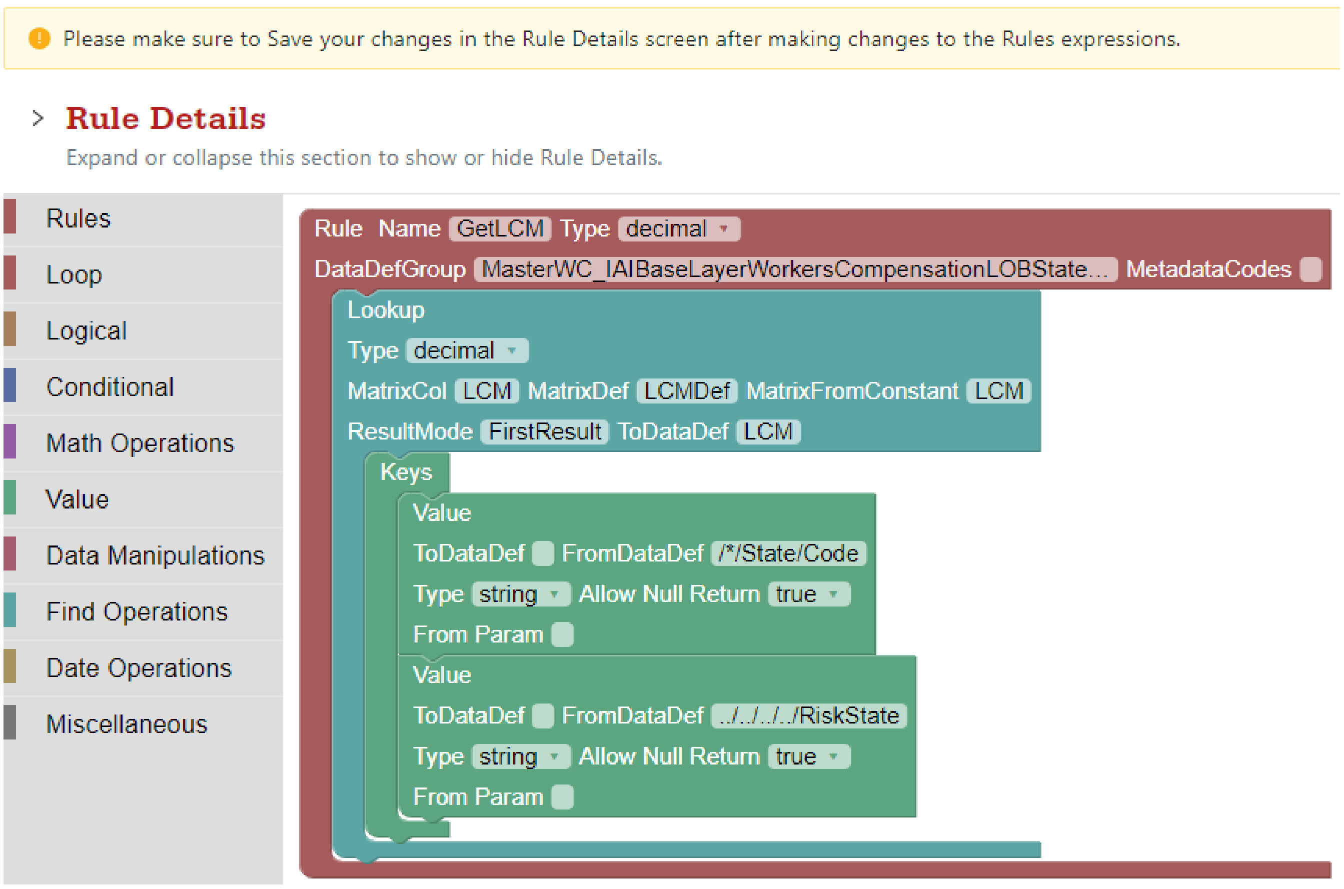
Expand or collapse this section to show or hide Rule Details.

Rule Name GetExposure Type decimal
DataDefGroup MasterWC_IAIBaseLayerWorkersCompensationLOBState... MetadataCodes
Product
ToDataDef Exposure
DecimalPlaces 0
Value
ToDataDef FromDataDef AnnualExposure
Type decimal Allow Null Return true
From Param
Constant Type decimal Value 0.01 ToDataDef

Here, we are coping value from field *FromDataDef*(FromDataDef="AnnualPayroll") to *ToDataDef*(ToDataDef="Exposure")

```
<rul:Copy Type="decimal" ToDataDef="Exposure" FromDataDef="AnnualPayroll"/>
```

GetLCM Blockly



Please make sure to Save your changes in the Rule Details screen after making changes to the Rules expressions.

> **Rule Details**
Expand or collapse this section to show or hide Rule Details.

Rules
Loop
Logical
Conditional
Math Operations
Value
Data Manipulations
Find Operations
Date Operations
Miscellaneous

Rule Name GetLCM Type decimal
DataDefGroup MasterWC_IAIBaseLayerWorkersCompensationLOBState... MetadataCodes
Lookup
Type decimal
MatrixCol LCM MatrixDef LCMDef MatrixFromConstant LCM
ResultMode FirstResult ToDataDef LCM
Keys
Value
ToDataDef FromDataDef /*/State/Code
Type string Allow Null Return true
From Param
Value
ToDataDef FromDataDef ../../../../RiskState
Type string Allow Null Return true
From Param

In below script, rate table name is *MatrixFromConstant*(MatrixFromConstant="LossCostMultiplier"). *MatrixDef* is rate table name with suffix *Def*(MatrixDef="LossCostMultiplierDef"). It is used to establish key-value relationship within rate table. The result will be stored in field *ToDataDef*(ToDataDef="LCM") and return type will be *Type*(Type="decimal"). *MatrixCol*(MatrixCol="LossCostMultiplier") specifies the value column returned to corresponding key passed.

```
      <rul:Lookup Type="decimal" ToDataDef="LCM" MatrixCol="LossCostMultiplier"
MatrixDef="LossCostMultiplierDef" MatrixFromConstant="LossCostMultiplier"
ResultMode="FirstResult">
        ...
        ...
```

rate_table_LCM

Keys are used to lookup the rate table. It needs to be in the same sequence, as columns in the rate table. Here, the first key "/*/State/Code"/ is CW, set from below line in Overall Rating.Rule.xml file.

```
    <rul:Constant Type="string" ToDataDef="State/Code">CW</rul:Constant>
```

The second key is retrieved from the rate table.

```
<rul:Keys>
          <rul:Value Type="string" FromDataDef="/*/State/Code"/>
          <rul:Value Type="string" FromDataDef="ClassCode"/>
        </rul:Keys>
```

eg. CalculateAnnualPremium

This rule file calculates premium using formula : (LC *USLH*LCM* EXP).

coverage_annual_prem

Here, the multiplication operation result will be stored in field *ToDataDef*(ToDataDef="ClassAnnualPremium").

```
  <rul:Rule Name="CalculateAnnualPremium" Type="none"
DataDefGroup="MasterWCBaseLayerLocationClassification"
MetadataCodes="RuleTypeCountrywide">
    <rul:Product ToDataDef="ClassAnnualPremium" DecimalPlaces="0">
      <rul:FirstValue Type="decimal" FromConstant="0.0" FromDataDef="Exposure"
Order="DataDefInputParamConstant"/>
      <rul:FirstValue Type="decimal" FromConstant="0.0" FromDataDef="LCM"
Order="DataDefInputParamConstant"/>
      ...
      ...
```

FromDataDef(FromDataDef="Exposure") field is at the same level *DataDefGroup*(DataDefGroup="MasterWCBaseLayerLocationClassification"). If field is defined beyond this level, we have to give relative xpath of field location against the current level *DataDefGroup*(DataDefGroup="MasterWCBaseLayerLocationClassification").

eg. CalculateTotalPremium

coverage_ctp

In below script, *Name*(Name="CalculateTotalPremium") denotes name of the rule. If the rule has return type, it can be specified using *Type*(Type="decimal"). *DataDefGroup*(DataDefGroup="MasterWCBaseLayerLocationClassification") specifies the current level at which we are executing the rule.

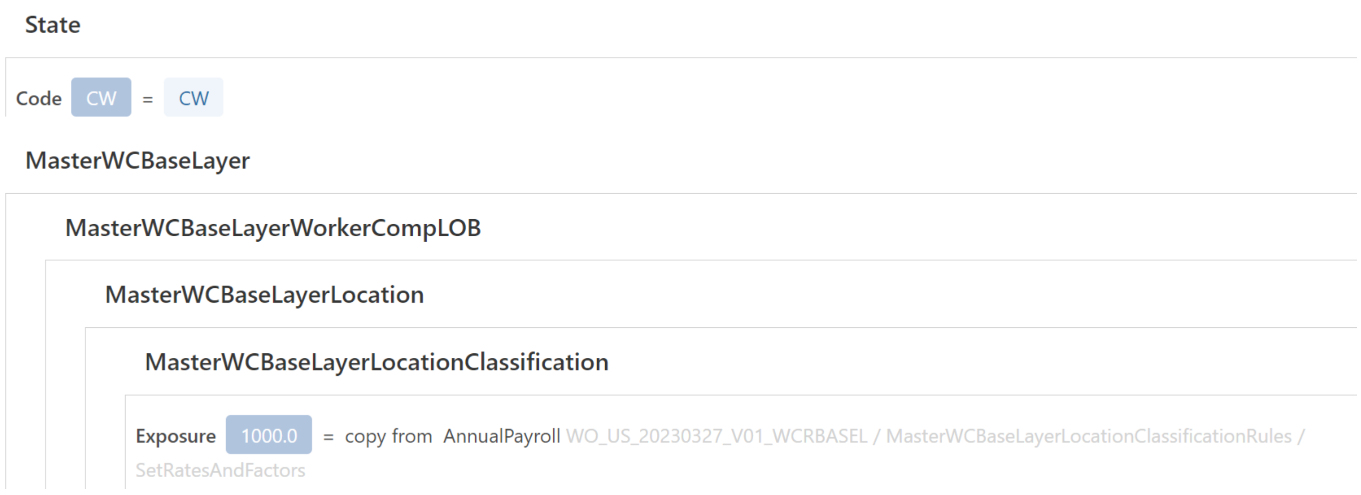
```
  <rul:Rule Name="CalculateTotalPremium" Type="decimal"
DataDefGroup="MasterWCBaseLayerLocationClassification"
MetadataCodes="RuleTypeCountrywide">
```

Then, we run rule *Rule*(Rule="CalculateTotalPremium") in same file *FileName*(FileName="MasterWCBaseLayerLocationClassification").The calculated result will be summed up and stored in upper levels. In this rule, we are make call to more rules following the same sequence.

```
<rul:Rule Name="CalculateTotalPremium" Type="decimal"
DataDefGroup="MasterWCBaseLayerLocationClassification"
MetadataCodes="RuleTypeCountrywide">
    <rul:Sequence>
      <rul:RunRule Type="none"
FileName="MasterWCBaseLayerLocationClassificationRules" Rule="SetRatesAndFactors"
ClearCache="true"/>
      <rul:RunRule Type="none"
FileName="MasterWCBaseLayerLocationClassificationRules"
Rule="CalculateAnnualPremium" ClearCache="true"/>
      ...
    <rul:Sequence/>
```

The Rating Worksheet

In rating worksheet (**WO_US_20230327_V01_WCRBASEL / MasterWCBaseLayerLocationClassificationRules / SetRatesAndFactors**), *WO_US_20230327_V01_WCRBASEL* gives project name, *MasterWCBaseLayerLocationClassificationRules* is rule filename, *SetRatesAndFactors* denotes rule name.


State

Code CW = CW

MasterWCBaseLayer

MasterWCBaseLayerWorkerCompLOB

MasterWCBaseLayerLocation

MasterWCBaseLayerLocationClassification

Exposure 1000.0 = copy from AnnualPayroll WO_US_20230327_V01_WCRBASEL / MasterWCBaseLayerLocationClassificationRules / SetRatesAndFactors

Explanation on Rules with Attributes

Rules Section

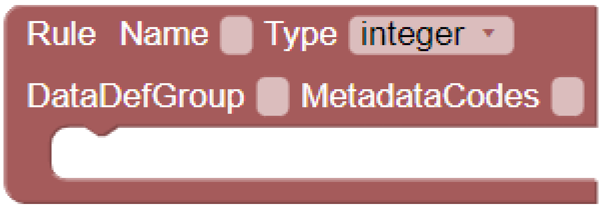
- **rul:Rule** : It is used to define a rule.

Name : It specifies name of the rule.

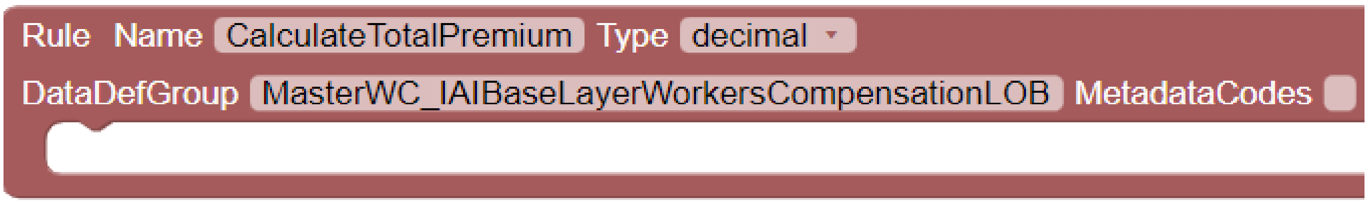
Type : It specifies return type of a rule.

DataDefGroup : It defines the current element when we enter the rule.

MetadataCodes : It gives information about metadata.



```
<rul:Rule Name="InitializeRuleSet"
DataDefGroup="GeneralLiabilityAbuseMolestationExcl"
MetadataCodes="RuleTypeSystem">
```


Rule Name CalculateTotalPremium Type decimal
DataDefGroup MasterWC_IAIBaseLayerWorkersCompensationLOB MetadataCodes

- **rul:Run Rule** : It is used to make call to a rule.

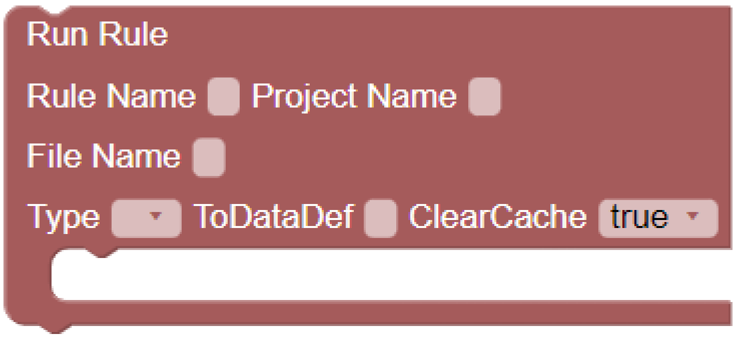
Type : It specifies return type of a rule.

FileName : It denotes the name of rule.

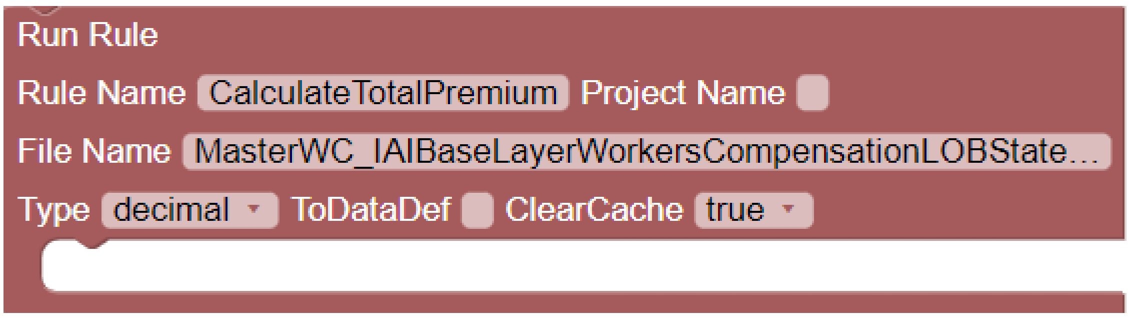
Rule : It denotes name of function inside rule.

ToDataDef : The xpath mentioned points to the location where the result will be stored.

ClearCache : It resets cache.

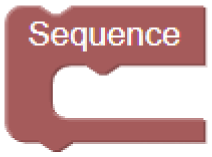


```
<rul:RunRule Type="none" FileName="GeneralLiabilityAbuseMolestationExclRules"
Rule="SetPremium" ClearCache="true" />
```

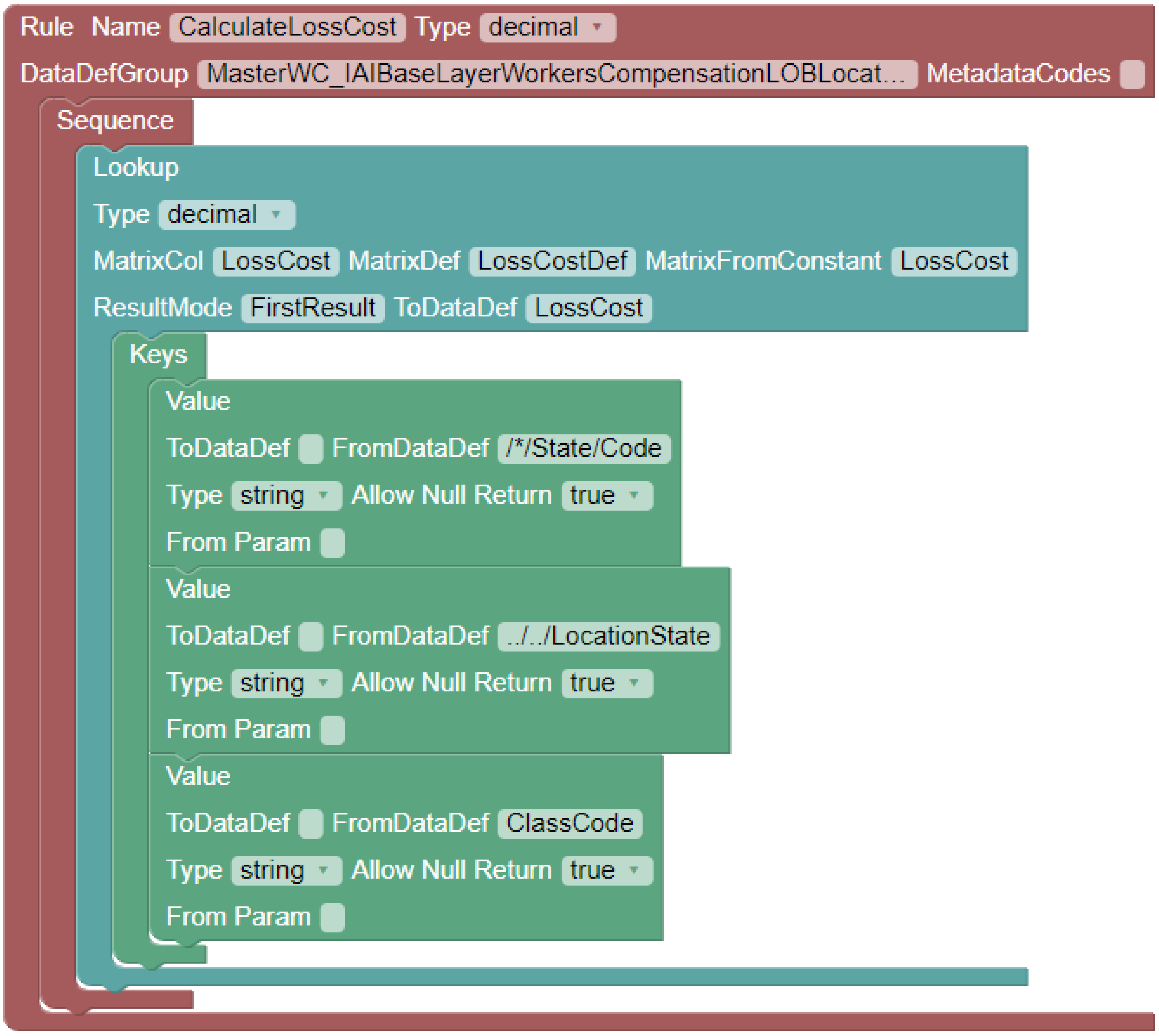

Run Rule
Rule Name CalculateTotalPremium Project Name
File Name MasterWC_IAIBaseLayerWorkersCompensationLOBState...
Type decimal ToDataDef ClearCache true

Loop Section

- **rul:Sequence** : It executes blocks within, in the given sequence.



```
<rul:Sequence>
            <rul:RunRule Type="none"
FileName="GeneralLiabilityAbuseOrMolestationExclusionSpecifiedProfessionalServices
Rules" Rule="SetPremium" ClearCache="true" />
            <rul:RunRule Type="none"
FileName="GeneralLiabilityAbuseOrMolestationExclusionSpecifiedProfessionalServices
Rules" Rule="SetPremiumIndicator" ClearCache="true" />
</rul:Sequence>
```

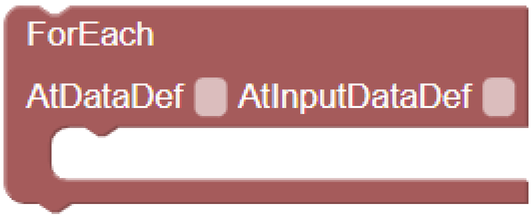


```
Rule Name CalculateLossCost Type decimal
DataDefGroup MasterWC_IAIBaseLayerWorkersCompensationLOBLocat... MetadataCodes
  Sequence
    Lookup
    Type decimal
    MatrixCol LossCost MatrixDef LossCostDef MatrixFromConstant LossCost
    ResultMode FirstResult ToDataDef LossCost
      Keys
        Value
        ToDataDef  FromDataDef /*/State/Code
        Type string Allow Null Return true
        From Param
        Value
        ToDataDef  FromDataDef ../../LocationState
        Type string Allow Null Return true
        From Param
        Value
        ToDataDef  FromDataDef ClassCode
        Type string Allow Null Return true
        From Param
```

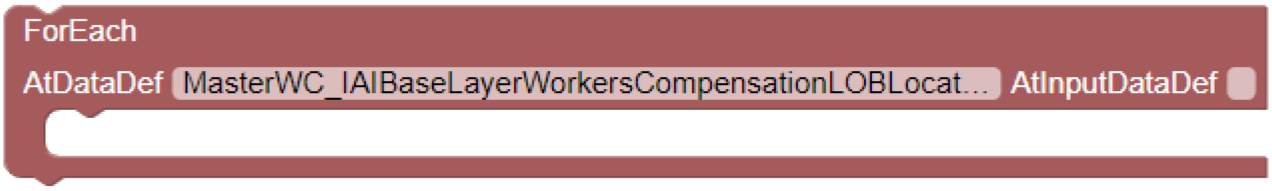
- **rul:ForEach** : It works as for-loop, iterates and executes rule against each layer.

AtDataDef : The xpath specifies the xml table (points to current XML element), which is to be iterated.

AtInputDataDef : It pushes the mentioned xpath value into context, without changing the "current" pointer. This can be used, when we want the xpath to be executed to a specific element without changing the current XML element, since the other xpaths could execute against the current one.



```
<rul:ForEach
AtDataDef="GeneralLiabilityAbuseOrMolestationExclusionSpecifiedProfessionalService
sDetailTable/GeneralLiabilityAbuseOrMolestationExclusionSpecifiedProfessionalServi
cesDetail">
  <rul:RunRule Type="decimal"
FileName="GeneralLiabilityAbuseOrMolestationExclusionSpecifiedProfessionalServices
DetailRules" Rule="CalculateTotalPremium" ClearCache="true" />
</rul:ForEach>
```

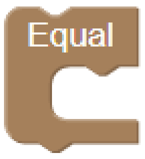


- **rul:Break** : It stops execution and returns None.

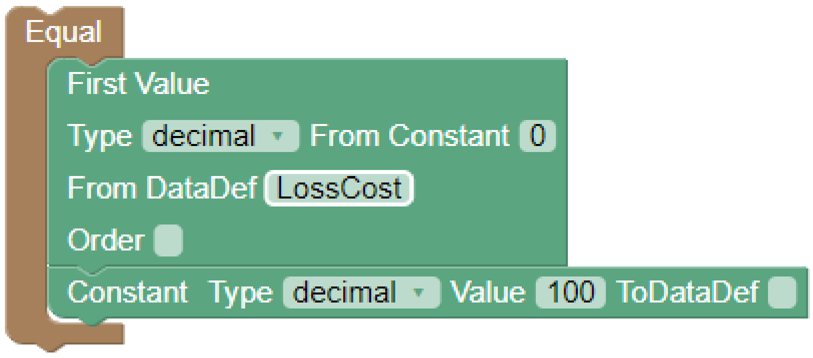
```
<rul:Sum>
  <rul:Constant Type="integer">1</rul:Constant>
  <rul:Break />
</rul:Sum>
```

Logical Section

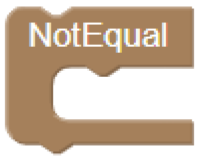
- **rul:Equal** : Compares if values are exactly same.



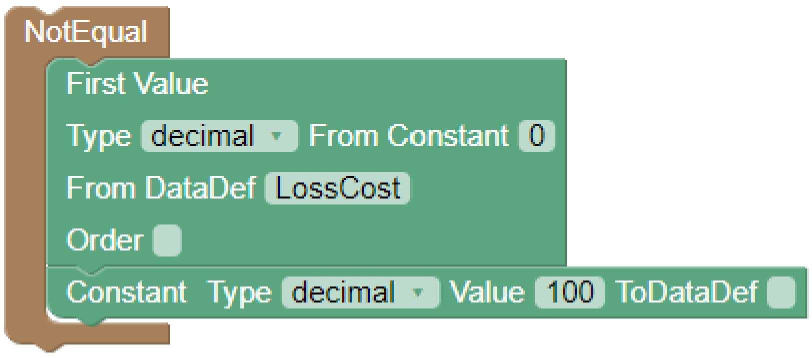
```
<rul:Equal>
  <rul:FirstValue Type="integer" FromConstant="0" FromDataDef="PremiumIndicator"
Order="DataDefInputParamConstant" />
  <rul:Constant Type="integer">1</rul:Constant>
</rul:Equal>
```


Equal
First Value
Type decimal From Constant 0
From DataDef LossCost
Order
Constant Type decimal Value 100 ToDataDef

- **rul:NotEqual** : Compares if values are different.



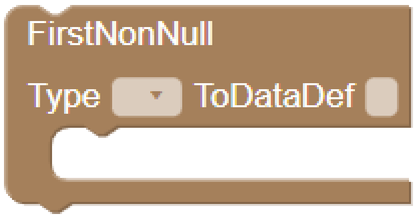
```
<rul:NotEqual>
  <rul:FirstValue Type="decimal" FromConstant="0" FromDataDef="Premium"
Order="DataDefInputParamConstant" />
  <rul:Constant Type="decimal">0</rul:Constant>
</rul:NotEqual>
```


NotEqual
First Value
Type decimal From Constant 0
From DataDef LossCost
Order
Constant Type decimal Value 100 ToDataDef

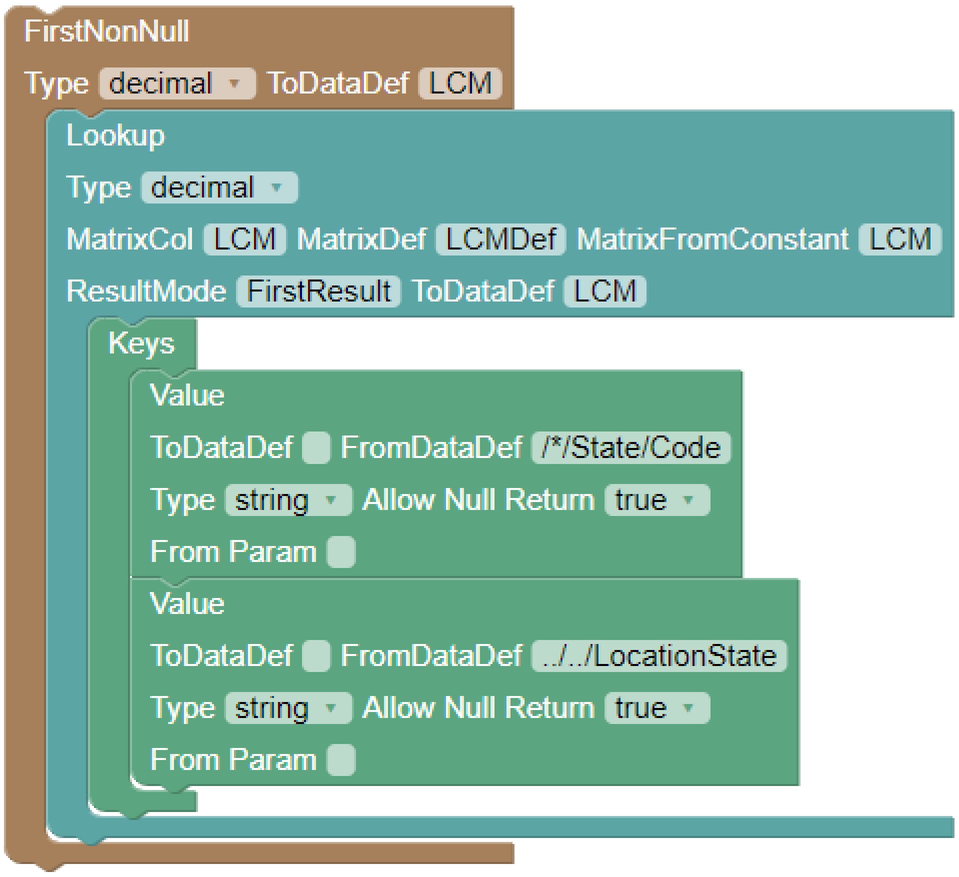
- **rul:FirstNonNull** : Checks if first value is present.

Type : It specifies type of first value.

ToDataDef : It saves first value to location given in xpath.

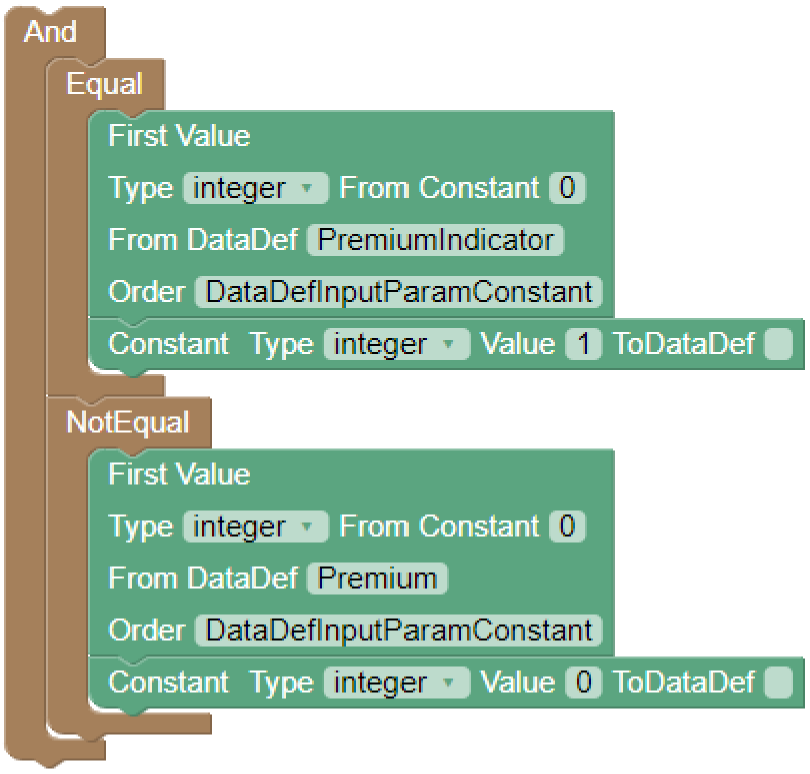


```
<rul:FirstNonNull Type="string">
  <rul:Lookup Type="string" MatrixCol="Code" MatrixDef="NoDedStatCodeDef"
MatrixFromConstant="NoDedStatCode" ResultMode="SingleResult">
    <rul:Keys>
      <rul:Constant Type="string">CW</rul:Constant>
      <rul:Constant Type="string">Y</rul:Constant>
    </rul:Keys>
  </rul:Lookup>
</rul:FirstNonNull>
```


FirstNonNull
Type decimal ToDataDef LCM
Lookup
Type decimal
MatrixCol LCM MatrixDef LCMDef MatrixFromConstant LCM
ResultMode FirstResult ToDataDef LCM
Keys
Value
ToDataDef FromDataDef /*/State/Code
Type string Allow Null Return true
From Param
Value
ToDataDef FromDataDef ../../LocationState
Type string Allow Null Return true
From Param

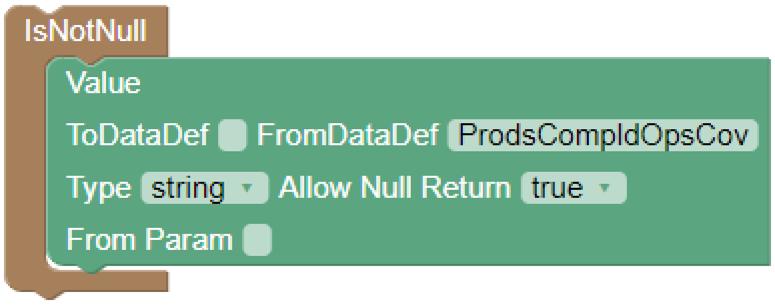
- **rul:And** : It is used to check mutiple conditions simultaneously.

```
<rul:And>
  <rul:Equal>
    <rul:FirstValue Type="integer" FromConstant="0" FromDataDef="PremiumIndicator"
Order="DataDefInputParamConstant" />
    <rul:Constant Type="integer">1</rul:Constant>
  </rul:Equal>
  <rul:NotEqual>
    <rul:FirstValue Type="decimal" FromConstant="0" FromDataDef="Premium"
Order="DataDefInputParamConstant" />
    <rul:Constant Type="decimal">0</rul:Constant>
  </rul:NotEqual>
</rul:And>
```

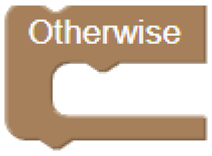

And
Equal
First Value
Type integer From Constant 0
From DataDef PremiumIndicator
Order DataDefInputParamConstant
Constant Type integer Value 1 ToDataDef
NotEqual
First Value
Type integer From Constant 0
From DataDef Premium
Order DataDefInputParamConstant
Constant Type integer Value 0 ToDataDef

- **rul:IsNotNull** : Checks if value is present.

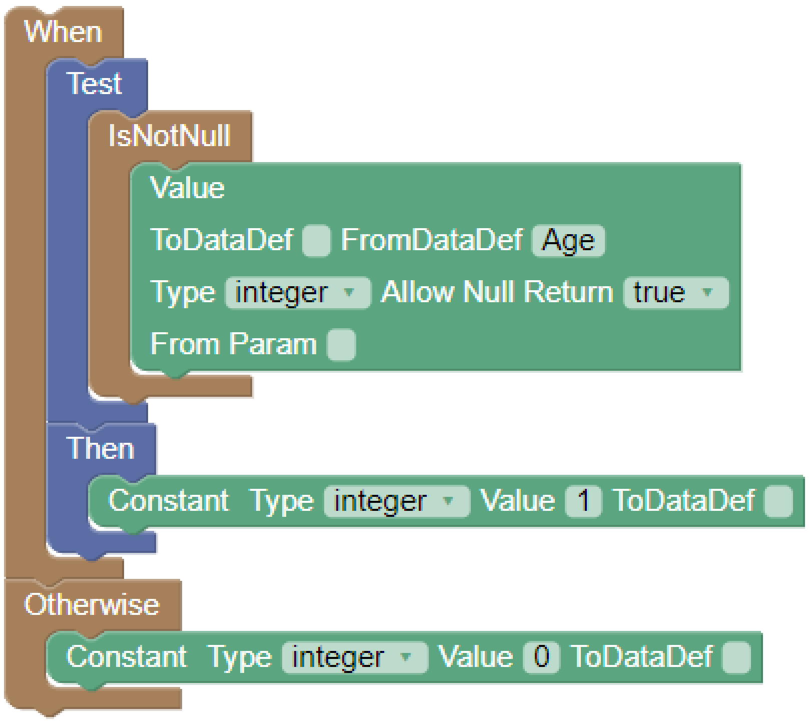
```
<rul:IsNotNull>
  <rul:Value Type="string" FromDataDef="ProdsCompldOpsCov" />
</rul:IsNotNull>
```



- **rul:Otherwise** : When condition given in test evalutes to False, we execute statements given in Otherwise block.

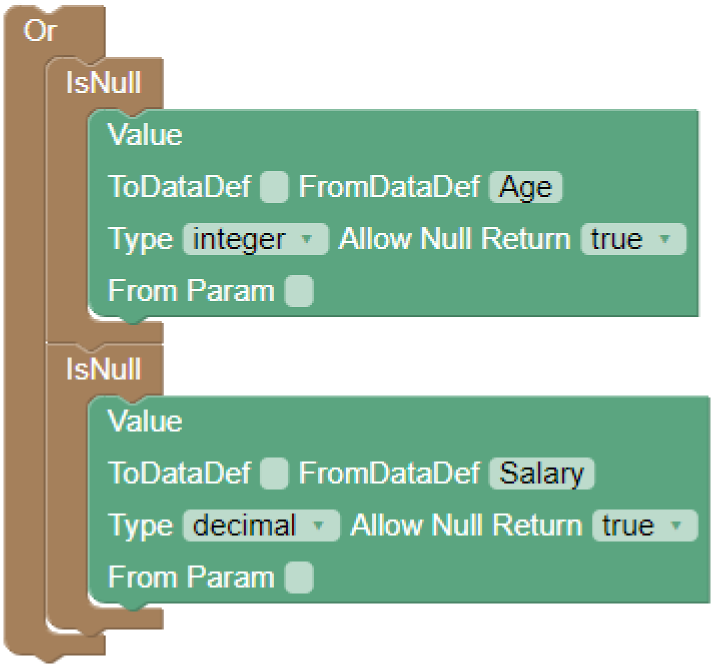


```
<rul:When>
  <rul:Test>
    <rul:And>
      <rul:NotExist
AtInputDataDef="../../../../../../GeneralLiabilitySupplementalExtendedReportingPer
iodEndtTable/GeneralLiabilitySupplementalExtendedReportingPeriodEndt" />
      <rul:NotExist
AtInputDataDef="../../../../../../GeneralLiabilitySupplementalExtendedReportingPer
iodEndtSpecificAccsProdsWorkOrLocationsTable/GeneralLiabilitySupplementalExtendedR
eportingPeriodEndtSpecificAccsProdsWorkOrLocations" />
    </rul:And>
  </rul:Test>
  <rul:Then>
    <rul:Constant Type="string">Basic</rul:Constant>
  </rul:Then>
</rul:When>
<rul:Otherwise>
  <rul:Constant Type="string">Supplemental Extended Reporting
Period</rul:Constant>
</rul:Otherwise>
```


When
Test
IsNotNull
Value
ToDataDef FromDataDef Age
Type integer Allow Null Return true
From Param
Then
Constant Type integer Value 1 ToDataDef
Otherwise
Constant Type integer Value 0 ToDataDef

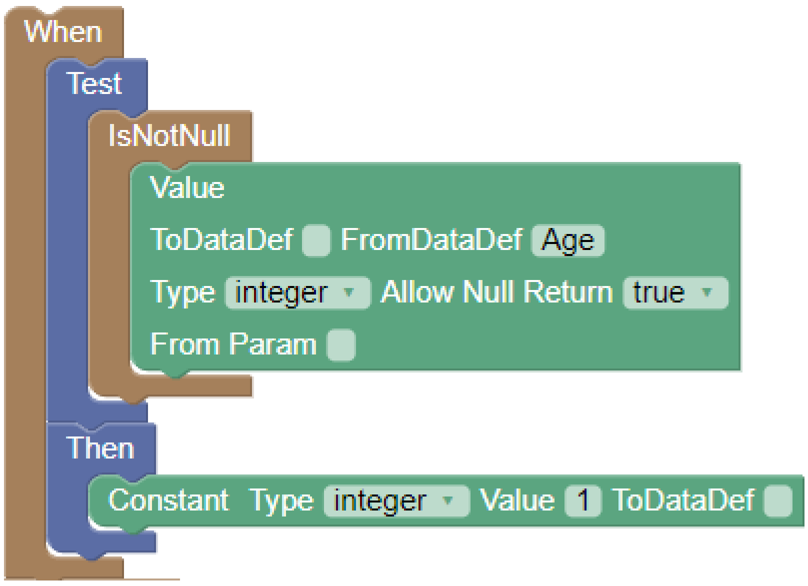
- **rul:Or** : Checks whether one condition, from multiple conditions is True.

```
<rul:Test>
  <rul:Or>
    <rul:NotExist
AtInputDataDef="../../../../../../GeneralLiabilitySupplementalExtendedReportingPer
iodEndtTable/GeneralLiabilitySupplementalExtendedReportingPeriodEndt" />
    <rul:NotExist
AtInputDataDef="../../../../../../GeneralLiabilitySupplementalExtendedReportingPer
iodEndtSpecificAccsProdsWorkOrLocationsTable/GeneralLiabilitySupplementalExtendedR
eportingPeriodEndtSpecificAccsProdsWorkOrLocations" />
  </rul:Or>
</rul:Test>
```



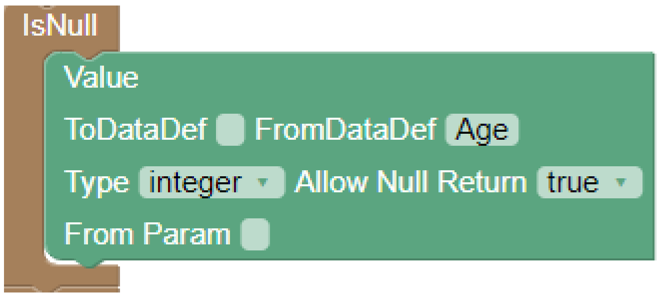
- **rul:When** : Checks if condition given in Test block is True.

```
<rul:When>
  <rul:Test>
    <rul:Equal>
      <rul:Length>
        <rul:Value Type="string" FromParam="strExposureStatCodeliquor" />
      </rul:Length>
      <rul:Constant Type="integer">7</rul:Constant>
    </rul:Equal>
  </rul:Test>
  <rul:Then>
    <rul:Sequence>
      <rul:Value Type="string" ToDataDef="ExposureStatCode"
FromParam="strExposureStatCodeliquor" />
    </rul:Sequence>
  </rul:Then>
</rul:When>
```



- **rul:IsNull** : Checks if value is not present.

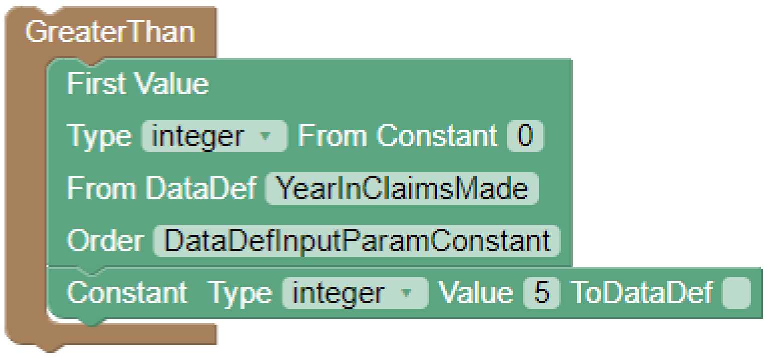
```
<rul:IsNull>
  <rul:Value Type="decimal"
FromDataDef="../ProductWithdrawalDeductibleFactorOverride" />
</rul:IsNull>
```



- **rul:GreaterThan** : Checks if first element is greater than second, in given block.

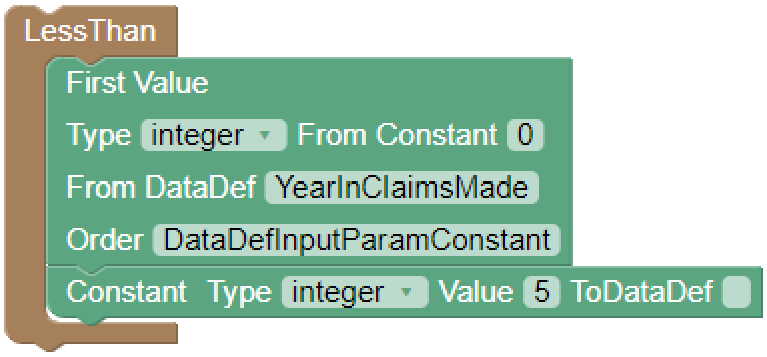
GreaterThan

```
<rul:GreaterThan>
  <rul:FirstValue Type="integer" FromConstant="0" FromDataDef="YearInClaimsMade"
Order="DataDefInputParamConstant" />
  <rul:Constant Type="integer">5</rul:Constant>
</rul:GreaterThan>
```

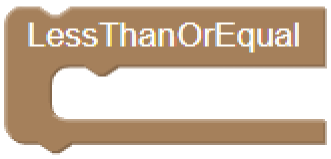

GreaterThan
First Value
Type integer From Constant 0
From DataDef YearInClaimsMade
Order DataDefInputParamConstant
Constant Type integer Value 5 ToDataDef

- **rul:LessThan** : Checks if first element is less than second, in given block.

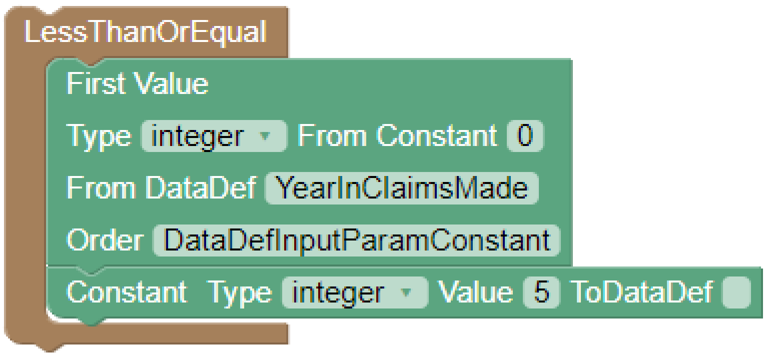
```
<rul:LessThan>
  <rul:Length>
    <rul:Value Type="string" FromParam="prodWithLCM" />
  </rul:Length>
  <rul:Constant Type="integer">3</rul:Constant>
</rul:LessThan>
```


LessThan
First Value
Type integer From Constant 0
From DataDef YearInClaimsMade
Order DataDefInputParamConstant
Constant Type integer Value 5 ToDataDef

- **rul:LessThanOrEqual** : Checks if first element is lesser than or equal to second element, in given block.



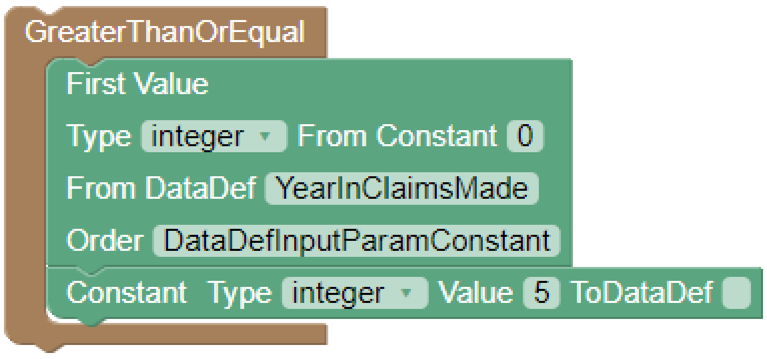
```
<rul:LessThanOrEqual>
  <rul:FirstValue Type="long" FromConstant="0"
FromDataDef="../OwnersContractorsExposure" Order="DataDefInputParamConstant" />
  <rul:Constant Type="integer">1000000</rul:Constant>
</rul:LessThanOrEqual>
```


LessThanOrEqual
First Value
Type integer From Constant 0
From DataDef YearInClaimsMade
Order DataDefInputParamConstant
Constant Type integer Value 5 ToDataDef

- **rul:GreaterThanOrEqual** : Checks if first element is greater than or equal to second element, in given block.

GreaterThanOrEqual

```
<rul:GreaterThanOrEqual>
  <rul:FirstValue Type="decimal" FromConstant="0.0"
FromDataDef="ERPCredibilityFactor" Order="DataDefInputParamConstant" />
  <rul:Constant Type="decimal">0.07</rul:Constant>
</rul:GreaterThanOrEqual>
```

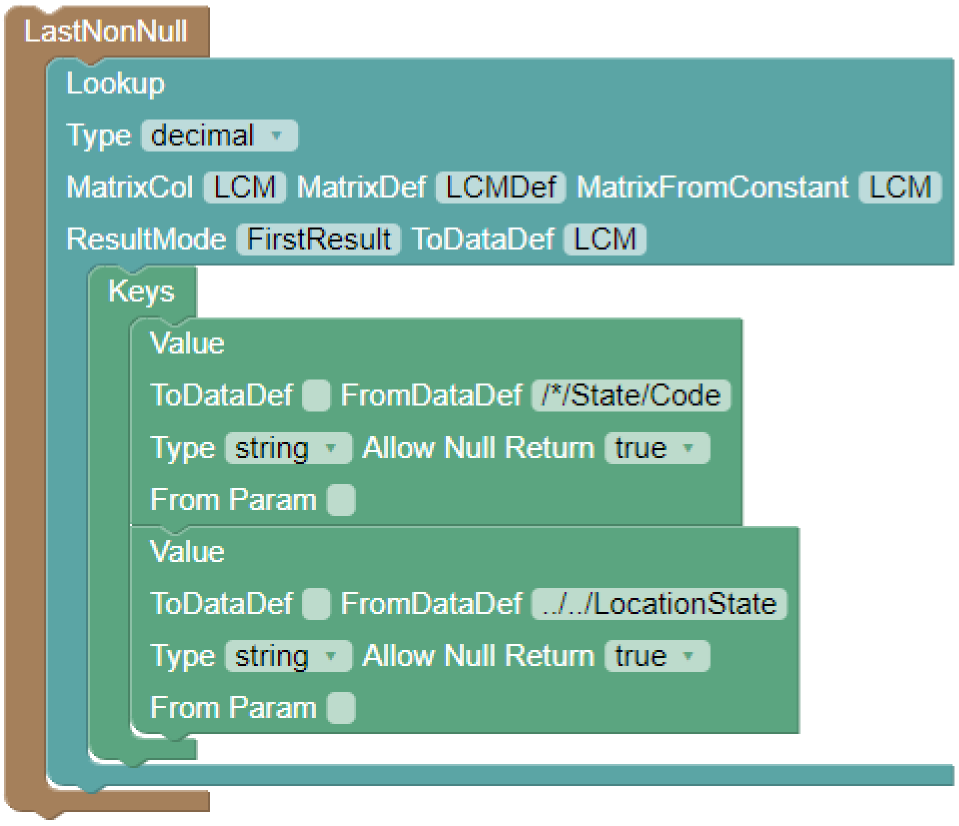

GreaterThanOrEqual
First Value
Type integer From Constant 0
From DataDef YearInClaimsMade
Order DataDefInputParamConstant
Constant Type integer Value 5 ToDataDef

- **rul:LastNonNull** : Checks if last element is present.

LastNonNull

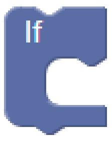
```
<rul:LastNonNull Type="string">
  <rul:Lookup Type="string" MatrixCol="Code" MatrixDef="NoDedStatCodeDef"
MatrixFromConstant="NoDedStatCode" ResultMode="SingleResult">
    <rul:Keys>
      <rul:Constant Type="string">CW</rul:Constant>
```

```
        <rul:Constant Type="string">Y</rul:Constant>
      </rul:Keys>
    </rul:Lookup>
  </rul:LastNonNull>
```

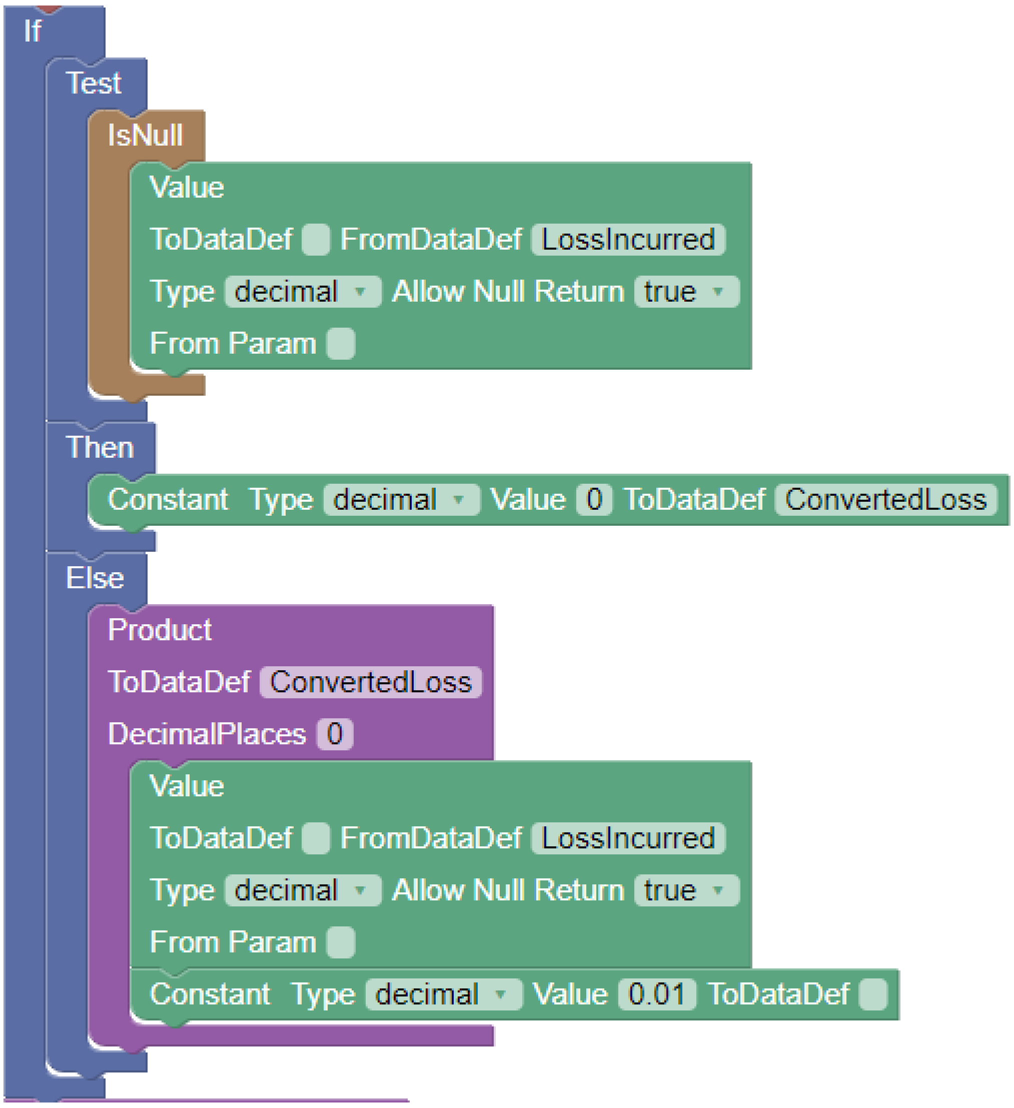

LastNonNull
Lookup
Type decimal
MatrixCol LCM MatrixDef LCMDef MatrixFromConstant LCM
ResultMode FirstResult ToDataDef LCM
Keys
Value
ToDataDef FromDataDef /*/State/Code
Type string Allow Null Return true
From Param
Value
ToDataDef FromDataDef ../../LocationState
Type string Allow Null Return true
From Param

Conditional Section

- **rul:If** : It is used to check a condition.

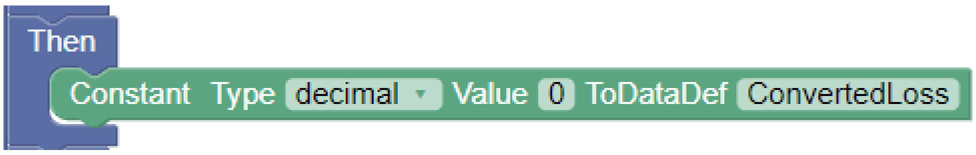


```
<rul:If>
  <rul:Test>
    <rul:Equal>
      <rul:Length>
        <rul:Value Type="string" FromParam="strExposureStatCodeliquor" />
      </rul:Length>
      <rul:Constant Type="integer">7</rul:Constant>
    </rul:Equal>
  </rul:Test>
  <rul:Then>
    <rul:Sequence>
      <rul:Value Type="string" ToDataDef="ExposureStatCode"
FromParam="strExposureStatCodeliquor" />
    </rul:Sequence>
  </rul:Then>
  <rul:Else>
    <rul:Sequence>
      <rul:Value Type="string" ToDataDef="ExposureStatCode"
FromParam="strExposureCWCodeliquor" />
    </rul:Sequence>
  </rul:Else>
</rul:If>
```



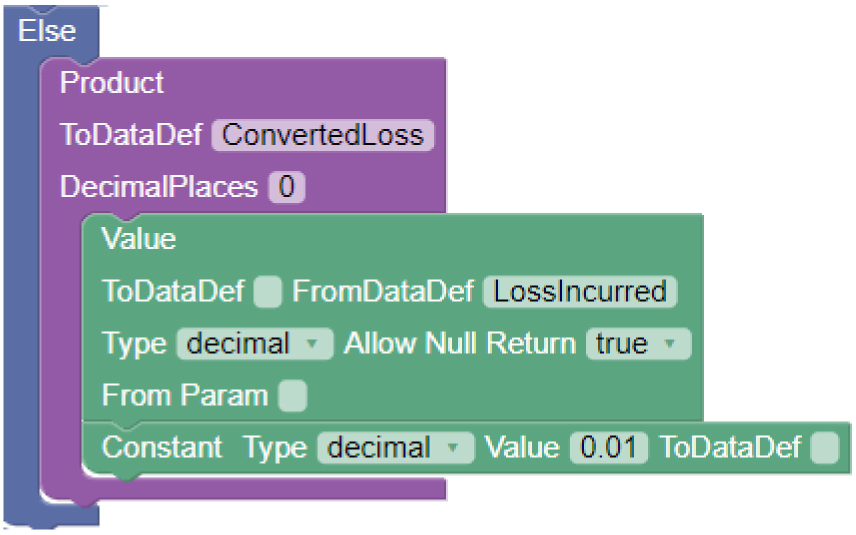
- **rul:Then** : It specifis the statements to execute, if condition is true.

```
<rul:Then>
  <rul:Sequence>
    <rul:Value Type="string" ToDataDef="ExposureStatCode"
FromParam="strExposureStatCodeliquor" />
  </rul:Sequence>
</rul:Then>
```



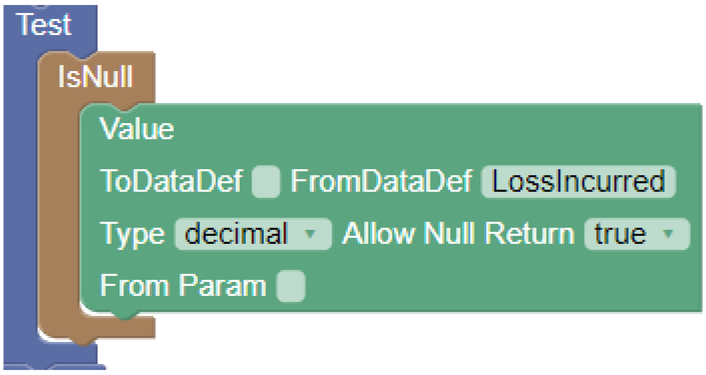
- **rul:Else** : It specifis the statements to execute, if condition is false.

```
<rul:Else>
  <rul:Sequence>
    <rul:Value Type="string" ToDataDef="ExposureStatCode"
FromParam="strExposureCWCodeliquor" />
  </rul:Sequence>
</rul:Else>
```



- **rul:Test** : It is used to check if a condition is true or false.

```
<rul:Test>
  <rul:Equal>
    <rul:Length>
      <rul:Value Type="string" FromParam="strExposureStatCodeliquor" />
    </rul:Length>
    <rul:Constant Type="integer">7</rul:Constant>
  </rul:Equal>
</rul:Test>
```

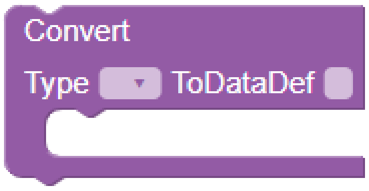


Math Operations Section

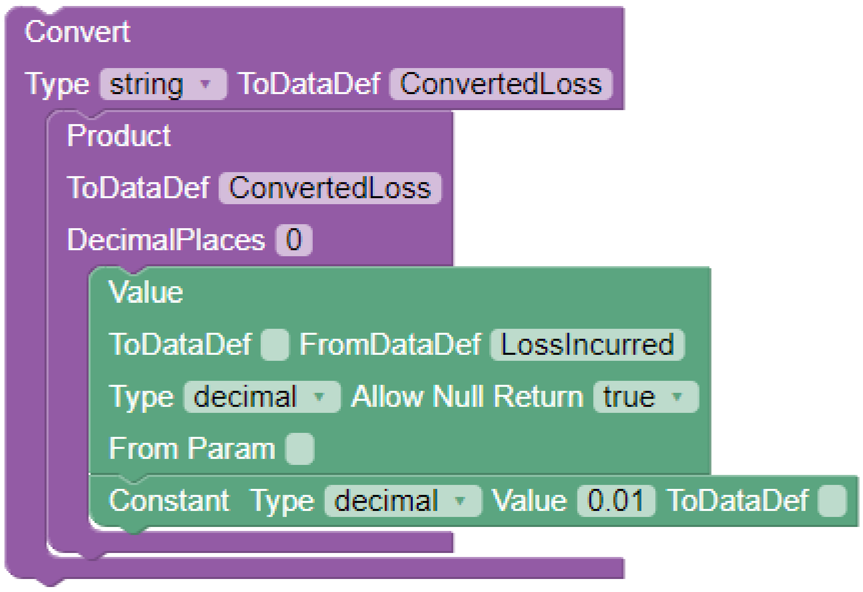
- **rul:Convert** : It is used to change type of field.

Type : It specifies the type, to be converted.

ToDataDef : The xpath mentioned points to the location where the converted field will be stored.

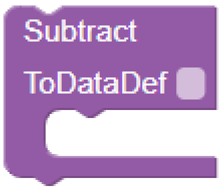


```
<rul:Convert Type="decimal">
  <rul:Divide DecimalPlaces="0">
    <rul:Value Type="decimal" FromParam="productWithdrawalExposure" />
    <rul:Value Type="integer" FromParam="productWithdrawalReportingBasis" />
  </rul:Divide>
</rul:Convert>
```

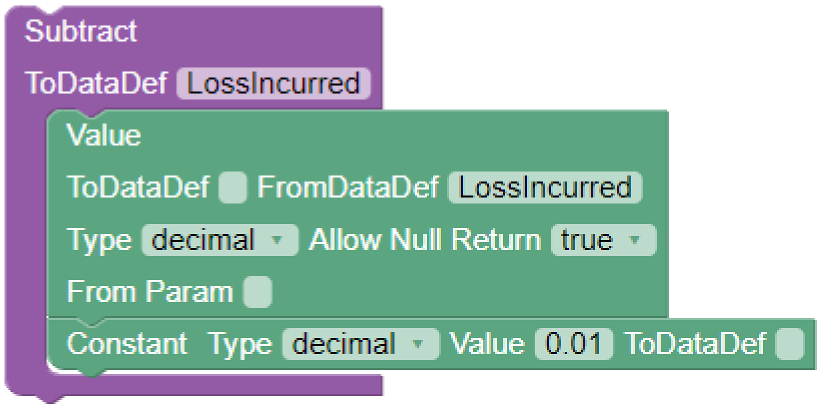


- **rul:Subtract** : It calculates difference between given fields.

ToDataDef : The mentioned xpath points to the location, where the calculated difference will be stored.

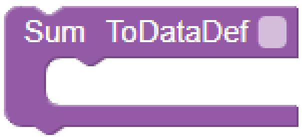


```
<rul:Subtract>
  <rul:Value Type="decimal" FromDataDef="CSLILF" />
  <rul:Value Type="decimal" FromDataDef="DeductibleFactor" />
</rul:Subtract>
```

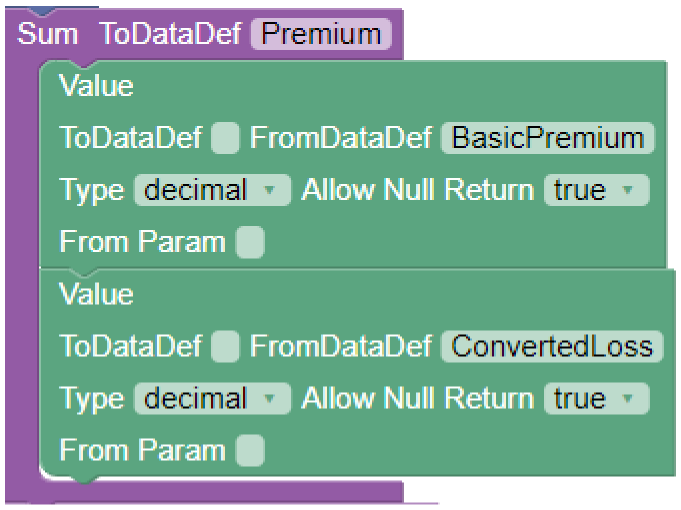

Subtract
ToDataDef LossIncurred
Value
ToDataDef FromDataDef LossIncurred
Type decimal Allow Null Return true
From Param
Constant Type decimal Value 0.01 ToDataDef

- **rul:Sum** : It returns addition of given fields.

ToDataDef : The mentioned xpath points to the location, where the calculated addition will be stored.



```
<rul:Sum>
<rul:Product>
  <rul:FirstValue Type="decimal" FromConstant="0.0" FromDataDef="FinalRate"
Order="DataDefInputParamConstant" />
  <rul:Value Type="decimal" FromParam="calcExposureLessOrEqualOneMillionOrHundred"
/>
</rul:Product>
<rul:Value Type="decimal" FromParam="calcPremiumOverMillionOrHundred" />
</rul:Sum>
```

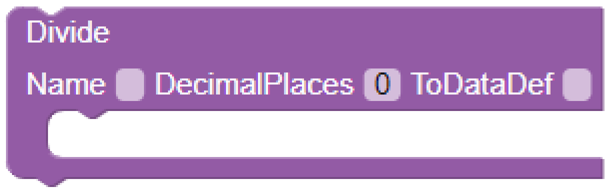

Sum ToDataDef Premium
Value
ToDataDef FromDataDef BasicPremium
Type decimal Allow Null Return true
From Param
Value
ToDataDef FromDataDef ConvertedLoss
Type decimal Allow Null Return true
From Param

- **rul:Divide** : It returns division of given fields.

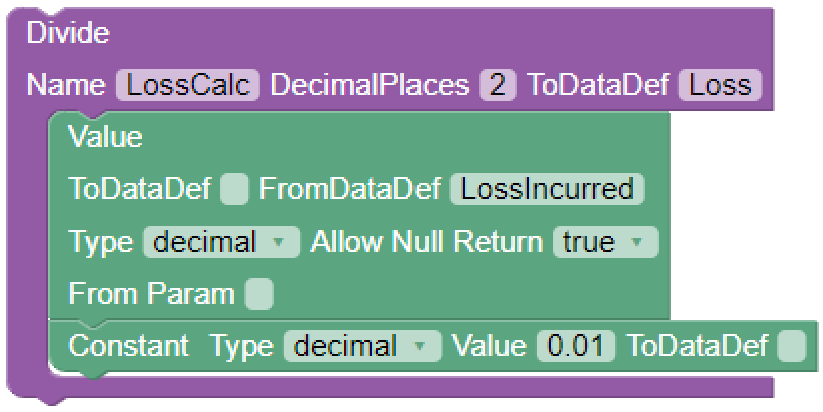
Name : It specifies name of the division rule.

DecimalPlaces : For the division result, it specifies the no. of places after decimal point.

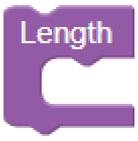
ToDataDef : The mentioned xpath points to the location, where the division result will be stored.


Divide
Name DecimalPlaces 0 ToDataDef

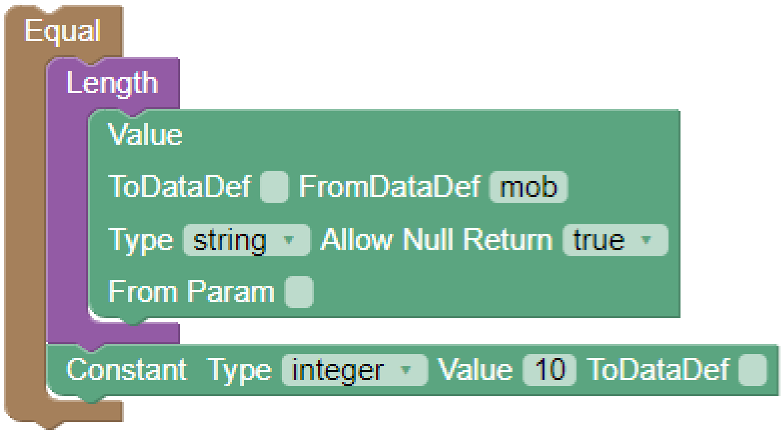
```
<rul:Divide ToDataDef="ERPSublinePresentAvgRatePremOps">
  <rul:Value Type="decimal" FromDataDef="AnnualBasicLimitsCoPremiumPremOps" />
  <rul:Value Type="long" FromDataDef="ERPExposuresOnSpecialUWBasisPremOpsCurrent"
/>
</rul:Divide>
```


Divide
Name LossCalc DecimalPlaces 2 ToDataDef Loss
Value
ToDataDef FromDataDef LossIncurred
Type decimal Allow Null Return true
From Param
Constant Type decimal Value 0.01 ToDataDef

- **rul:Length** : It returns length of field.

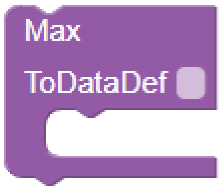


```
<rul:Equal>
  <rul:Length>
    <rul:Value Type="string" FromParam="prodWithLCM" />
  </rul:Length>
  <rul:Constant Type="integer">3</rul:Constant>
</rul:Equal>
```

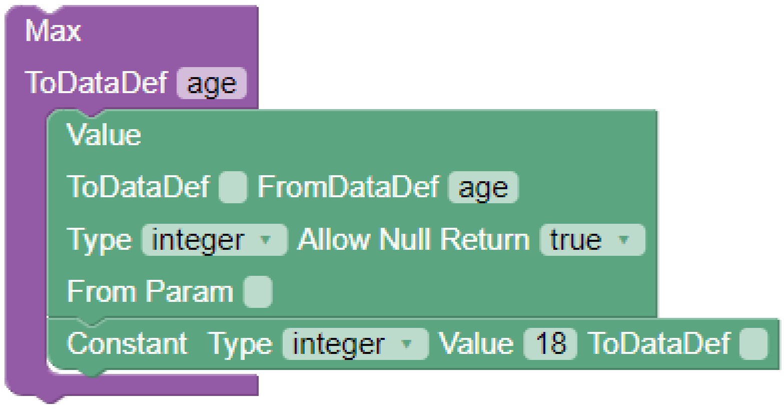


- **rul:Max** : It returns greatest value among given fields.

ToDataDef : The mentioned xpath points to the location, where greatest value among given fields will be stored.

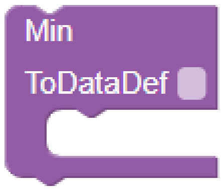


```
<rul:Max>
  <rul:ForEach
AtDataDef="../GeneralLiabilityLocationTable/GeneralLiabilityLocation">
    <rul:ForEach
AtDataDef="GeneralLiabilityClassificationTable/GeneralLiabilityClassification">
      <rul:FirstValue Type="decimal" FromConstant="0.0"
FromDataDef="GeneralLiabilityClassificationPremOpsCoverage/MinPremium"
Order="DataDefInputParamConstant" />
    </rul:ForEach>
  </rul:ForEach>
  <rul:Constant Type="decimal">0</rul:Constant>
</rul:Max>
```

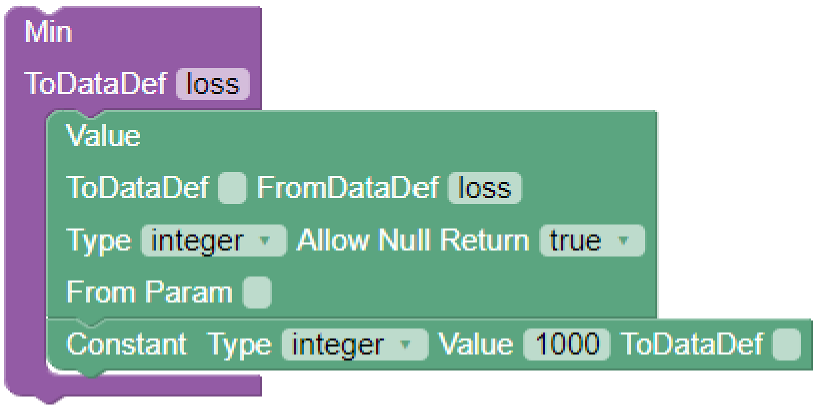

Max
ToDataDef age
Value
ToDataDef FromDataDef age
Type integer Allow Null Return true
From Param
Constant Type integer Value 18 ToDataDef

- **rul:Min** : It returns least value among given fields.

ToDataDef : The mentioned xpath points to the location, where lowest value among given fields will be stored.



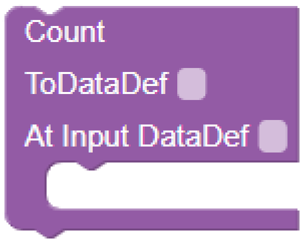
```
<rul:Min>
  <rul:ForEach
AtDataDef="../GeneralLiabilityLocationTable/GeneralLiabilityLocation">
    <rul:ForEach
AtDataDef="GeneralLiabilityClassificationTable/GeneralLiabilityClassification">
      <rul:FirstValue Type="decimal" FromConstant="0.0"
FromDataDef="GeneralLiabilityClassificationPremOpsCoverage/MinPremium"
Order="DataDefInputParamConstant" />
    </rul:ForEach>
  </rul:ForEach>
  <rul:Constant Type="decimal">0</rul:Constant>
</rul:Min>
```


Min
ToDataDef loss
Value
ToDataDef FromDataDef loss
Type integer Allow Null Return true
From Param
Constant Type integer Value 1000 ToDataDef

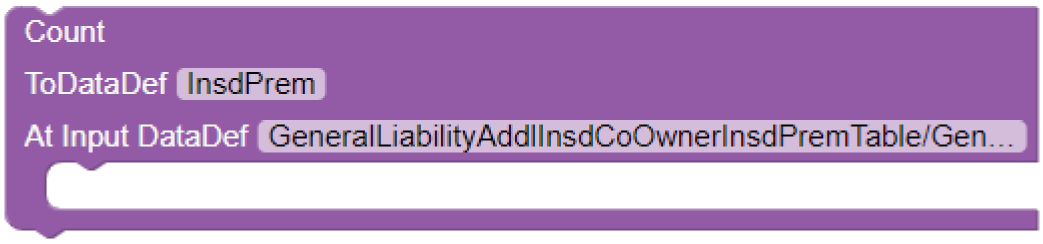
- **rul:Count** : It returns no. of elements under xpath mentioned in *AtInputDataDef*.

ToDataDef : It saves no. of elements under xpath mentioned in *AtInputDataDef*.

AtInputDataDef : The mentioned xpath points value into the context, without changing the current pointer.



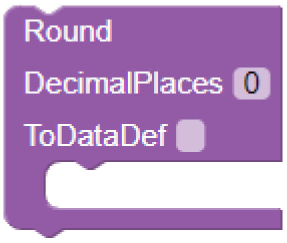
```
<rul:Count
AtInputDataDef="GeneralLiabilityAddlInsdCoOwnerInsdPremTable/GeneralLiabilityAddlI
nsdCoOwnerInsdPrem" />
```


Count
ToDataDef InsdPrem
At Input DataDef GeneralLiabilityAddlInsdCoOwnerInsdPremTable/Gen...

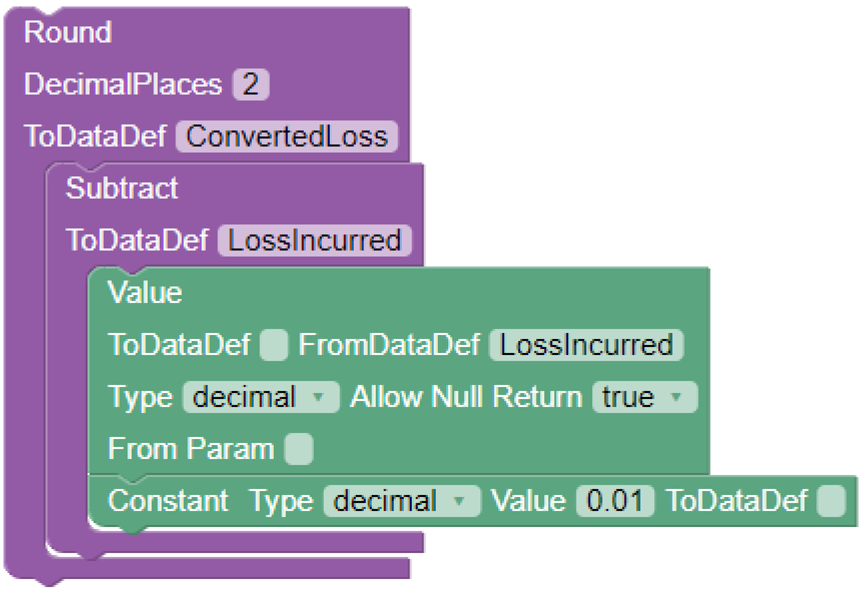
- **rul:Round** : It returns the calculated result with specified no. of places after decimal point.

DecimalPlaces : It mentions the no. of places after decimal point.

ToDataDef : The mentioned xpath points to the location, where the rounded result will be stored.



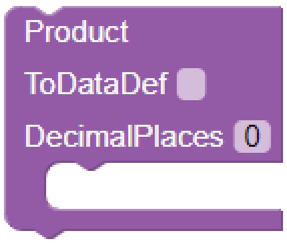
```
<rul:Round ToDataDef="FinalILF" DecimalPlaces="3">
  <rul:Subtract>
    <rul:Value Type="decimal" FromDataDef="CSLILF" />
    <rul:Value Type="decimal" FromDataDef="DeductibleFactor" />
  </rul:Subtract>
</rul:Round>
```


Round
DecimalPlaces 2
ToDataDef ConvertedLoss
Subtract
ToDataDef LossIncurred
Value
ToDataDef FromDataDef LossIncurred
Type decimal Allow Null Return true
From Param
Constant Type decimal Value 0.01 ToDataDef

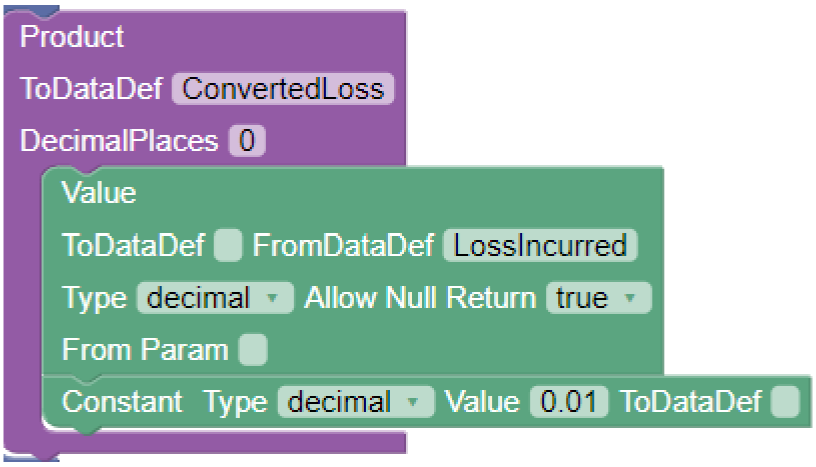
- **rul:Product** : It returns multiplication result of given fields.

DecimalPlaces : It mentions the no. of places after decimal point.

ToDataDef : The mentioned xpath points to the location, where multiplication result will be stored.



```
<rul:Product ToDataDef="Premium" DecimalPlaces="0">
  <rul:FirstValue Type="decimal" FromConstant="0.0" FromDataDef="ManualPremium"
Order="DataDefInputParamConstant" />
  <rul:FirstValue Type="decimal" FromConstant="0.0"
FromDataDef="../../PackageModFactor" Order="DataDefInputParamConstant" />
</rul:Product>
```


Product
ToDataDef ConvertedLoss
DecimalPlaces 0
Value
ToDataDef FromDataDef LossIncurred
Type decimal Allow Null Return true
From Param
Constant Type decimal Value 0.01 ToDataDef

Value Section

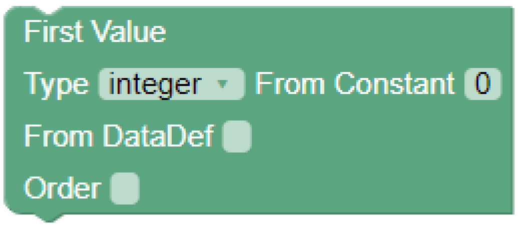
- **rul:FirstValue** : It returns first value

Type : It mentions type of first value.

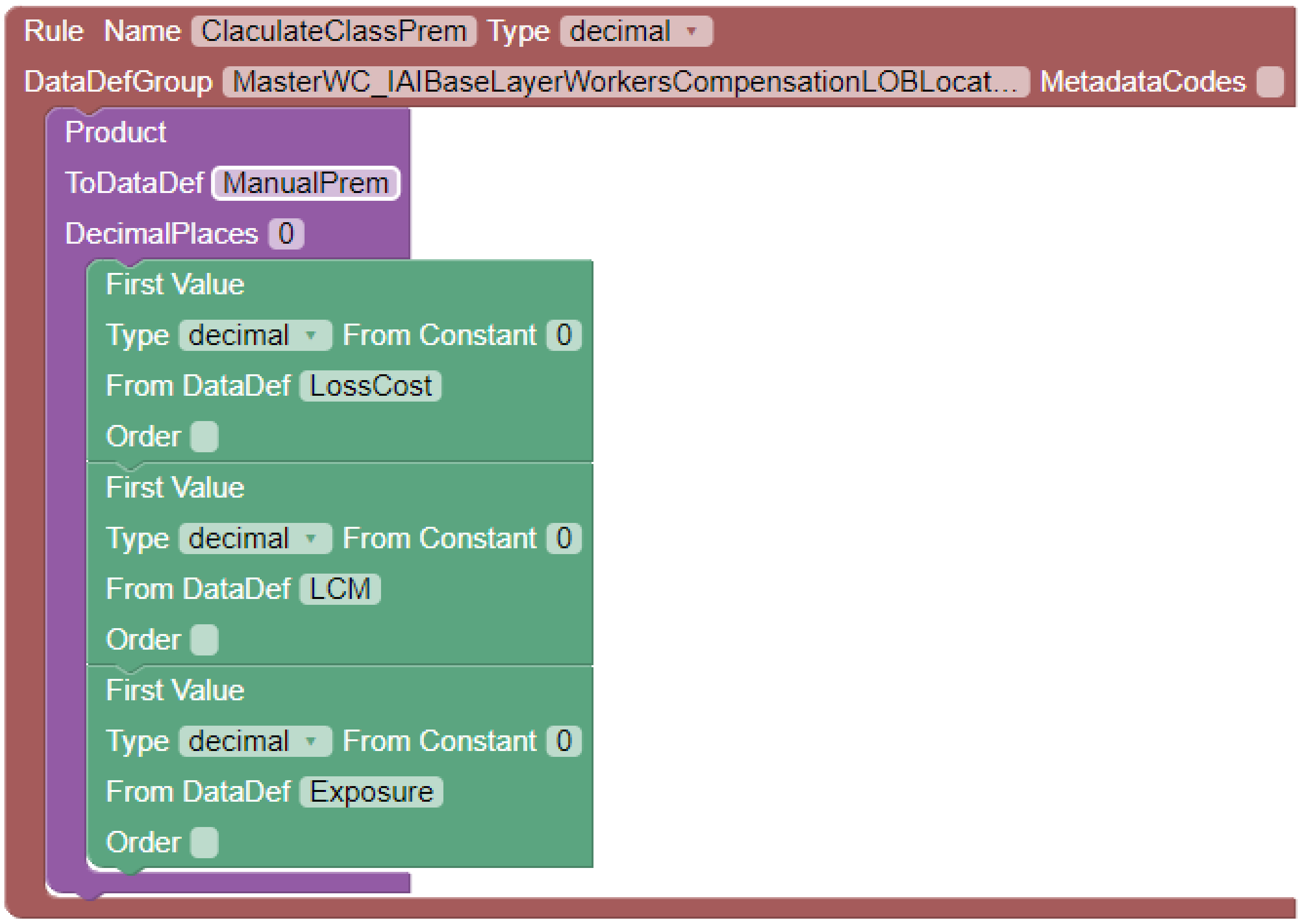
FromConstant : To set default value.

FromDataDef : The mentioned xpath points to the location, from where we fetch value.

Order : To get value from user as input.



```
<rul:Equal>
  <rul:FirstValue Type="integer" FromConstant="0" FromDataDef="PremiumIndicator"
Order="DataDefInputParamConstant" />
  <rul:Constant Type="integer">1</rul:Constant>
</rul:Equal>
```

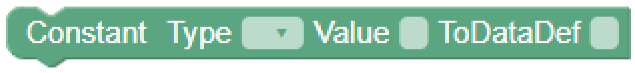

Rule Name ClaculateClassPrem Type decimal
DataDefGroup MasterWC_IAIBaseLayerWorkersCompensationLOBLocat... MetadataCodes
Product
ToDataDef ManualPrem
DecimalPlaces 0
First Value
Type decimal From Constant 0
From DataDef LossCost
Order
First Value
Type decimal From Constant 0
From DataDef LCM
Order
First Value
Type decimal From Constant 0
From DataDef Exposure
Order

- **rul:Constant** : It defines a constant.

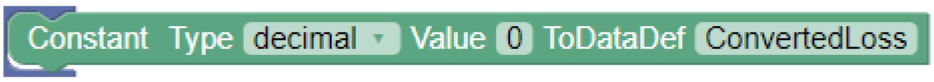
Type : It mentions type of constant.

Value : To set value of constant.

ToDataDef : The mentioned xpath points to the location, to store constant.



```
<rul:Constant Type="integer">1</rul:Constant>
```



- **rul:Value** : It defines a value. It can get value from pre-defined Param, XML or set value to XML.

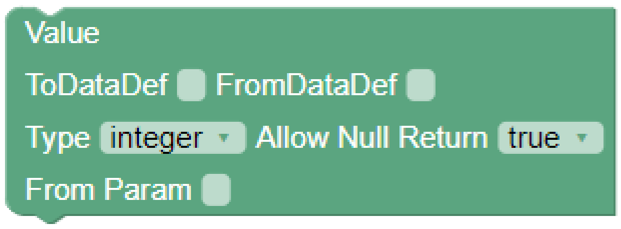
Type : It mentions type of value.

AllowNullReturn : It can have null as return type.

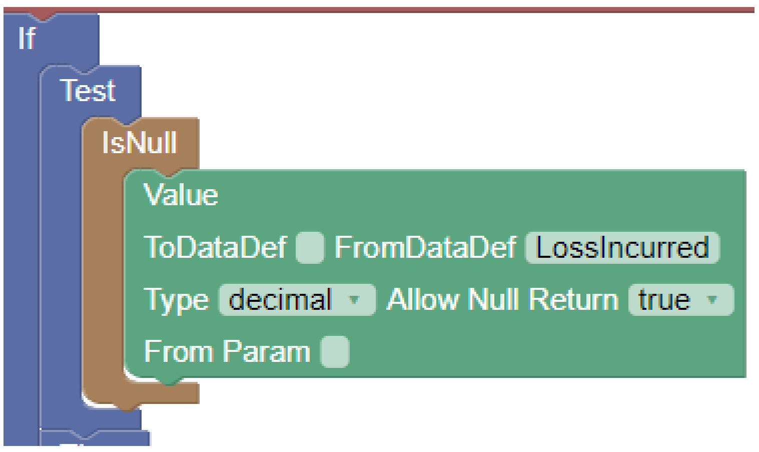
ToDataDef : The mentioned xpath points to the location, to store value.

FromDataDef : The mentioned xpath points to the location, to fetch value from.

FromParam : To get value from pre-defined parameter.



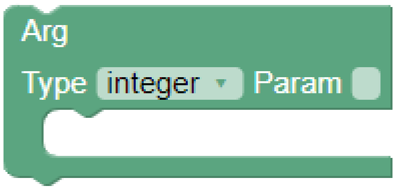
```
<rul:Value Type="string" FromDataDef="/*/State/Code" />
```


If
Test
IsNull
Value
ToDataDef FromDataDef LossIncurred
Type decimal Allow Null Return true
From Param

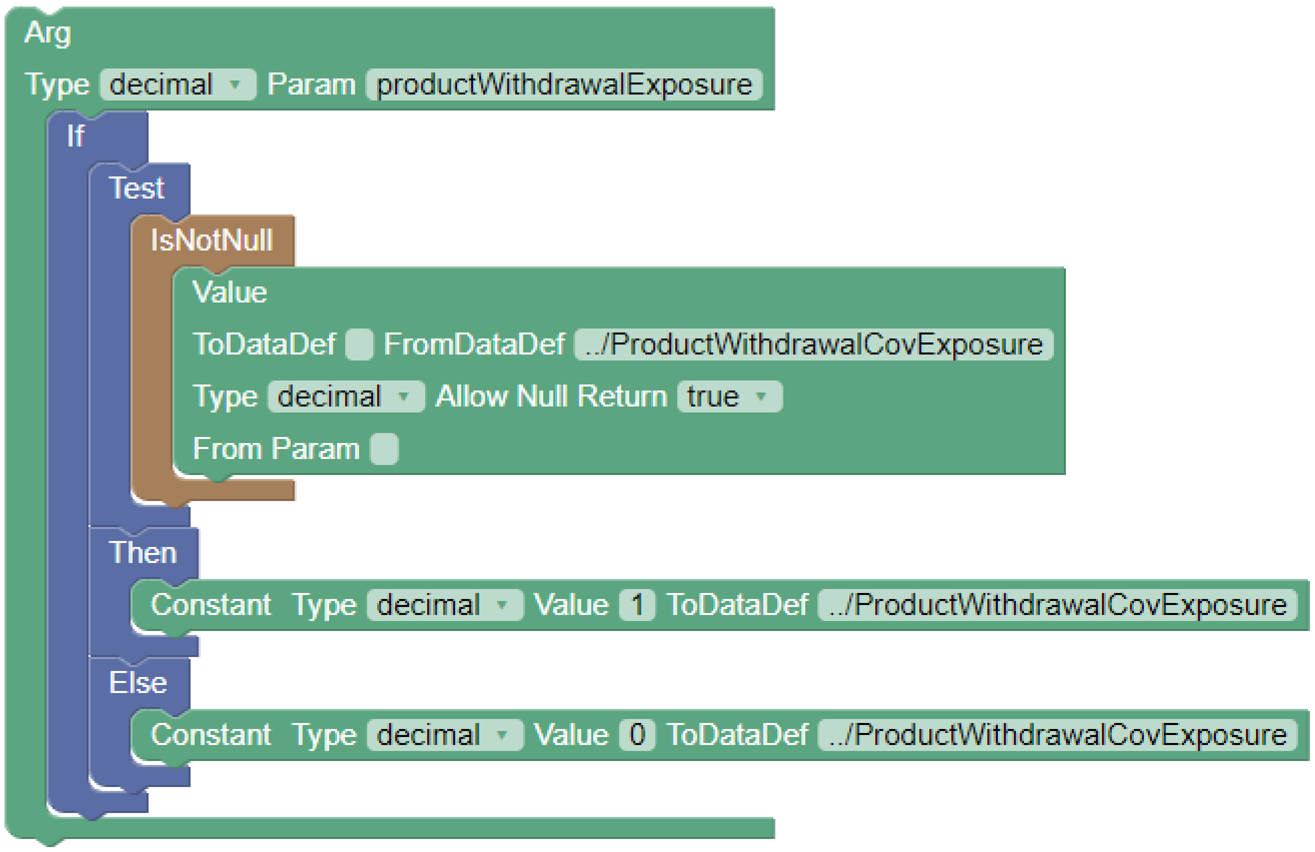
- **rul:Arg** : It denotes an argument (actual parameter).

Type : It specifies type of filed.

Param : It is used to specify field name to refer to.

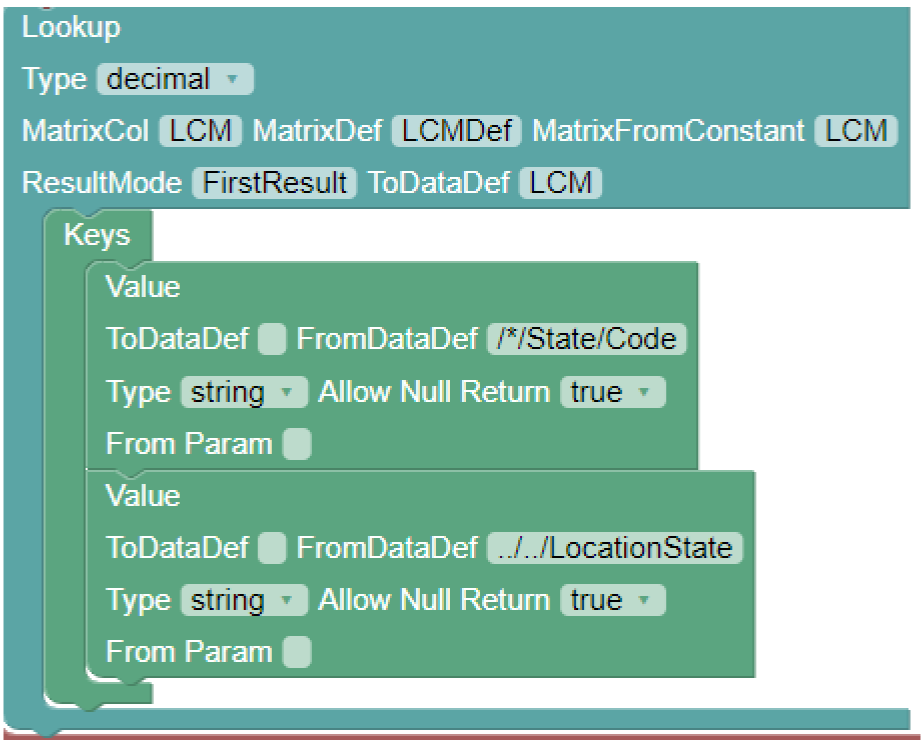


```
<rul:Arg Type="decimal" Param="productWithdrawalExposure">
  <rul:If>
    <rul:Test>
      <rul:IsNotNull>
        <rul:Value Type="decimal" FromDataDef="../ProductWithdrawalCovExposure" />
      </rul:IsNotNull>
    </rul:Test>
    <rul:Then>
      <rul:FirstValue Type="decimal" FromConstant="0.0"
FromDataDef="../ProductWithdrawalCovExposure" Order="DataDefInputParamConstant" />
    </rul:Then>
    <rul:Else>
      <rul:Constant Type="decimal">0</rul:Constant>
    </rul:Else>
  </rul:If>
</rul:Arg>
```


Arg
Type decimal Param productWithdrawalExposure
If
Test
IsNotNull
Value
ToDataDef FromDataDef ../ProductWithdrawalCovExposure
Type decimal Allow Null Return true
From Param
Then
Constant Type decimal Value 1 ToDataDef ../ProductWithdrawalCovExposure
Else
Constant Type decimal Value 0 ToDataDef ../ProductWithdrawalCovExposure

- **rul:Keys** : Key is used to map value from rate table.

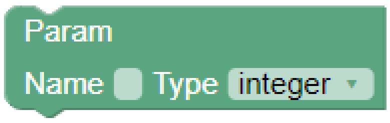
```
<rul:Lookup Type="string" MatrixCol="Code" MatrixDef="AddlInsdStatCodeDef"
MatrixFromConstant="AddlInsdStatCode" ResultMode="FirstResult">
  <rul:Keys>
    <rul:Value Type="string" FromDataDef="/*/State/Code" />
    <rul:Constant Type="string">Y</rul:Constant>
  </rul:Keys>
</rul:Lookup>
```


Lookup
Type decimal
MatrixCol LCM MatrixDef LCMDef MatrixFromConstant LCM
ResultMode FirstResult ToDataDef LCM
Keys
Value
ToDataDef FromDataDef /*/State/Code
Type string Allow Null Return true
From Param
Value
ToDataDef FromDataDef ../../LocationState
Type string Allow Null Return true
From Param

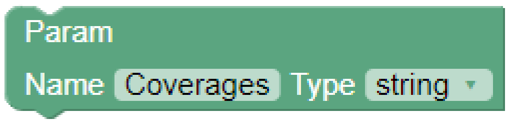
- **rul:Param** : It defines a formal parameter. We can fetch its value using attribute FromParam

Type : It mentions type of parameter.

Name : It specifies name of parameter.



```
<rul:Param Name="Coverages" Type="string" />
```



Data Manipulations Section

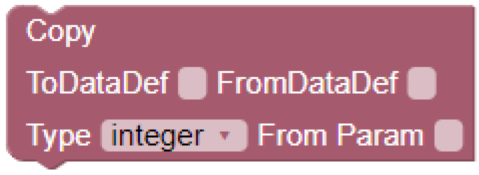
- **rul:Copy** : It saves value of field from *FromDataDef* location to *ToDataDef* location.

Type : It mentions the type of field.

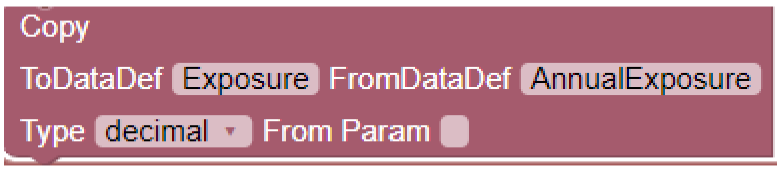
ToDataDef : It mentions the destination location using xpath, where we want to save the field value.

FromDataDef : It mentions the source location using xpath, from where we want to save the field value.

FromParam : It mentions name of pre-defined parameter.


Copy
ToDataDef FromDataDef
Type integer From Param

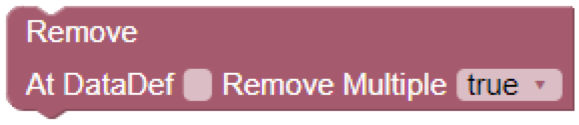
```
<rul:Copy Type="decimal" ToDataDef="LiabilityLossCostMultiplier"
FromDataDef="../../../../../PolicyLiabilityLossCostMultiplier" />
```


Copy
ToDataDef Exposure FromDataDef AnnualExposure
Type decimal From Param

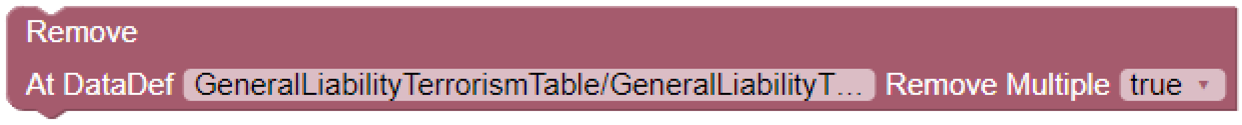
- **rul:Remove** : It eliminates elements falling under xpath, mentioned in *AtDataDef*.

AtDataDef : It mentions the destination location using xpath, where we want to save the field value.

RemoveMultiple : To eliminate all occurrences.

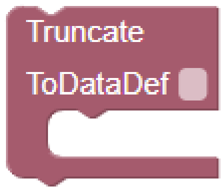

Remove
At DataDef Remove Multiple true

```
<rul:Sequence>
  <rul:Remove AtDataDef="GeneralLiabilityTerrorismTable/GeneralLiabilityTerrorism"
RemoveMultiple="true" />
</rul:Sequence>
```

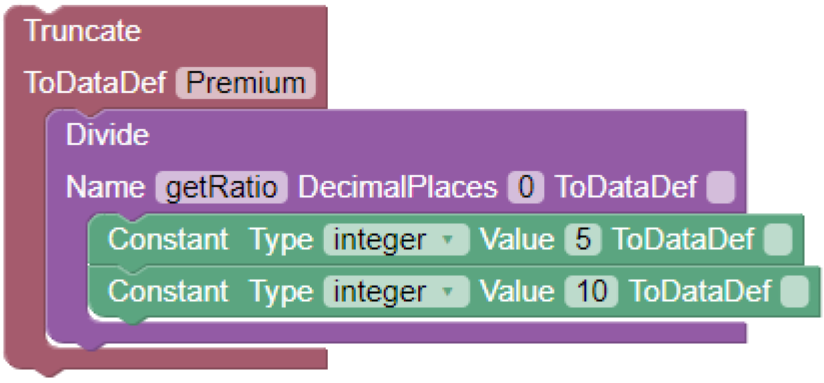

Remove
At DataDef GeneralLiabilityTerrorismTable/GeneralLiabilityT... Remove Multiple true

- **rul:Truncate** : It rounds a number down to the nearest integer and returns result.

ToDataDef : It mentions the destination location using xpath, where we want to save the result.

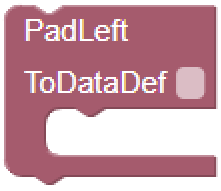

Truncate
ToDataDef

```
<rul:Truncate>
  <rul:Divide DecimalPlaces="0">
    <rul:Value Type="long" FromDataDef="../LiquorExposure" />
    <rul:Value Type="integer" FromParam="reportingBasisLiquor" />
  </rul:Divide>
</rul:Truncate>
```

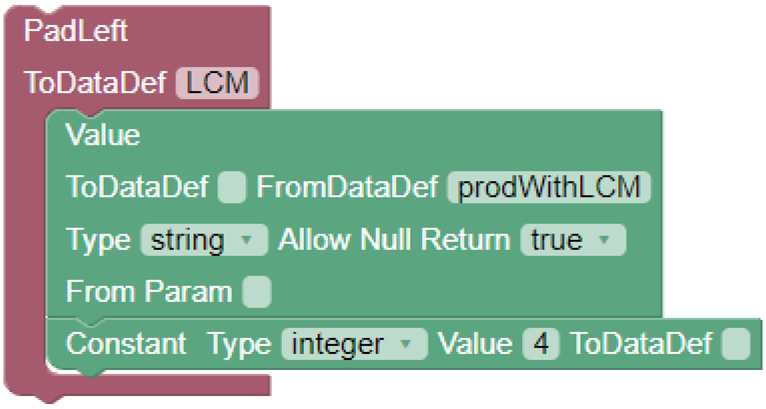

Truncate
ToDataDef Premium
Divide
Name getRatio DecimalPlaces 0 ToDataDef
Constant Type integer Value 5 ToDataDef
Constant Type integer Value 10 ToDataDef

- **rul:PadLeft** : It will right align the string, using a specified character (space is default) as the fill character.

ToDataDef : It mentions the destination location using xpath, where we want to save the aligned result.



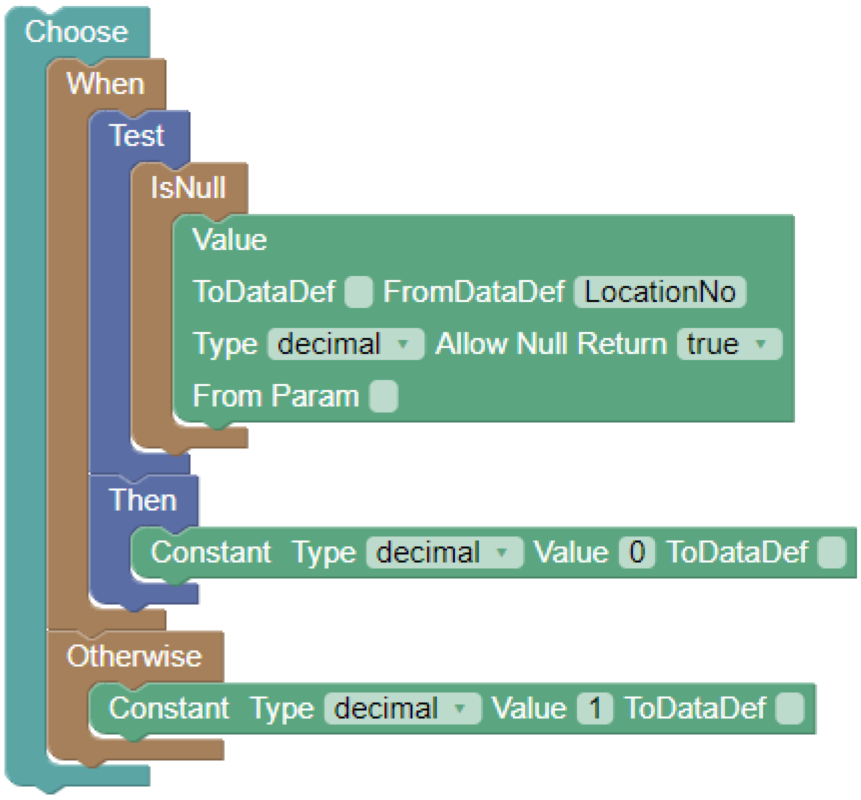
```
<rul:PadLeft ToDataDef="LCMStatCode">
  <rul:Value Type="string" FromParam="prodWithLCM" />
  <rul:Constant Type="integer">3</rul:Constant>
  <rul:Constant Type="string">0</rul:Constant>
</rul:PadLeft>
```


PadLeft
ToDataDef LCM
Value
ToDataDef FromDataDef prodWithLCM
Type string Allow Null Return true
From Param
Constant Type integer Value 4 ToDataDef

Find Operations Section

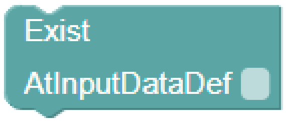
- **rul:Choose** : It runs a set of statements, based on condition.

```
<rul:Choose>
  <rul:When>
    <rul:Test>
      <rul:Equal>
        <rul:Value Type="string" FromParam="productWithdrawalPremiumBasis" />
        <rul:Constant Type="string">No Exposure</rul:Constant>
      </rul:Equal>
    </rul:Test>
    <rul:Then>
      <rul:Sequence>
        <rul:Constant Type="string" ToDataDef="ExposureStatCode"></rul:Constant>
      </rul:Sequence>
    </rul:Then>
  </rul:When>
  <rul:Otherwise>
    <rul:Sequence>
      <rul:Constant Type="string" ToDataDef="ExposureStatCode"></rul:Constant>
    </rul:Sequence>
  </rul:Otherwise>
</rul:Choose>
```

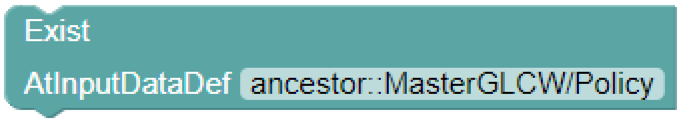

Choose
When
Test
IsNull
Value
ToDataDef FromDataDef LocationNo
Type decimal Allow Null Return true
From Param
Then
Constant Type decimal Value 0 ToDataDef
Otherwise
Constant Type decimal Value 1 ToDataDef

- **rul:Exist** : To check if a element(located by AtInputDataDef) is present.

AtInputDataDef : It mentions the source location using xpath, from where we fetch the field.



```
<rul:Exist AtInputDataDef="ancestor::MasterGLCW/Policy" />
```



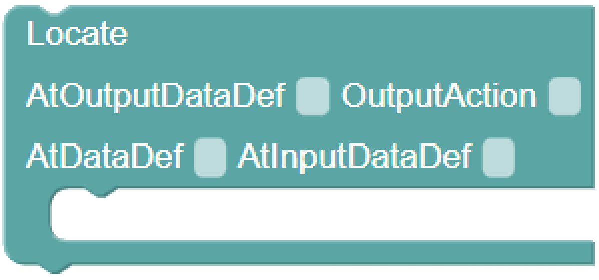
- **rul:Locate** : It points to specific node of XML tree structure.

AtOutputDataDef : The mentioned xpath points value into the context, without changing the current pointer. This is used to store result to node.

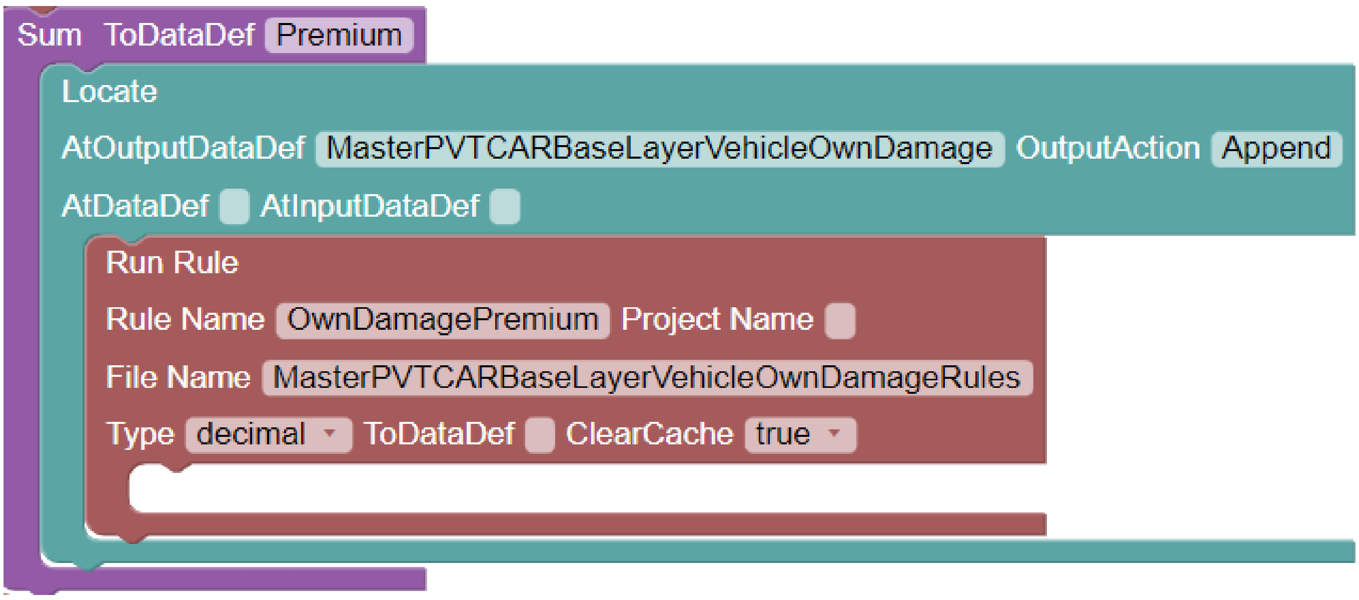
OutputAction : It specifies more methods to do at *AtOutputDataDef* location.

AtDataDef : It mentions the location using xpath, pointing to the field.

AtInputDataDef : The mentioned xpath points value into the context, without changing the current pointer. This is used to fetch value from node.



```
<rul:Locate AtOutputDataDef="GeneralLiabilityClassificationLiquorCoverage"
OutputAction="Append">
  <rul:Sequence />
</rul:Locate>
```


Sum ToDataDef Premium
Locate
AtOutputDataDef MasterPVTCARBaseLayerVehicleOwnDamage OutputAction Append
AtDataDef AtInputDataDef
Run Rule
Rule Name OwnDamagePremium Project Name
File Name MasterPVTCARBaseLayerVehicleOwnDamageRules
Type decimal ToDataDef ClearCache true

- **rul:Lookup** : It fetches value from rate table, based on corresponding key.

RateTableReturnType : It mentions type of value fetched from rate table.

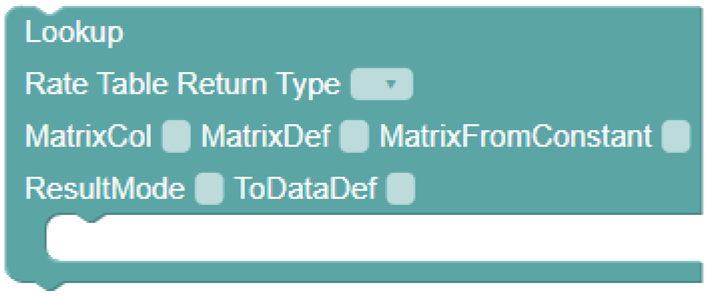
MatrixCol : It specifies name of key column.

MatrixDef : It has name of rate table+Def. This is used by backend script for key-value pair mapping.

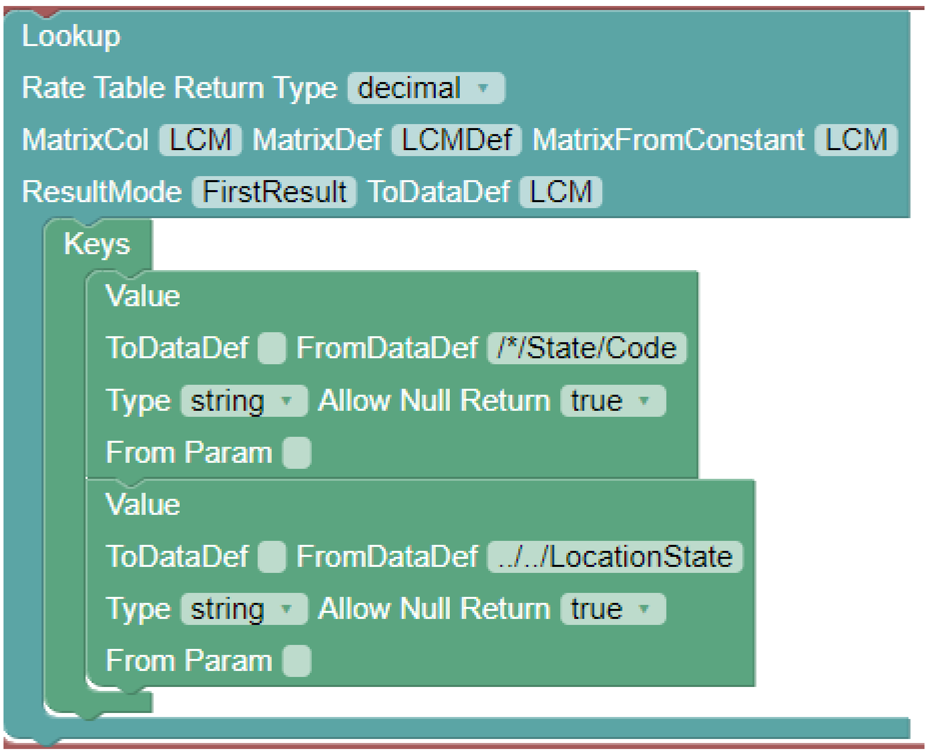
MatrixFromConstant : It suggests name of rate table.

ResultMode : It specifies form of result.

ToDataDef : The mentioned xpath points to the location, to store mapped value.

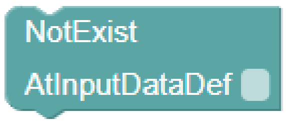


```
<rul:Lookup Type="string" MatrixCol="Code" MatrixDef="AddlInsdStatCodeDef"
MatrixFromConstant="AddlInsdStatCode" ResultMode="FirstResult">
  <rul:Keys>
    <rul:Value Type="string" FromDataDef="/*/State/Code" />
    <rul:Constant Type="string">Y</rul:Constant>
  </rul:Keys>
</rul:Lookup>
```

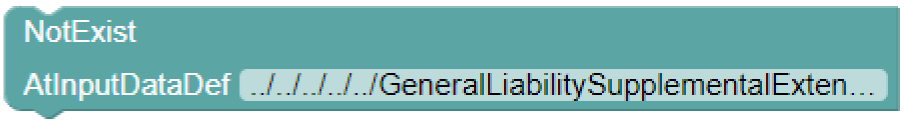

Lookup
Rate Table Return Type decimal
MatrixCol LCM MatrixDef LCMDef MatrixFromConstant LCM
ResultMode FirstResult ToDataDef LCM
Keys
Value
ToDataDef FromDataDef /*/State/Code
Type string Allow Null Return true
From Param
Value
ToDataDef FromDataDef ../../LocationState
Type string Allow Null Return true
From Param

- **rul:NotExist** : To check if field is not present.

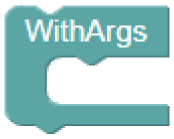
AtInputDataDef : The mentioned xpath points value into the context, without changing the current pointer.



```
<rul:NotExist
AtInputDataDef="../../../../../GeneralLiabilitySupplementalExtendedReportingPeriod
EndtLiquorLiabTable/GeneralLiabilitySupplementalExtendedReportingPeriodEndtLiquorL
iab" />
```

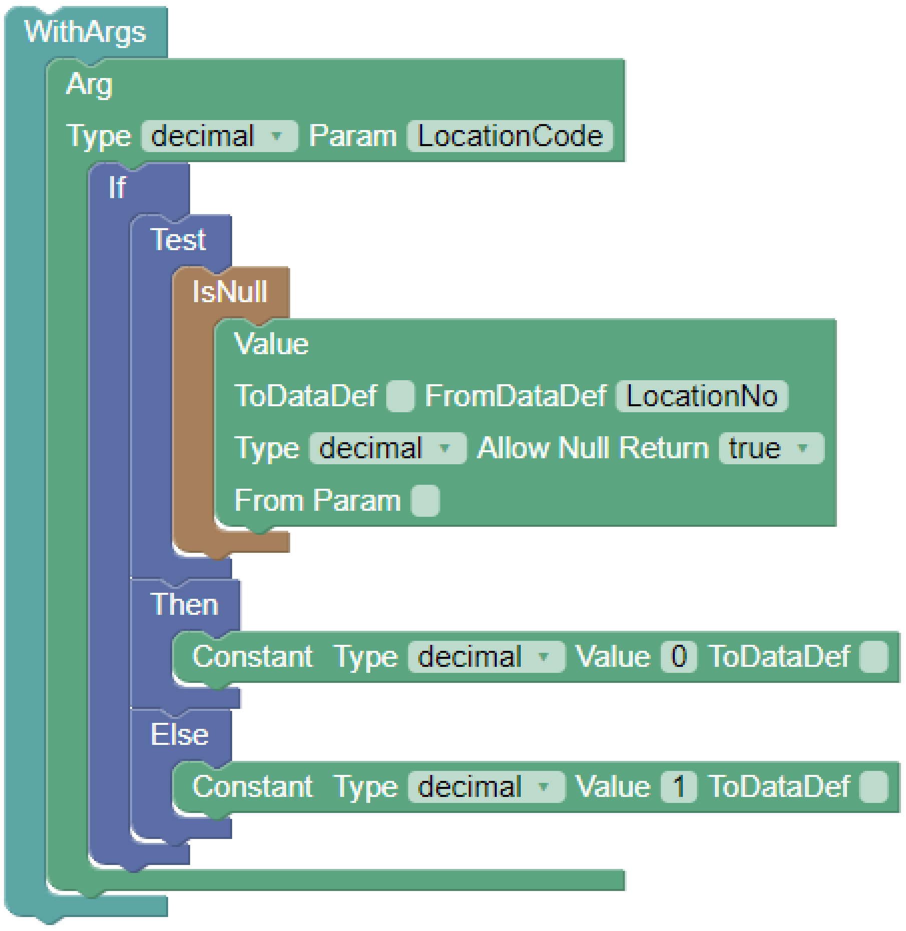

NotExist
AtInputDataDef ../../../../../GeneralLiabilitySupplementalExten...

- **rul:WithArgs** : To use previously defined or calculated field.



```
<rul:WithArgs>
    <rul:Arg Type="string" Param="productWithdrawalPremiumBasis">
        <rul:If>
            <rul:Test>
                <rul:IsNotNull>
```

```
                        <rul:Value Type="string"
FromDataDef="../ProductWithdrawalPremiumBasis" />
                    </rul:IsNotNull>
                </rul:Test>
                <rul:Then>
                    <rul:FirstValue Type="string" FromConstant=""
FromDataDef="../ProductWithdrawalPremiumBasis" Order="DataDefInputParamConstant"
/>
                </rul:Then>
                <rul:Else>
                    <rul:Constant Type="string"></rul:Constant>
                </rul:Else>
            </rul:If>
        </rul:Arg>
</rul:WithArgs>
```

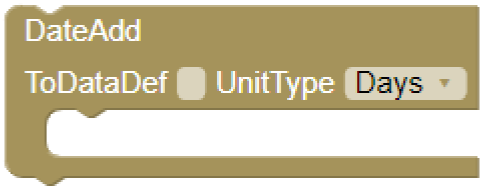

WithArgs
Arg
Type decimal Param LocationCode
If
Test
IsNull
Value
ToDataDef FromDataDef LocationNo
Type decimal Allow Null Return true
From Param
Then
Constant Type decimal Value 0 ToDataDef
Else
Constant Type decimal Value 1 ToDataDef

Date Operations Section

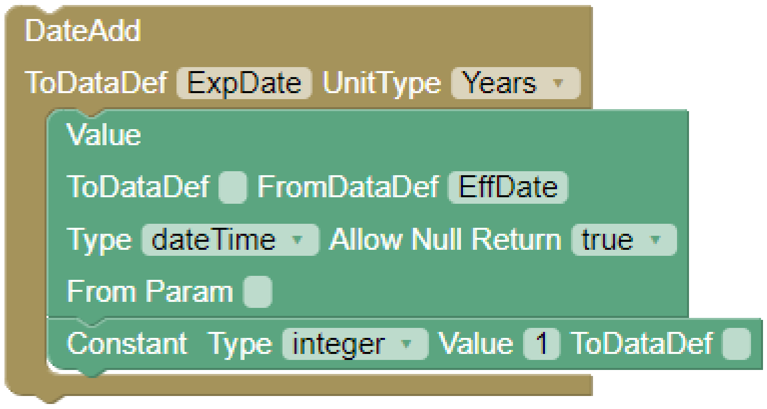
- **rul:DateAdd** : To add duration to date given in *FromDataDef*.

ToDataDef : The mentioned xpath points to the location, where the date with addition will be stored.

UnitType : It represents unit of date (eg. days, months, years).

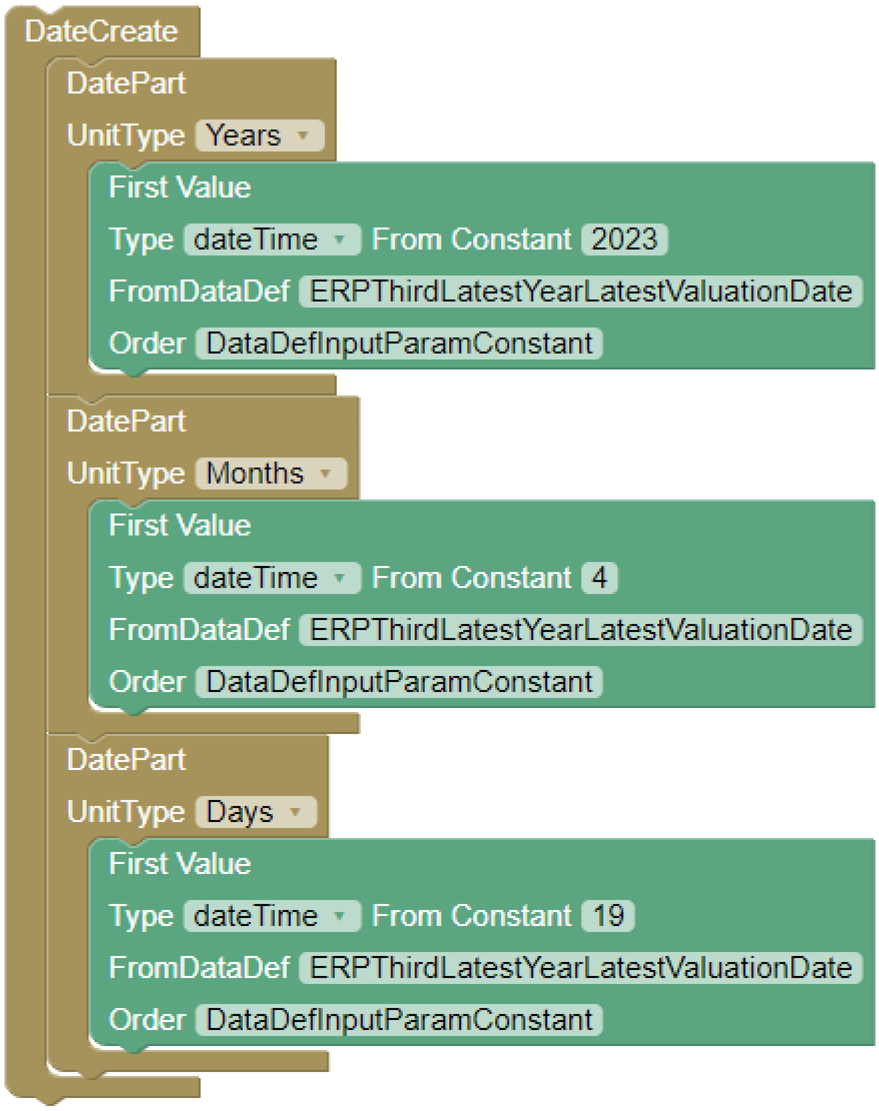


```
<rul:DateAdd ToDataDef="ExpDate" UnitType="Years">
  <rul:Value Type="dateTime" FromDataDef="EffDate" />
  <rul:Constant Type="integer">1</rul:Constant>
</rul:DateAdd>
```


DateAdd
ToDataDef ExpDate UnitType Years
Value
ToDataDef FromDataDef EffDate
Type dateTime Allow Null Return true
From Param
Constant Type integer Value 1 ToDataDef

- **rul:DateCreate** : To write a date.

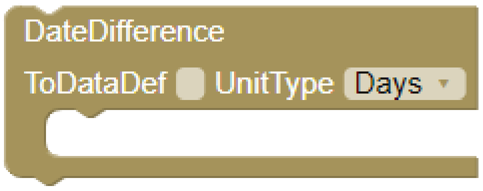
```
<rul:DateCreate>
  <rul:DatePart UnitType="Years">
    <rul:FirstValue Type="dateTime" FromConstant="01/01/0001"
FromDataDef="ERPThirdLatestYearLatestValuationDate"
Order="DataDefInputParamConstant" />
  </rul:DatePart>
  <rul:DatePart UnitType="Months">
    <rul:FirstValue Type="dateTime" FromConstant="01/01/0001"
FromDataDef="ERPThirdLatestYearLatestValuationDate"
Order="DataDefInputParamConstant" />
  </rul:DatePart>
  <rul:DatePart UnitType="Days">
    <rul:FirstValue Type="dateTime" FromConstant="01/01/0001"
FromDataDef="ERPThirdLatestYearLatestValuationDate"
Order="DataDefInputParamConstant" />
  </rul:DatePart>
</rul:DateCreate>
```


DateCreate
DatePart
UnitType Years
First Value
Type dateTime From Constant 2023
FromDataDef ERPThirdLatestYearLatestValuationDate
Order DataDefInputParamConstant
DatePart
UnitType Months
First Value
Type dateTime From Constant 4
FromDataDef ERPThirdLatestYearLatestValuationDate
Order DataDefInputParamConstant
DatePart
UnitType Days
First Value
Type dateTime From Constant 19
FromDataDef ERPThirdLatestYearLatestValuationDate
Order DataDefInputParamConstant

- **rul:DateDifference** : To find difference in dates.

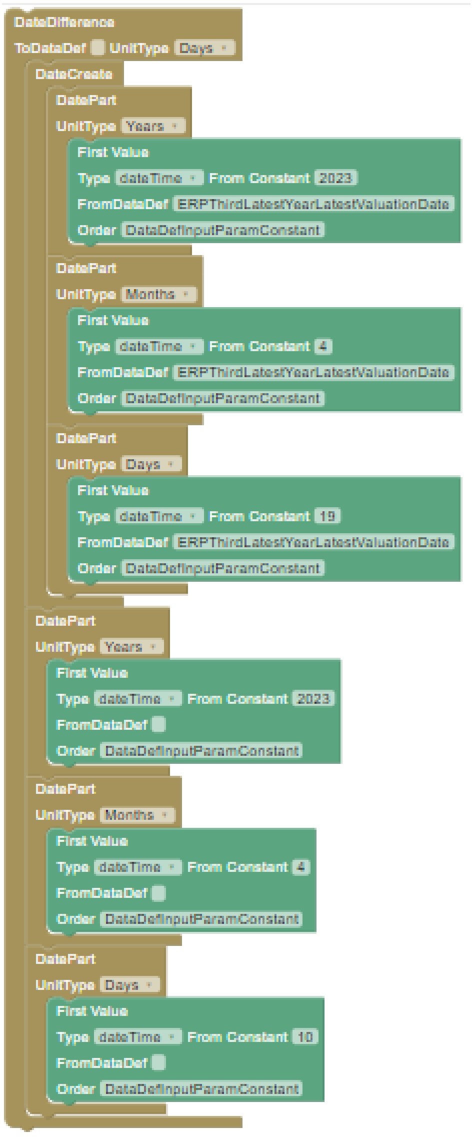
ToDataDef : The mentioned xpath points to the location, where the calculation of difference in dates will be stored.

UnitType : It represents unit of date (eg. days, months, years).



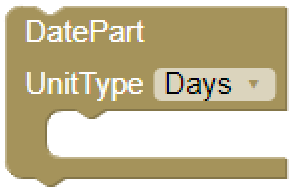
```
<rul:DateDifference UnitType="Days">
    <rul:DateCreate>
        <rul:DatePart UnitType="Years">
            <rul:FirstValue Type="dateTime" FromConstant="01/01/0001"
FromDataDef="ERPLatestYearLatestValuationDate" Order="DataDefInputParamConstant"
/>
        </rul:DatePart>
        <rul:DatePart UnitType="Months">
            <rul:FirstValue Type="dateTime" FromConstant="01/01/0001"
```

```
FromDataDef="ERPLatestYearLatestValuationDate" Order="DataDefInputParamConstant"
/>
        </rul:DatePart>
        <rul:DatePart UnitType="Days">
            <rul:FirstValue Type="dateTime" FromConstant="01/01/0001"
FromDataDef="ERPLatestYearLatestValuationDate" Order="DataDefInputParamConstant"
/>
        </rul:DatePart>
    </rul:DateCreate>
    <rul:DateCreate>
        <rul:DatePart UnitType="Years">
            <rul:FirstValue Type="dateTime" FromConstant="01/01/0001"
FromDataDef="ERPLatestYearEffectiveDate" Order="DataDefInputParamConstant" />
        </rul:DatePart>
        <rul:DatePart UnitType="Months">
            <rul:FirstValue Type="dateTime" FromConstant="01/01/0001"
FromDataDef="ERPLatestYearEffectiveDate" Order="DataDefInputParamConstant" />
        </rul:DatePart>
        <rul:DatePart UnitType="Days">
            <rul:FirstValue Type="dateTime" FromConstant="01/01/0001"
FromDataDef="ERPLatestYearEffectiveDate" Order="DataDefInputParamConstant" />
        </rul:DatePart>
    </rul:DateCreate>
</rul:DateDifference>
```

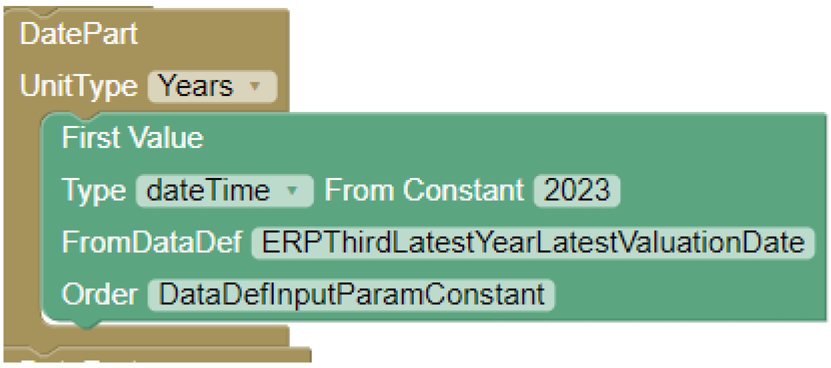

DateDifference
ToDataDef UnitType Days
DateCreate
DatePart
UnitType Years
First Value
Type dateTime From Constant 2023
FromDataDef ERPThirdLatestYearLatestValuationDate
Order DataDefInputParamConstant
DatePart
UnitType Months
First Value
Type dateTime From Constant 4
FromDataDef ERPThirdLatestYearLatestValuationDate
Order DataDefInputParamConstant
DatePart
UnitType Days
First Value
Type dateTime From Constant 19
FromDataDef ERPThirdLatestYearLatestValuationDate
Order DataDefInputParamConstant
DatePart
UnitType Years
First Value
Type dateTime From Constant 2023
FromDataDef
Order DataDefInputParamConstant
DatePart
UnitType Months
First Value
Type dateTime From Constant 4
FromDataDef
Order DataDefInputParamConstant
DatePart
UnitType Days
First Value
Type dateTime From Constant 10
FromDataDef
Order DataDefInputParamConstant

- **rul:DatePart** : It is used to get year or month or day value from date value, like 2023-05-17.

UnitType : It represents unit of date (eg. days, months, years).


DatePart
UnitType Days

```
<rul:DatePart UnitType="Years">
    <rul:FirstValue Type="dateTime" FromConstant="01/01/0001"
FromDataDef="ERPLatestYearLatestValuationDate" Order="DataDefInputParamConstant"
/>
</rul:DatePart>
```


DatePart
UnitType Years
First Value
Type dateTime From Constant 2023
FromDataDef ERPThirdLatestYearLatestValuationDate
Order DataDefInputParamConstant

The Attribute Tags

Some commonly used Attribute tags are :

rul:Constant = { "Type": ["boolean", "dateTime",...], "ToDataDef": ["../WorkersCompWaiverOfOurRightToRecoverFromOthersEndorsementDetailCoverage/SpecificMinPremiumIndicator", "ARDIndicator",..]}

rul:Copy = { "Type": ["dateTime", "decimal", ..], "ToDataDef": ["ARate", "AccidentalDeathBenefitsPremium",..],"FromDataDef": ["../../../../../../../../../../BOPCyberIncidentExclusionTable/BOPCyberIncidentExclusion[1]/Factor",..],"FromParam": ["AgreedValParam",..]}

rul:DateAdd = { "ToDataDef": ["ExpDate"], .. },

rul:DatePart = { "UnitType": ["Days", "Months", "Years"] }

rul:Else = {}

rul:FirstNonNull = { "Type": ["boolean", "dateTime", "decimal", "integer", "string"], "ToDataDef": ["ErcFormName", "ErcFormNumber", "NumEmployees"] }

rul:FirstValue = { "Type": ["dateTime", "decimal", ..], "FromConstant": ["", "0", ..], "FromDataDef": ["../../../../../../../../../../../../../CyberIncidentExclusionCOLExcptnsFactorBGI", ..],"Order": ["DataDefInputParamConstant"], "FromParam": ["firstHighestMedicalExpenseBenefitsCoverageLimitParam",..]}

rul:ForEach = { "AtDataDef": ["../../../../../../../../../../CommercialPropertyFloodCovEndtBlanketRatingTable/CommercialPropertyFloodCovEndtBlanketRating",..],"AtInputDataDef": ["../../../../../../../../../../../../CommercialPropertyBlanketRatingTable/CommercialPropertyBlanketRating",..]}

rul:Locate = { "AtOutputDataDef": ["BOPALChangesTable/BOPALChanges[1]",..], "OutputAction": ["Append"], "AtDataDef": ["BOPAddlLiabExposuresCoverage",..], "AtInputDataDef": ["../../../../BOPStructureLiabMedExpensesBldgCoverage",..], "Type": ["none"] }

rul:Lookup = { "Type": ["dateTime", "decimal", "integer", "string"], "MatrixCol": ["ARateClassCodeIndicator",...], "MatrixDef": ["ACVFactorDef",..], "MatrixFromConstant": ["ACVFactor",..], "ResultMode": ["FirstResult", ..], "ToDataDef": ["AnnualAggregateFactor",..]}

rul:Product = { "ToDataDef": ["AccidentalDeathBenefitsBasePremium",..], "DecimalPlaces": ["0",..]}

rul:Rule = { "Name": ["ACondominiumClassificationMustBeSelectedWhenCondominiumAssociationIsYes",..], "DataDefGroup": ["BOP",..], "MetadataCodes": ["RuleTypeCountrywide",..], "Type": ["dateTime",..]}

rul:RunRule = { "Type": ["dateTime",..], "FileName": ["BOPALChangesRules",..], "Rule": ["ACondominiumClassificationMustBeSelectedWhenCondominiumAssociationIsYes",..], "ProjectName": ["AU CW 20191201 V01",..], "ClearCache": ["true"], "ToDataDef": ["ARateClassCodeIndicator",..])

rul:Sequence = {}

rul:Sum = { "ToDataDef": ["AccountsReceivablePremium",..]}

rul:Test = {}

rul:Then = {}

rul:Value = { "Type": ["boolean", ..], "FromDataDef": ["../../../../../../../../../../BOPCyberIncidentExclusionTable/BOPCyberIncidentExclusion[1]/Factor",..],"FromParam": ["ARDEffDate",..], "AllowNullReturn": ["true"], "ToDataDef": ["AddedPersonalInjuryProtectionBaseLossCost",..]}

The Combination Tags

Some commonly used combination tags are :

rul:If = ["rul:Test", "rul:Then", "rul:Else"]

rul:Rule = ["rul:Sequence", "rul:If", "rul:Lookup", "rul:Sum", "rul:Param", "rul:FirstNonNull", "rul:RunRule", "rul:FirstValue", "rul:Convert", "rul:Round", "rul:Constant", "rul:WithArgs", "rul:Locate"]

rul:Test = ["rul:And", "rul:Equal", "rul:LessThanOrEqual", "rul:IsNull", "rul:NotEqual", "rul:NotExist", "rul:GreaterThan", "rul:Or", "rul:Exist", "rul:IsNotNull", "rul:GreaterThanOrEqual", "rul:LessThan"]

rul:RunRule = ["rul:Arg"]

rul:DateAdd = ["rul:Value", "rul:Constant"]

rul:DatePart = ["rul:FirstValue", "rul:Value", "rul:DateCreate"]

rul:ForEach = ["rul:RunRule", "rul:Locate", "rul:If", "rul:ForEach", "rul:FirstValue", "rul:Constant", "rul:WithArgs", "rul:Choose", "rul:Product", "rul:Exist", "rul:Break", "rul:Convert", "rul:Sum", "rul:FromList", "rul:Round", "rul:Sequence"]

rul:Sum = ["rul:Convert", "rul:RunRule", "rul:Locate", "rul:Value", "rul:Constant", "rul:ForEach", "rul:FirstValue", "rul:Product", "rul:If", "rul:Subtract", "rul:Sum", "rul:Divide", "rul:DatePart", "rul:Count", "rul:Break", "rul:WithArgs", "rul:DateDifference", "rul:Choose", "rul:Round", "rul:Truncate"]

rul:Product = ["rul:FirstValue", "rul:Divide", "rul:Sum", "rul:Product", "rul:Subtract", "rul:Truncate", "rul:Round", "rul:Constant", "rul:Convert", "rul:Value", "rul:RunRule", "rul:Minus", "rul:If", "rul:WithArgs", "rul:Choose", "rul:Lookup"]

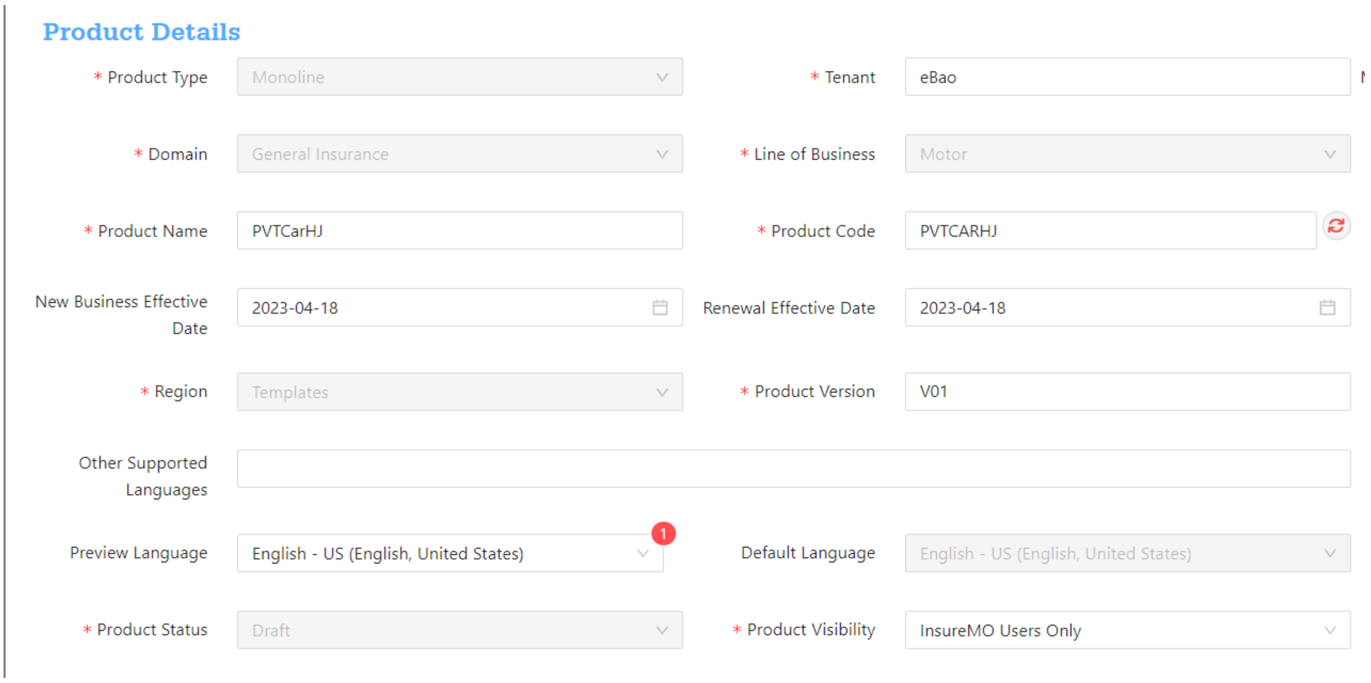
rul:MetaData = ["rul:MetaDataCode"]

Examples

To create a new product,

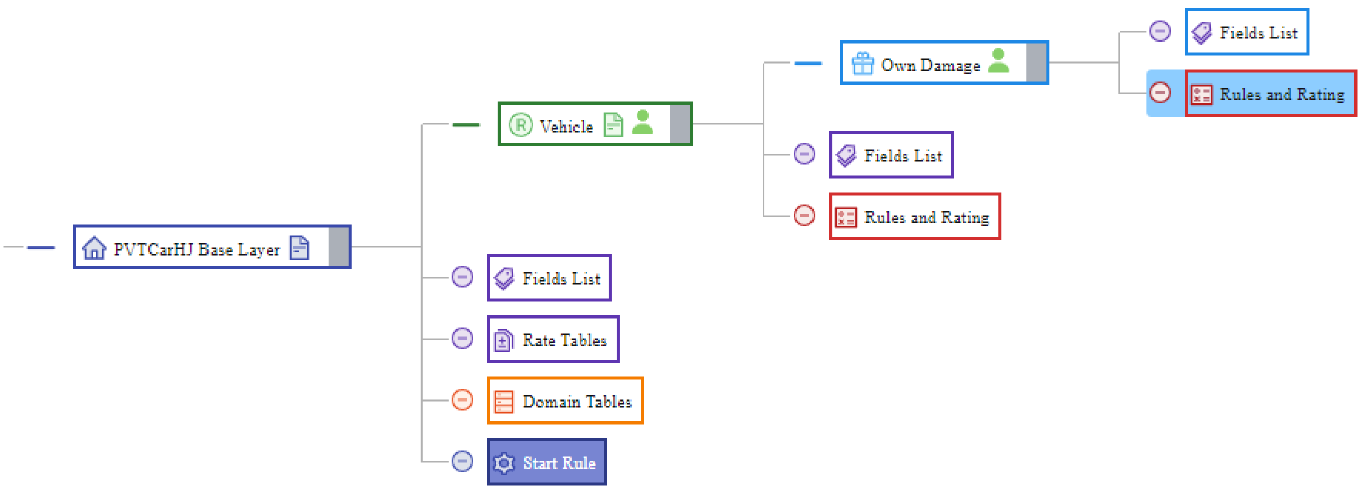
- **Policy - Risk - Coverage**

To calculate *Premium* = *SellingPrice* * *BaseRate* at coverage level.



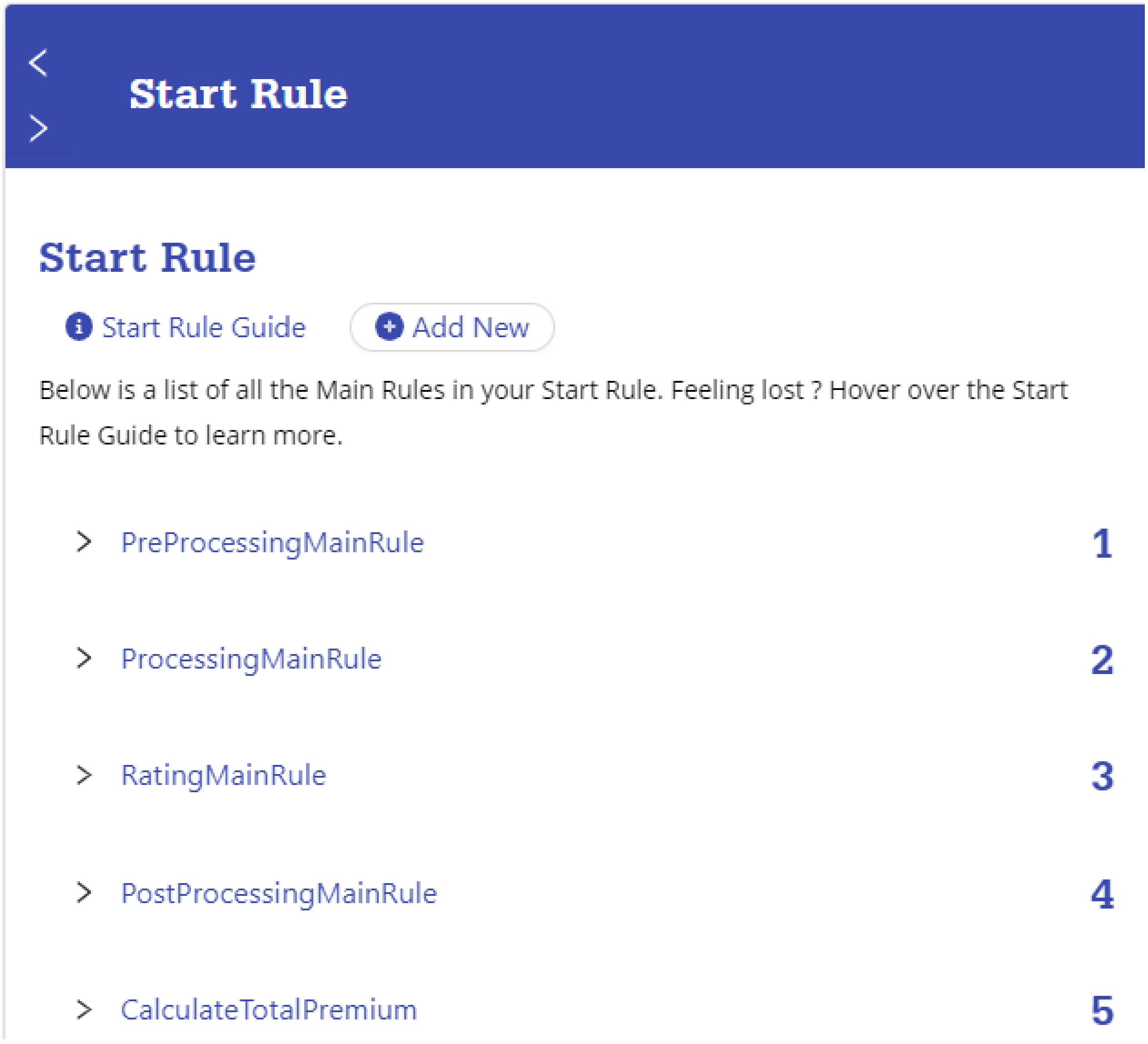
Product Details

* Product Type	Monoline	* Tenant	eBao
* Domain	General Insurance	* Line of Business	Motor
* Product Name	PVTCarHJ	* Product Code	PVTCARHJ
New Business Effective Date	2023-04-18	Renewal Effective Date	2023-04-18
* Region	Templates	* Product Version	V01
Other Supported Languages			
Preview Language	English - US (English, United States)	Default Language	English - US (English, United States)
* Product Status	Draft	* Product Visibility	InsureMO Users Only



In Fields List, add Data Labels required for calculation. At every level, add *Premium, PolicyTermPremium* Data Labels into Fields List.

In Start Rule, with *Add New* button, add rule *CalculateTotalPremium*. In this rule, we are making call to its child node. Here, it is Risk node.



Start Rule

Start Rule

Start Rule Guide Add New

Below is a list of all the Main Rules in your Start Rule. Feeling lost ? Hover over the Start Rule Guide to learn more.

PreProcessingMainRule 1

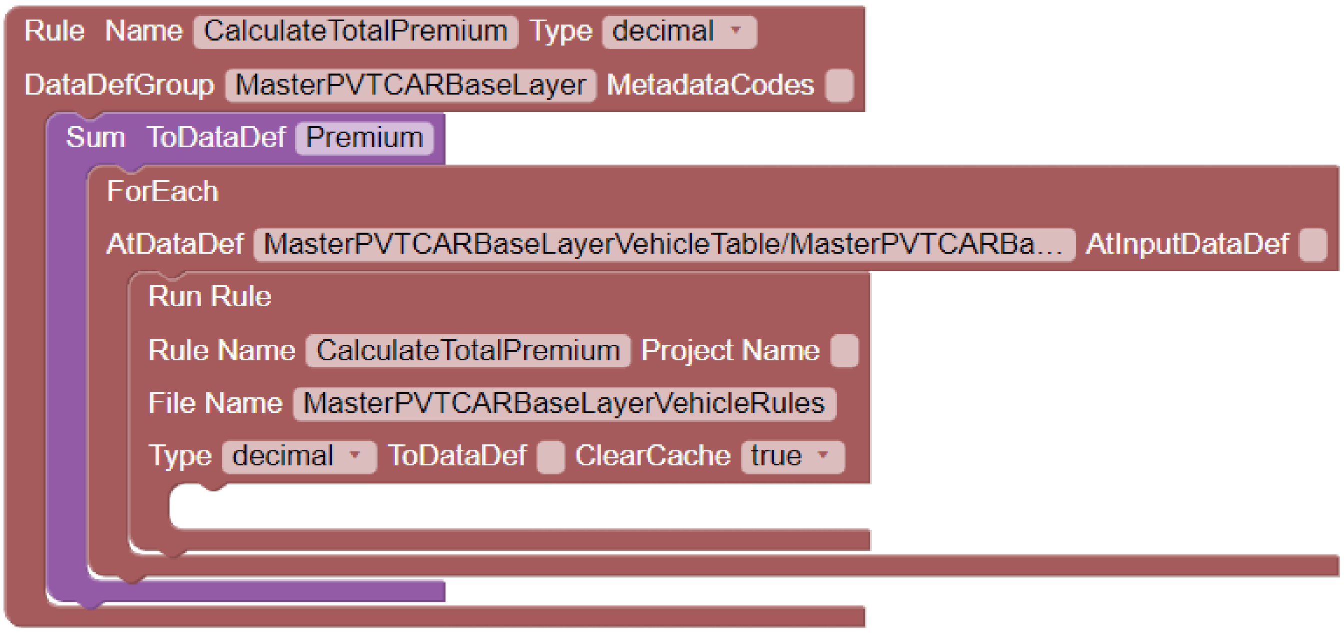
ProcessingMainRule 2

RatingMainRule 3

PostProcessingMainRule 4

CalculateTotalPremium 5

In *ForEach* block, for attribute *AtDataDef*, we write elementCodeTable/elementCode of the child node. Here, it is *MasterPVTCARBaseLayerVehicleTable/MasterPVTCARBaseLayerVehicle*. Attribute *filename* will have elementCodeRules. Here, it is *MasterPVTCARBaseLayerVehicleRules*. Based on return type of rule, please select value for attribute *Type*. Here, it is *decimal*.



Rule Name CalculateTotalPremium Type decimal

DataDefGroup MasterPVTCARBaseLayer MetadataCodes

Sum ToDataDef Premium

ForEach

AtDataDef MasterPVTCARBaseLayerVehicleTable/MasterPVTCARBa... AtInputDataDef

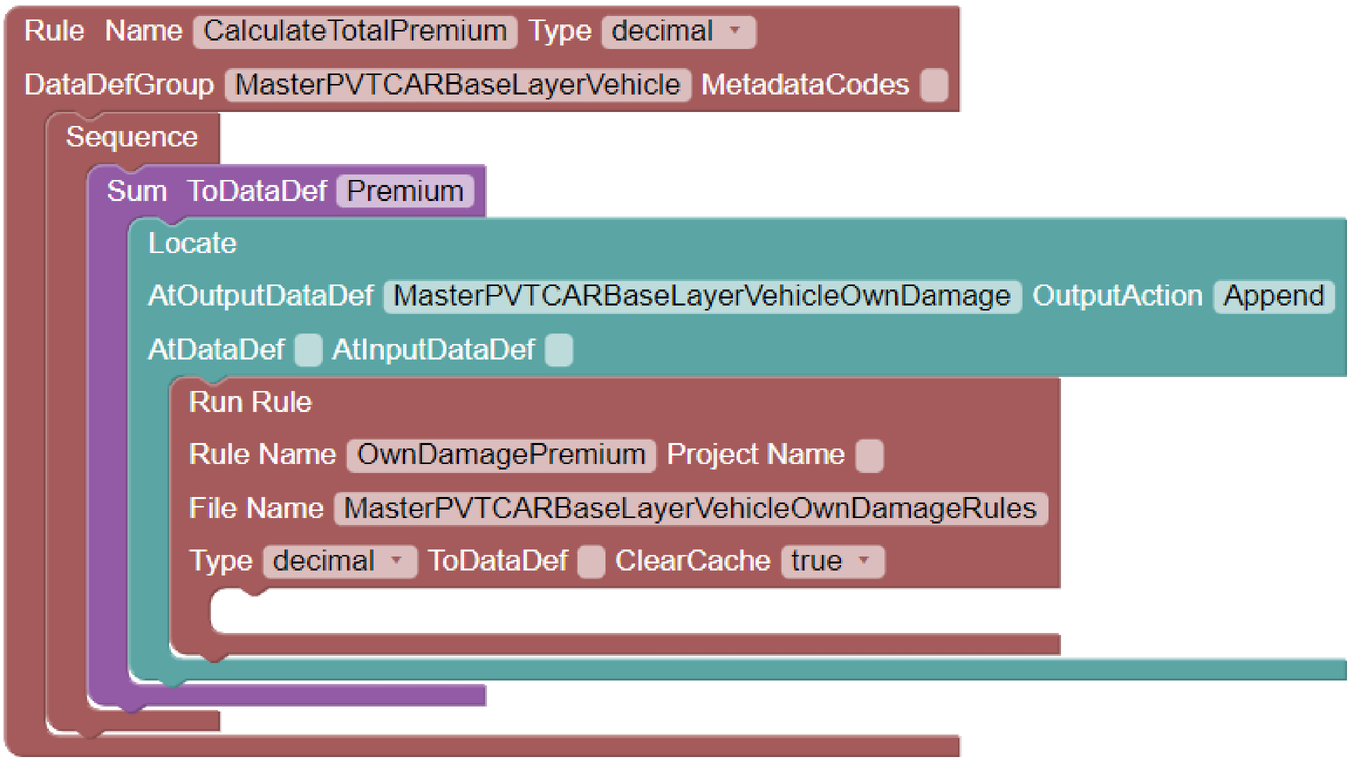
Run Rule

Rule Name CalculateTotalPremium Project Name

File Name MasterPVTCARBaseLayerVehicleRules

Type decimal ToDataDef ClearCache true

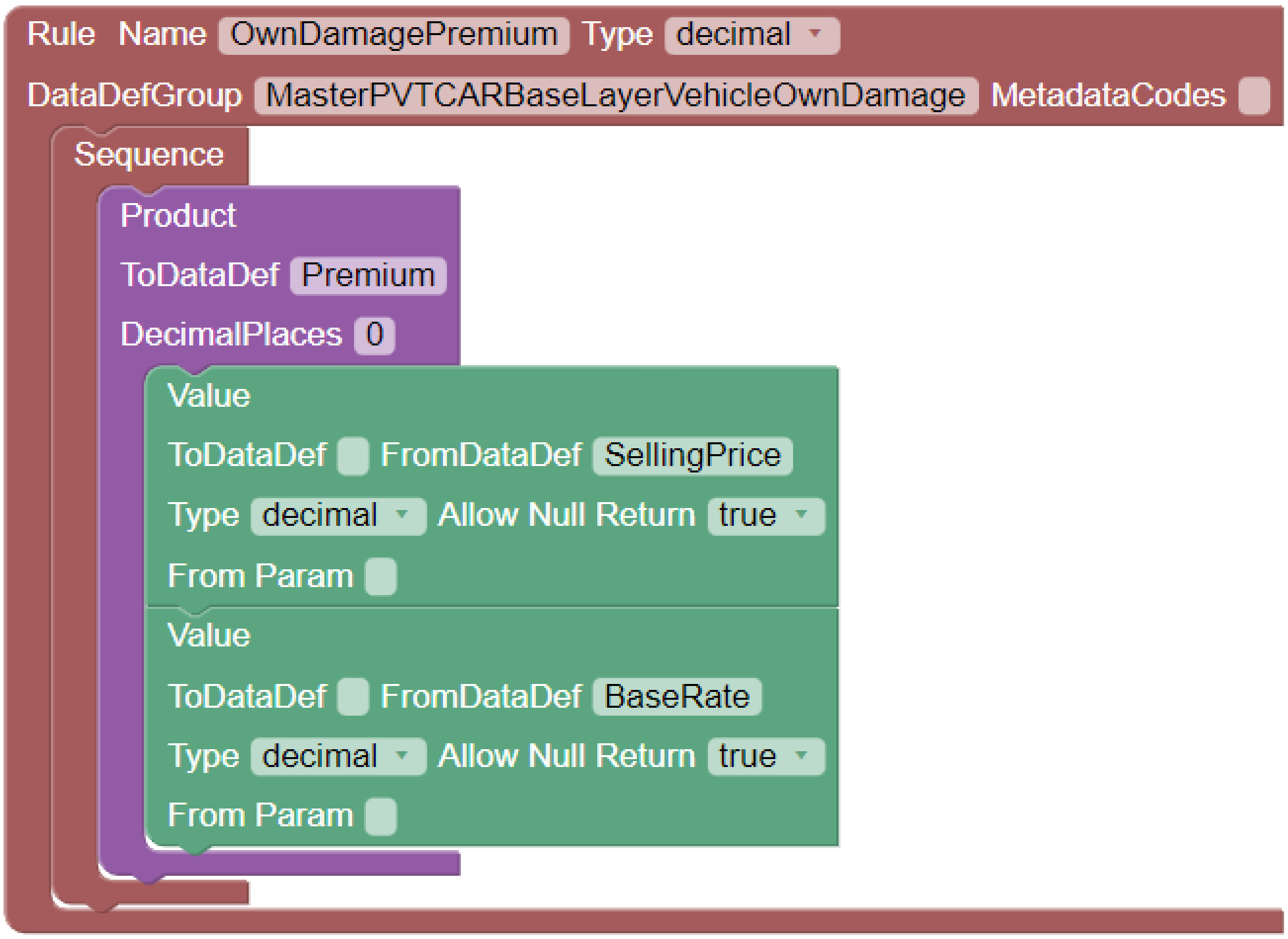
At Risk level, add rule *CalculateTotalPremium*. In this rule, we are making call to its child node. Here, it is Coverage. For Coverage we do not follow elementCodeTable/elementCode structure. It is beacause Coverage falls at same level.



```
Rule  Name CalculateTotalPremium Type decimal
DataDefGroup MasterPVTCARBaseLayerVehicle MetadataCodes
  Sequence
    Sum  ToDataDef Premium
      Locate
      AtOutputDataDef MasterPVTCARBaseLayerVehicleOwnDamage OutputAction Append
      AtDataDef  AtInputDataDef
        Run Rule
        Rule Name OwnDamagePremium Project Name
        File Name MasterPVTCARBaseLayerVehicleOwnDamageRules
        Type decimal ToDataDef  ClearCache true
```

We use Locate block. We pass elementCode of child node in attribute *AtOutputDataDef* with *OutputAction* as Append. It is beacuse we want to make call to this rule and append its result as output. Here, elementcode is *MasterPVTCARBaseLayerVehicleOwnDamage*. Attribute *filename* will have elementCodeRules. Here, it is *MasterPVTCARBaseLayerVehicleOwnDamageRules*. Based on return type of rule, please select value for attribute *Type*. Here, it is *decimal*.

At Coverage, we add rule to calculate *Premium = SellingPrice * BaseRate*



```
Rule  Name OwnDamagePremium Type decimal
DataDefGroup MasterPVTCARBaseLayerVehicleOwnDamage MetadataCodes
  Sequence
    Product
    ToDataDef Premium
    DecimalPlaces 0
      Value
      ToDataDef  FromDataDef SellingPrice
      Type decimal Allow Null Return true
      From Param
      Value
      ToDataDef  FromDataDef BaseRate
      Type decimal Allow Null Return true
      From Param
```