GRS Best Practice Guide

Example 1—

12/19/2025

Example 1—Condition & Action

Contents

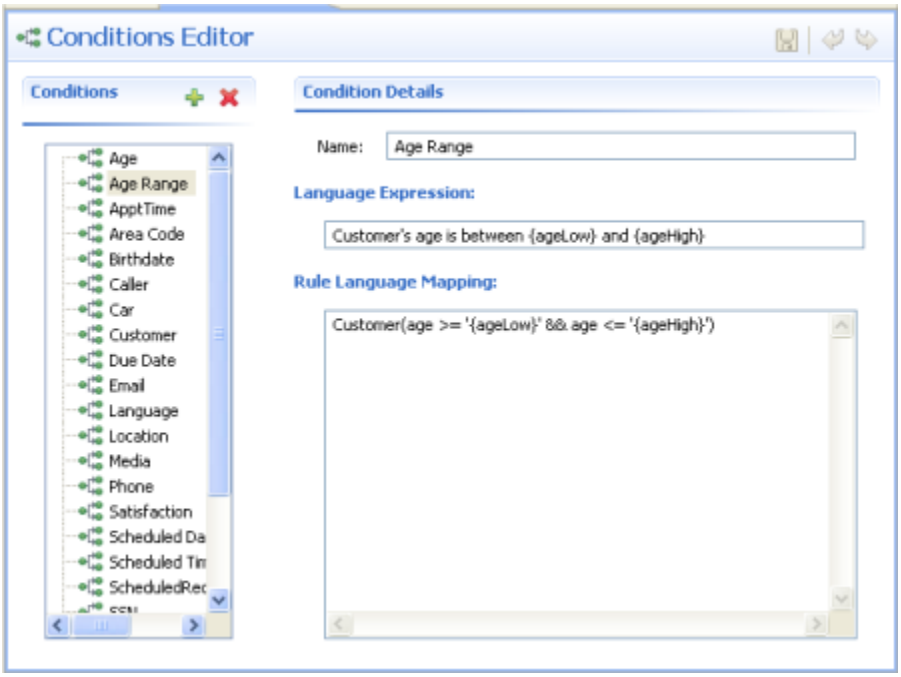
Age Range Condition

If a customer's age is within a specific range, a specific Agent Group will be targeted. In this scenario, the Condition is whether the customer's age falls within the range. In the Genesys Rules Development Tool, the conditions would be configured as follows:

```
Name: Age Range
Language Expression: Customer's age is between "{ageLow}" and "{ageHigh}"
Rule Language Mapping: Customer(age >= '{ageLow}' && age <= '{ageHigh}')
```

Do not use the word 'end' in rule language expressions. This causes rule parsing errors.

The figure below shows how this condition would appear in the Genesys Rules Development Tool.



Age Range Condition

Mapping Multiple Instances of a Rule Parameter to a Single Parameter Definition

At the point of creating parameters, instead of creating the "{ageLow}" and "{ageHigh}" parameters, the rule template developer could instead create a single "{age}" parameter and use the underscore notation shown in the example below to create indices of it for scenarios in which multiple instances of parameter with the same type (age) are required (most commonly used with ranges). For example:

```
"{age_1}", "{age_2}"...."{age_n}"
```

These will become editable fields. This feature is most typically used for defining ranges more efficiently.

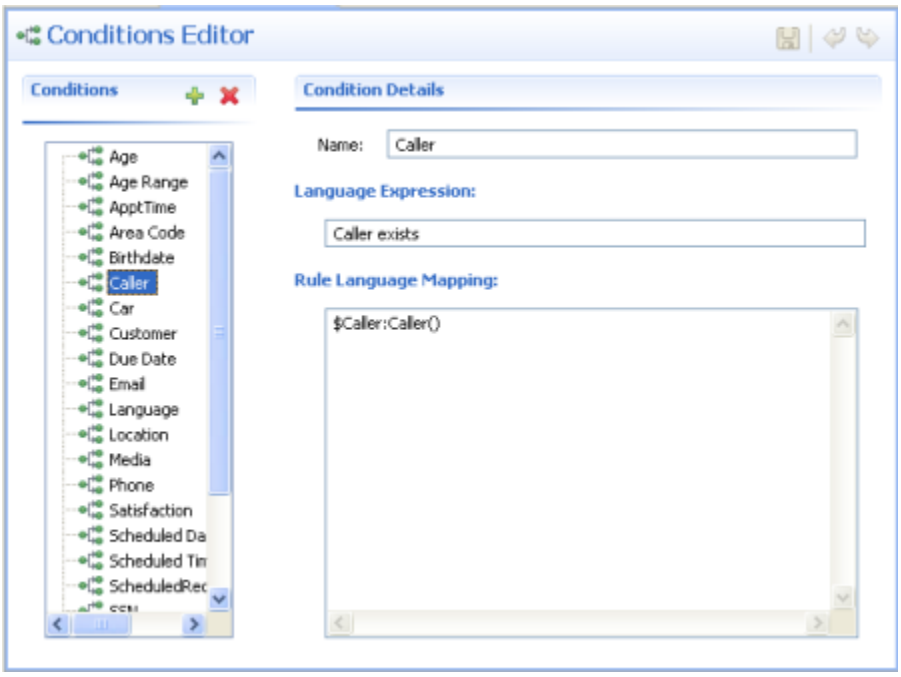
Caller Condition

In addition to testing that the Caller exists, the next condition also creates the `$Caller` variable which is used by actions to modify the Caller fact. The modified Caller will be returned in the results of the evaluation request.

You cannot create a variable more than once within a rule, and you cannot use variables in actions if the variables have not been defined in the condition.

```
Name: Caller
Language Expression:  Caller exists
Rule Language Mapping: $Caller:Caller
```

The figure below shows how this condition would appear in the Genesys Rules Development Tool.



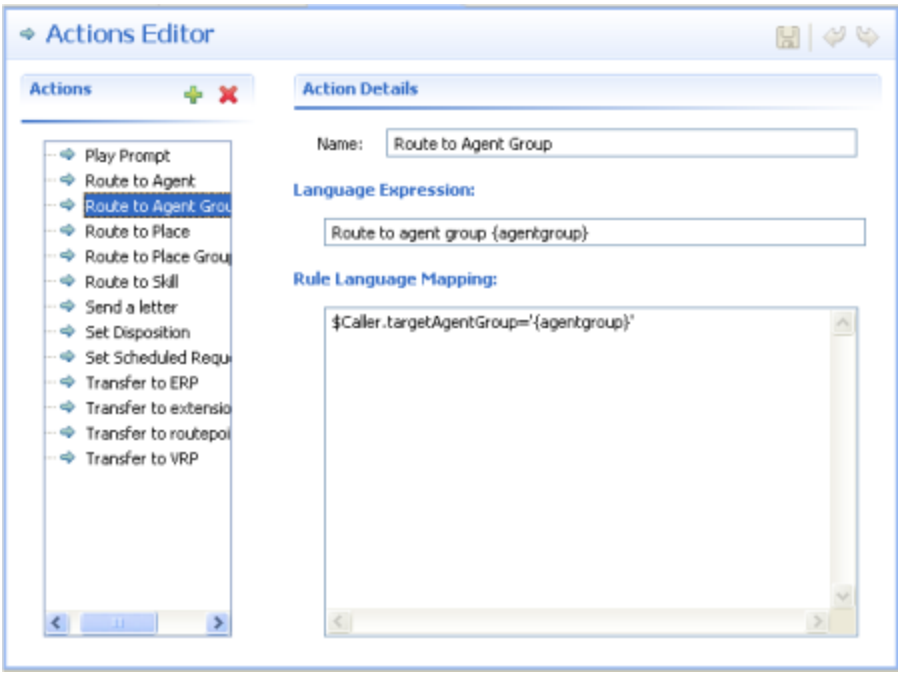
Caller Condition

Target Agent Group Action

The action would be configured as follows:

```
Name: Route to Agent Group
Language Expression: Route to agent group {agentGroup}
Rule Language Mapping: $Caller.targetAgentGroup='{agentgroup}'
```

The figure below shows how this action would appear in the Genesys Rules Development Tool.

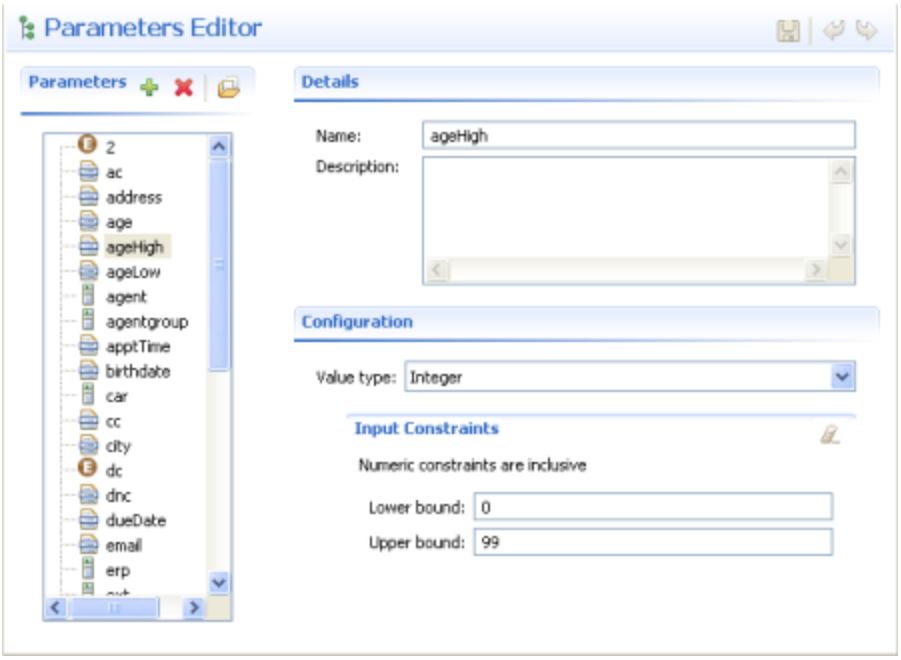


Target Agent Group

The condition in this example has two parameters:

- `"{ageLow}"`
- `"{ageHigh}"`

The action has the `"{agentGroup}"` parameter. Parameters are also configured in the Genesys Rules Development Tool. The Parameters Editor screenshot shows a sample `"{ageHigh}"` parameter. Refer to the Genesys Rules Development Tool Help for more details about how to configure parameters.



Parameters Editor Screen

The way the preceding example would work is as follows:

1. The rule developer creates a fact model (or the fact model could be included as part of a rule template that comes out of the box with a particular Genesys solution). The fact model describes the properties of the `Customer fact` and the `Caller` fact. In this case we can see that the `Customer` fact has a property called age (probably an integer) and the `Caller` fact has a property called `targetAgentGroup` (most likely a string).
2. The rule developer creates the `ageLow` and `ageHigh` parameters, which will become editable fields that the business user will fill in when they are authoring a business rule that uses this rule template. These parameters would be of type `Input Value` where the `Value Type` would likely be integer. The rule developer optionally can constrain the possible values that the business user will be able to enter by entering a Lower Bound and/or an Upper Bound.
3. The rule developer also creates the `agentGroup` parameter, which will likely be a selectable list whereby the business user would be presented with a drop-down list of values that are pulled from Genesys Configuration Server or from an external data source. The behavior of this parameter depends on the parameter type that is selected by the rule developer.
4. The rule developer creates a rule action and rule condition as previously described. The action and condition include rule language mappings that instruct the Rules Engine as to which facts to use or update based on information that is passed into the Rules Engine as part (of the rule evaluation request coming from a client application such as an SCXML application).
5. The rule developer publishes the rule template to the Rules Repository.
6. The rules author uses this rule template to create one or more business rules that utilize the conditions and actions in the combinations that are required to describe the business logic that the rules author wants to enforce. In this case, the previously described conditions and action above likely would be used together in a single rule, but the conditions and action could also be combined with other available conditions and actions to create different business policies.
7. The rules author deploys the rule package to the Rules Engine application server or cluster.

8. A client application such as a VXML or SCXML application invokes the Rules Engine and specifies the rule package to be evaluated. The request to the Rules Engine will include the input and output parameters for the fact model. In this example, it would have to include the age property of the Customer fact. This age might have been collected through GVP or extracted from a customer database prior the Rules Engine being called. Based on the value of the `Customer.age` fact property that is passed into the Rules Engine as part of the rules evaluation request, the Rules Engine will evaluate a particular set of the rules that have been deployed. In this example, it will evaluate whether `Customer.age` falls between the lower and upper boundaries that the rules author specified in the rule.
9. If the rule evaluates as true by the Rules Engine, the `targetAgentGroup` property of the `Caller` fact will be updated with the name of the Agent Group that was selected by the business rules author when the rule was written. The value of the `Caller.targetAgentGroup` property will be passed back to the client application for further processing. In this example, perhaps the value of `Caller.targetAgentGroup` will be mapped to a Composer application variable which will then be passed into the Target block to ask the Genesys Universal Routing Server to target that Agent Group.