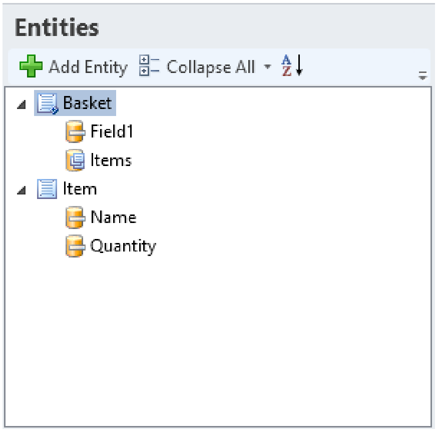
InRule Rule Authoring Iteration Patterns

Rule iteration, the repeated application of rules within a process, can be accomplished a number of different ways within a rule application. This document is not an exhaustive list but is intended to outline common approaches used for rule iteration. Namely:

1. Automatic rule sets written within a child entity context.
2. Parent Activation of Automatic rule sets written within a child entity context.
3. Using the Execute Member Rule Set action.
4. Using the While Loop element.

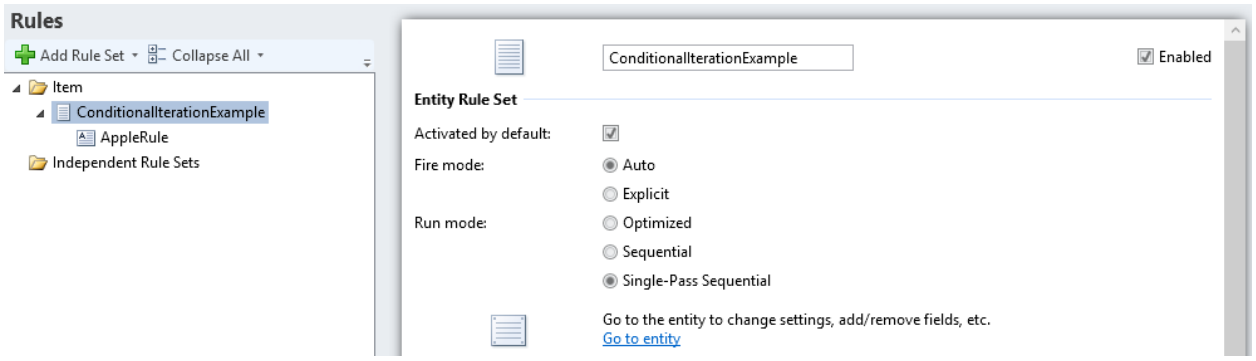
The included rule applications share the same entity schema and can be tested from within irVerify using the provided "SampleEntityState.xml" file.

Automatic Rule Sets Written Within a Child Entity Context



The rule application "1.AutoRuleSetInChildContext" contains two entities: "Basket" and "Item". The Item entity is related to the Basket entity via the "Items" entity collection.

This rule application will demonstrate creating a rule which interrogates each Item.Name in the Basket.Items collection for a value of "Apple". This will be accomplished by creating a rule set in the context of Item:



ConditionalIterationExample [✓ Enabled]

Entity Rule Set

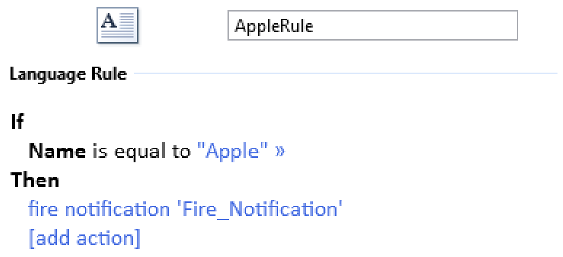
Activated by default: ✓

Fire mode: [✓ Auto] Explicit

Run mode: Optimized Sequential [✓ Single-Pass Sequential]

Go to the entity to change settings, add/remove fields, etc.
Go to entity

Setting the fire mode to "Auto" ensures that this rule set will execute for each instance of Item found within the rule application. The run mode setting of "Single-Pass Sequential" ensures that the rules within this rule set will only run once within each instance.


AppleRule

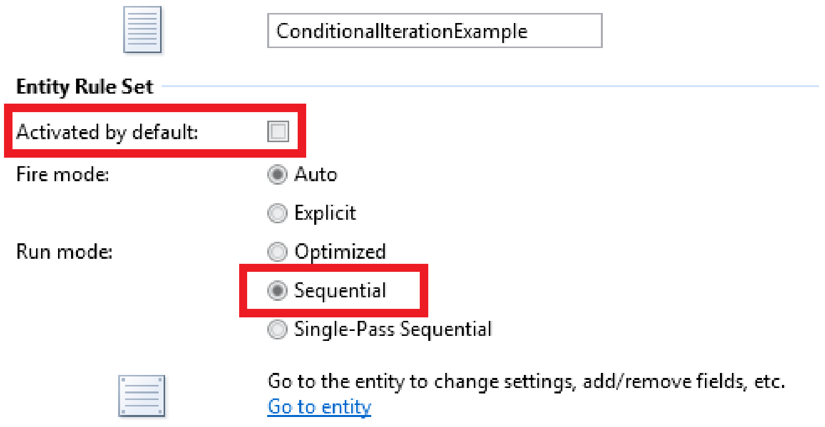
Language Rule

If
Name is equal to "Apple" »
Then
fire notification 'Fire_Notification'
[add action]

The “AppleRule” then interrogates the Item.Name field, and if it is equal the value “Apple” will execute a fire notification action.

Parent Activation of Automatic Rule Sets Written Within a Child Entity Context

Similar to the example above, the rule application “2.AutoRuleSetInChildContextWithActivation” uses an auto rule set written in the context of Item. This pattern is slightly different in that the rule set is configured to “Sequential”. This allows rule engine dependency tracking for the rules within this rule set, allowing the rules to be re-executed by the rules engine. The rule set is also deactivated by default:


ConditionalIterationExample

Entity Rule Set

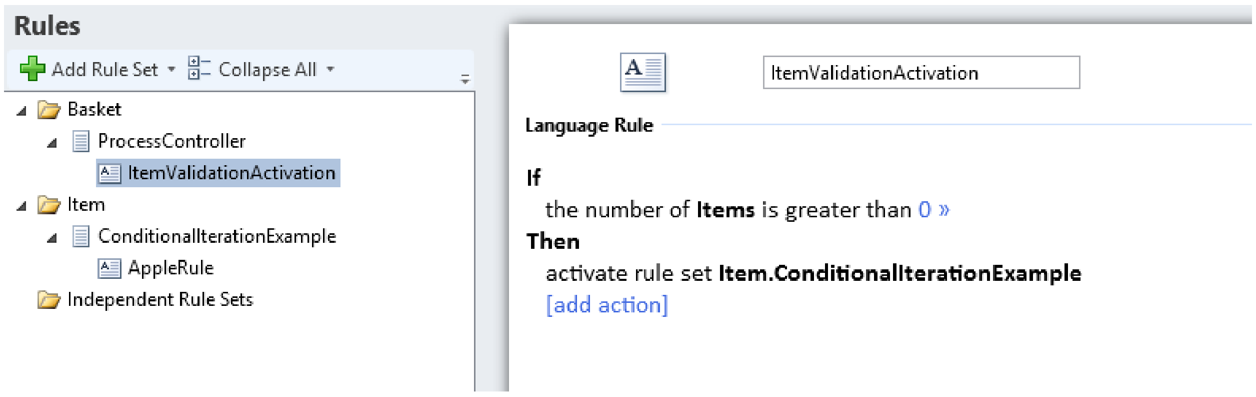
Activated by default: ☐

Fire mode: [✓ Auto] Explicit

Run mode: Optimized [✓ Sequential] Single-Pass Sequential

Go to the entity to change settings, add/remove fields, etc.
Go to entity

Deactivating a rule set allows a rule author to control when the rules within the rule set will be initially fired. This can be done using an Activate Rule Set action. An Activate Rule Set action can be used either from within any rule application context. In the provided example, the rule set is activated from within the parent context of “Basket”:



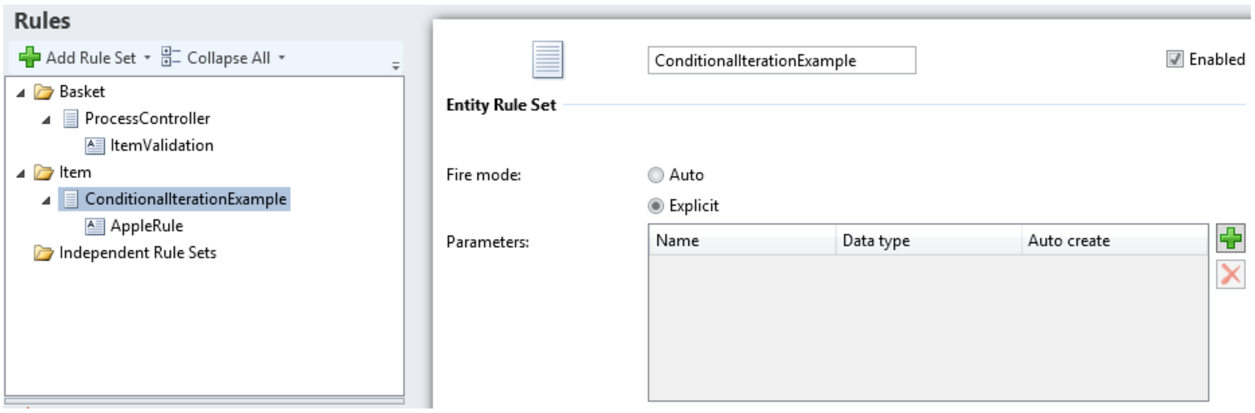
ItemValidationActivation

Language Rule

If
the number of **Items** is greater than 0 »
Then
activate rule set **Item.ConditionalIterationExample**
[add action]

Execute Member Rule Set

Similar to the previously demonstrated activation pattern, the execute member rule set pattern allows a rule author to explicitly specify when the rules within a rule set need to be executed. It differs from the activation pattern in that an explicit rule set has an implied run mode of “Single-Pass Sequential”, disabling dependency tracking for the rules within that rule set (and thus ensuring that the rules contained will execute exactly once per instance):



ConditionalIterationExample ☑ Enabled

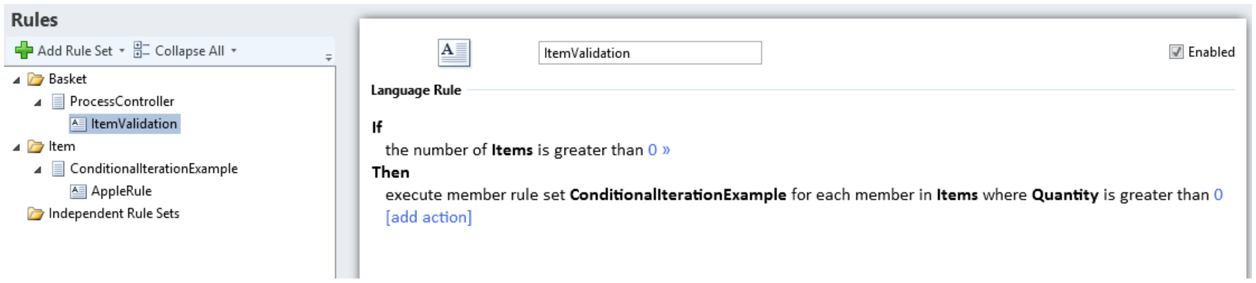
Entity Rule Set

Fire mode: Auto [✓ Explicit]

Parameters:

Name	Data type	Auto create

Similar to rule set activation, executing an explicit rule set can be called from within rules by using the Execute Rule Set action (when within the current context) or by using the Execute Member Rule Set action (when calling an explicit rule set for rule sets in a child context). In the provided example, we use an Execute Member Rule Set action in the context of Basket. It is also worth noting that the Execute Member Rule Set action can use a filtering expression, further qualifying which members within a collection should execute the target rule set. In the provided example, the rule set “ConditionalIterationExample” will only execute when the Item.Quantity is greater than 0:



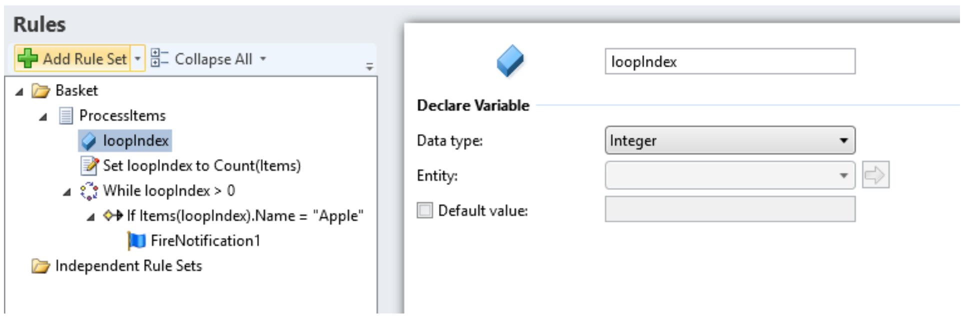
ItemValidation ☑ Enabled

Language Rule

If
the number of **Items** is greater than 0 »
Then
execute member rule set **ConditionalIterationExample** for each member in **Items** where **Quantity** is greater than 0
[add action]

While Loops

The While syntax element and the “While…Do” template are the most explicit forms of iteration within a rule application. The provided example “4.WhileLoopExample” processes Basket.Items as was done in the previous examples. The while loop element will be contained in a rule set in the context of Basket and the loop will require the use of a rule set variable “loopIndex” and an loopIndex assignment action to control the looping behavior:



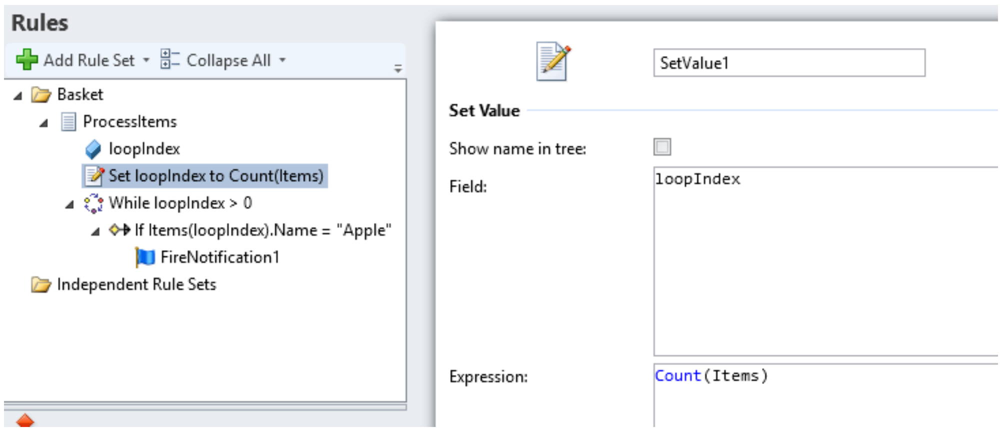
loopIndex

Declare Variable

Data type: Integer

Entity:

Default value:



SetValue1

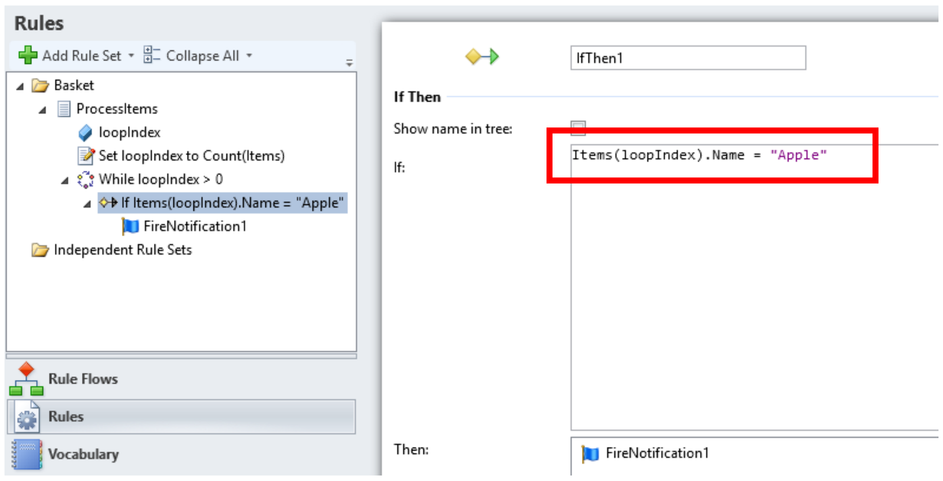
Set Value

Show name in tree:

Field: loopIndex

Expression: Count(Items)

The while loop will be set to run while the loopIndex has more than 0 items to process. Within the “Do” portion of the loop, we have added an IfThen template to conditionally look for an Item.Name of “Apple” as we have done in previous exercises. It is important to note that since this construct is in the context of “Basket” we must use the loopIndex to reference the target Item member we are currently interrogating:



IfThen1

If Then

Show name in tree:

If: `Items(loopIndex).Name = "Apple"`

Then: FireNotification1