This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

Genesys Rules System Deployment Guide

Rule Evaluation

5/9/2025

Rule Evaluation

We want to call the GRE from our client (VXML) application. For the rule to be evaluated properly, we will have to populate the fact properties of the `_GRS_Environment` and `Customer` facts correctly.

To test this rule evaluation, you can use a REST client, such as the free Firefox REST Client add-on, or you can test the rule by using Composer's Business Rule Block, which has a built in Test feature that provides sample values to the rule and evaluates the results.

In most cases, you will use Genesys Composer to build applications that will invoke the GRE. However, to simplify rule testing, it might be more convenient to use a REST client in the manner that is described here.

The request to the Rules Engine will be a POST request. The URL we will use to make the POST request will be constructed as follows:

```
http://[server:port]/genesys-rules-engine/knowledgebase/[package]
```

where: server is the IP address or host name of the application server on which the rules engine is running port is the listening port of the application server. For example, 8080 is the default Tomcat port. package is the name of the rule package to evaluate. In this example it is `my.test`.

So, the URL might look like this:

```
http://myserver:8080/genesys-rules-engine/knowledgebase/my.test
```

We have to populate the request body with the request, in XML format. In the request body we specify the two fact classes, both of which are prefixed with the package name; for example, `my.test._GRS_Environment` and `my.test.Customer`, respectively.

For the `_GRS_Environment` fact, we have to provide values for the fact properties phase and `businessContext_Level*`. Note that your request can include multiple values for the `businessContext_Level*` fact property, depending on the node(s) of the business structure at which you want the Rules Engine to evaluate rules.

In our case, let us assume that in this request, we want the Rules Engine to evaluate the rules at both the Finance Department level and the Accounts Payable level. In this case, in our request we will populate fact properties that specify both of these levels (`businessContext_Level1` and `businessContext_Level2`). Alternatively, if we omitted `businessContext_Level2` from the request, we could ask the Rules Engine to evaluate only the rules at the Finance Department level, which is

businessContext_Level1.

Note also that if you had any rules configured at the “global” level (which are configured for the rule package itself by selecting the name of the package in the navigation tree, and then selecting the Rules tab), they will always be evaluated for every request, without having to specify anything explicitly in the _GRS_Environment fact property.

The other _GRS_Environment fact property that we must populate in the request is the phase. In our example, all rules were written for the segmentation phase.

For the Customer fact, we must provide values for the fact properties segment and order. We can provide whatever values we want, in order to test the results of the rule evaluation. Note that the value that you provide for the segment fact property is case-sensitive, as is the value for the phase fact property. See the description of the customerSegment enumeration in Rule Template.

The following is an example of the request body:

```
<knowledgebase-request>
        <inOutFacts>

                <named-fact>
                        <id>env</id>
                        <fact class=”my.test._GRS_Environment”>
                                <phase>segmentation</phase>
                                <businessContext_Level1>Finance
Department</businessContext_Level1>
                                <businessContext_Level2>Accounts
Payable</businessContext_Level2>
                        </fact>
                </named-fact>

                <named-fact>
                        <id>customer</id>
                        <fact class=”my.test.Customer”>
                                <segment>gold</segment>
                                <order>6345.32</order>
                        </fact>
                </named-fact>

        </inOutFacts>
</knowledgebase-request>
```

Based on our rule configuration, we would expect that the Rule Engine would return a value of 1 for the offer property of the Caller fact, indicating that under these conditions (customer is Gold and the customer’s order value is greater than $5,000.00), we want to offer them a special promotion. This is because the parameter (specialOffer) that is being used in the rule action is a Boolean type. In this case, the response body will look like the following:

```
<knowledgebase-response>
        <inOutFacts>
```

```
                <named-fact>
                        <id>env</id>
                        <fact class="my.test._GRS_Environment">
                                <businessContext__Level2>Accounts
Payable</businessContext__Level2>
                                <businessContext__Level1>Finance
Department</businessContext__Level1>
                                <phase>segmentation</phase>
                        </fact>
                </named-fact>
                <named-fact>
                        <id>customer</id>
                        <fact class="my.test.Customer">
                                <order>6345.32</order>
                                <segment>gold</segment>
                                <offer>1</offer>
                        </fact>
                </named-fact>
        </inOutFacts>
        <executionResult>
                <rulesApplied>
                        <string>Row 1 DT-103 myRule</string>
                </rulesApplied>
        </executionResult>
</knowledgebase-response>
```

If you pass in values in your request that the Rules Engine will not evaluate to true, based on all of the rules that you have deployed, no value for the offer fact property will be returned in the result. For example, if you set the value of order to `2345.32`, the response body will look like the following:

```
<knowledgebase-response>
        <inOutFacts>
                <named-fact>
                        <id>env</id>
                        <fact class="my.test._GRS_Environment">
                                <businessContext__Level2>Accounts
Payable</businessContext__Level2>
                                <businessContext__Level1>Finance
Department</businessContext__Level1>
                                <phase>segmentation</phase>
                        </fact>
                </named-fact>
                <named-fact>
                        <id>customer</id>
                        <fact class="my.test.Customer">
                                <order>2345.32</order>
                                <segment>gold</segment>
                        </fact>
                </named-fact>
        </inOutFacts>
        <executionResult>
                <rulesApplied>
                </rulesApplied>
        </executionResult>
</knowledgebase-response>
```

Note that this is not the same as the value of offer being `0`. In this example, because all of the conditions in the rules were not met (evaluated as true by the Rules Engine), the action was not fired. Thus, offer has no value populated in the result. If you wanted the value of offer to be set to `0`, you would have to have a rule that included a rule action whereby the value of offer was unchecked

(remember that it is a Boolean parameter so it is either checked or unchecked by the rules author). If all of the conditions of such a rule were evaluated as true by the Rules Engine, the result would set offer to `0`.

If you want the response to include the offer fact property, with no value, it must be included in the request (even if no value is provided). In this case the `my.test.Customer` fact class would look like the following in the request:

```
<named-fact>
        <id>customer</id>
        <fact class=”my.test.Customer”>
                <segment>gold</segment>
                <order>2345.32</order>
                <offer></offer>
        </fact>
</named-fact>
```

And the response body would include the following section:

```
<named-fact>
        <id>customer</id>
        <fact class=”my.test.Customer”>
                <order>2345.32</order>
                <segment>gold</segment>
                <offer></offer>
        </fact>
</named-fact>
```

You can also try populating the request with values that will be relevant to the rule at the Accounts Payable level of the business structure—for example, `segment = bronze and order = 9345.33`. In this case, you should also see the value of order set to 1 in the response body.