Conversation Rules Templates Guide

Working with Composer's Business Rule Block

12/18/2025

Working with Composer's Business Rule Block

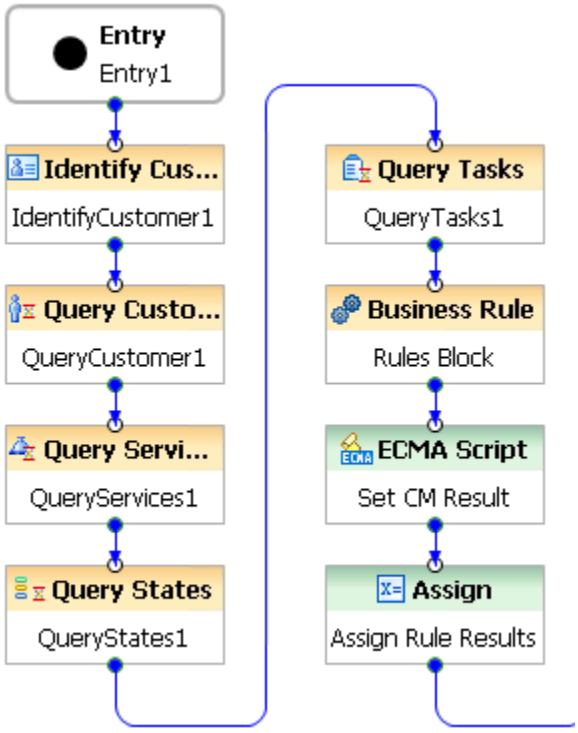
Contents

Summary

Once the Rule Packages (created from Rule Templates) that you want to work with are deployed to the Genesys Rules Engine, you can use the Business Rule block on the Server Side palette to create voice and routing applications that use business rules.

Use this block to have Composer query the Genesys Rules Authoring Tool (GRAT) for deployed packages. For the Rule Package that you specify, Composer will query the GRAT for the Facts associated with the Rule Package. You can then set values for the Facts, call the Genesys Rules Engine for evaluation, and save the results in a variable.

Simple Workflow



In this typical workflow:

1. The **Identify Customer** block is used to identify the customer based on certain search criteria, such as ANI.
2. The **Query Customer** block is used to pull out the customer profile data.
3. The **Query Services** block is called to pull any related services for this customer.

4. The **Query States** block is called to pull the states related to a particular service.
5. The **Query Task** block is called to pull the tasks associated with a particular service or state.
6. The **Business Rule** block may be placed anywhere in the workflow/callflow, assuming the data needed for the rules called by the Business Rule block have already been fetched.
7. The **Assign** block enables the workflow to access all the different decisions of the CM rule package. The decision(s) requested by the Business Rule block and made by the Rules Engine are returned to the workflow/callflow which carries them out. The GRS does not actually execute the decisions (e.g. update the customer profile, transfer to agent, an so on).

User Variables in the Workflow

In this example workflow, the following user variables have been defined to retrieve data necessary for, as well as demonstrate the various decisions made by, the CM rules.

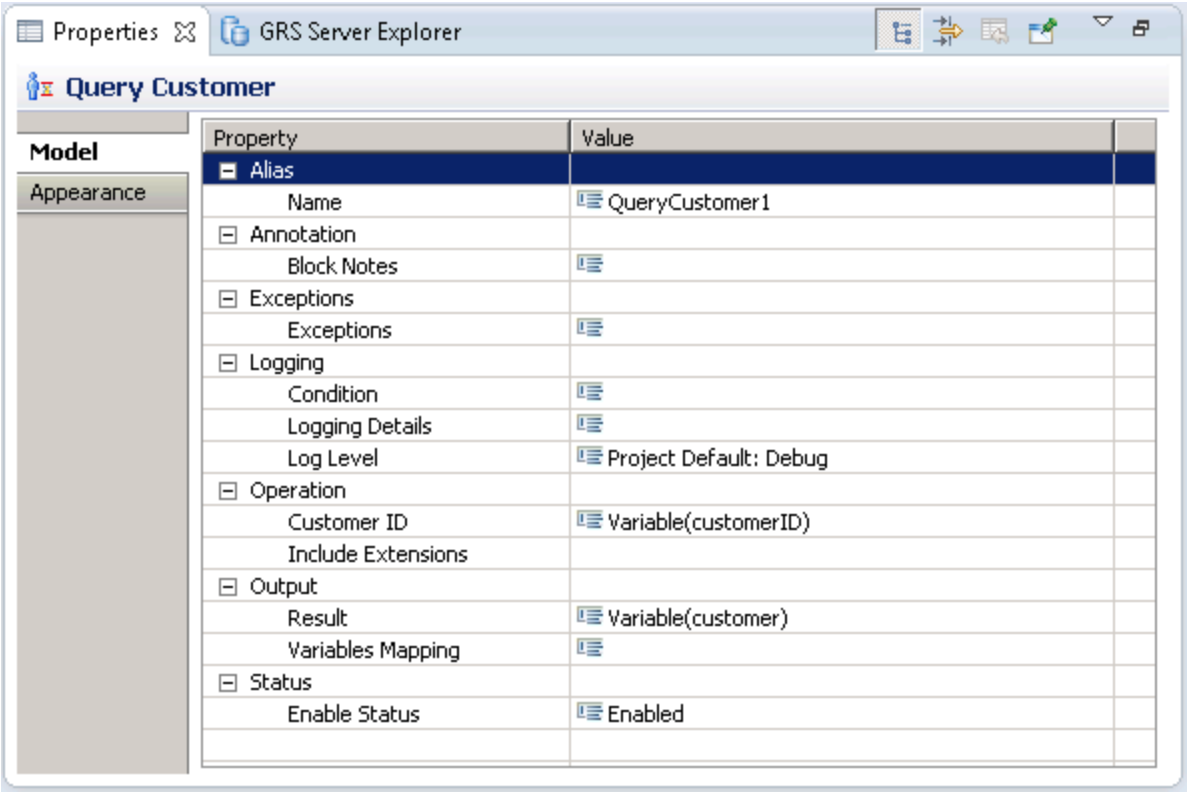
Variable Name	Default Value	Expected Type	Description
customer	undefined	JSON object	Customer profile data as returned by Query Customer block.
services	undefined	JSON array	Services of the customer as returned by Query Services block.
states	undefined	JSON array	States of a particular service as returned by Query States block.
tasks	undefined	JSON array	Tasks of a particular service or state as returned by Query Tasks block. (In Context Services, Tasks may be associated with service or state.)
contractEndDate	undefined	String	Contract end date timestamp string in the format of `2014-07-14T13:23:35.392Z`.
mediaType	undefined	String	Media type of the current interaction.
businessRulesResultObject	undefined	JSON object	Output of Business Rules block.
customerID	undefined	String	ID of the customer as returned by Identify Customer block.
serviceID	undefined	String	ID of a service of the customer to use in Query States and Query Tasks blocks.
stateID	undefined	String	ID of a state of a service

Variable Name	Default Value	Expected Type	Description
			of the customer to use in Query Task block. (Not used when querying tasks associated with service.)
cmResults	undefined	JSON object	Results from CM rules. This is extracted from businessRulesResultObject for easier access.
updatedFields	undefined	JSON object	CM Rules decision: Customer profile fields to update. Each key-value pair of this object correspond to a contact attribute. This is extracted from cmResults for easier access later in the workflow.
offerServiceResumption	false	Boolean	CM Rules decision: Whether to offer service resumption to customer. This is extracted from cmResults for easier access later in the workflow.
offerSurvey	false	Boolean	CM Rules decision: Whether to offer survey to customer. This is extracted from cmResults for easier access later in the workflow.
blockCommunication	false	Boolean	CM Rules decision: Whether to block further communication to customer. This is extracted from cmResults for easier access later in the workflow.
sendCommunication	undefined	String	CM Rules decision: Which media type to use for further communication with customer. This is extracted from cmResults for easier access later in the workflow.
requestedAgent	undefined	String	CM Rules decision:

Variable Name	Default Value	Expected Type	Description
			Which particular agent to route this customer to. This is extracted from cmResults for easier access later in the workflow.
requestedAgentGroup	undefined	String	CM Rules decision: Which agent group to route this customer to. This is extracted from cmResults for easier access later in the workflow.
requestedPlaceGroup	undefined	String	CM Rules decision: Which place group to route this customer to. This is extracted from cmResults for easier access later in the workflow.
requestedSkill	undefined	String	CM Rules decision: Which skill to route this customer to. This is extracted from cmResults for easier access later in the workflow.

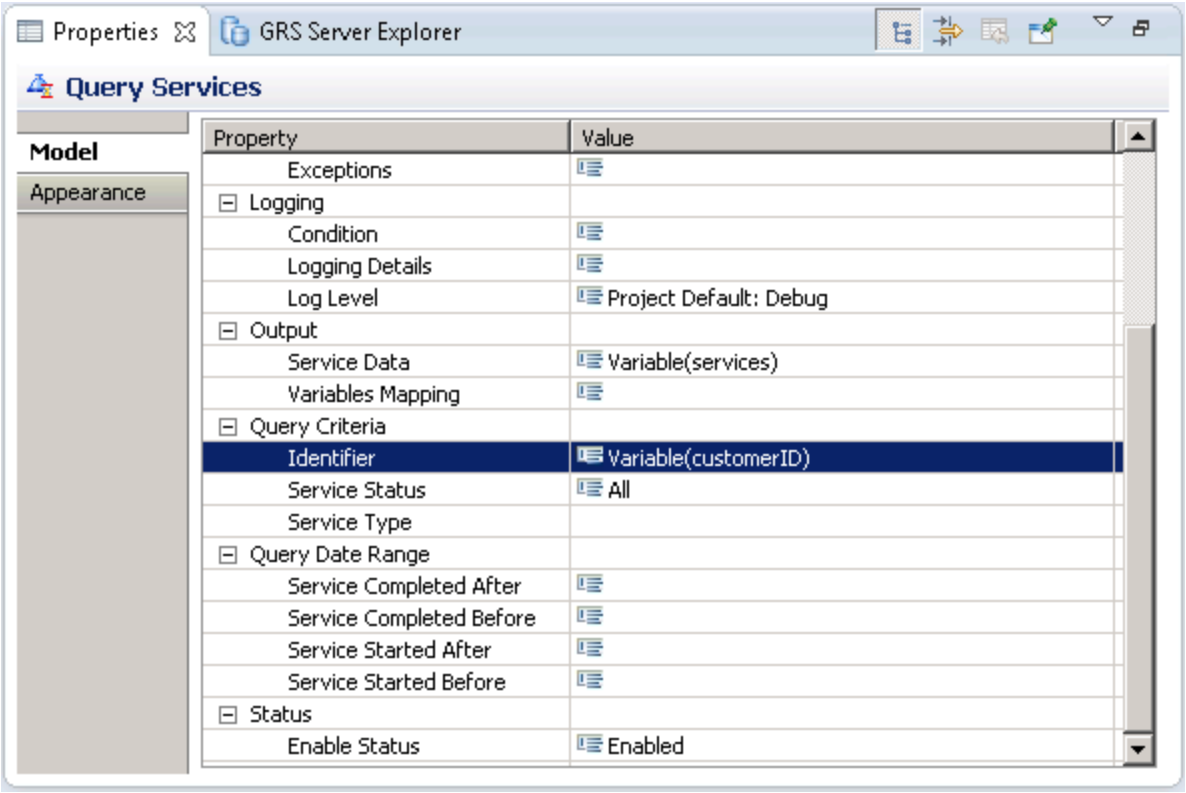
Query Customer Block

This is the Query Customer block. Note the Output: Result is stored in the user variable customer.



Query Services Block

The following is the **Query Services** block.

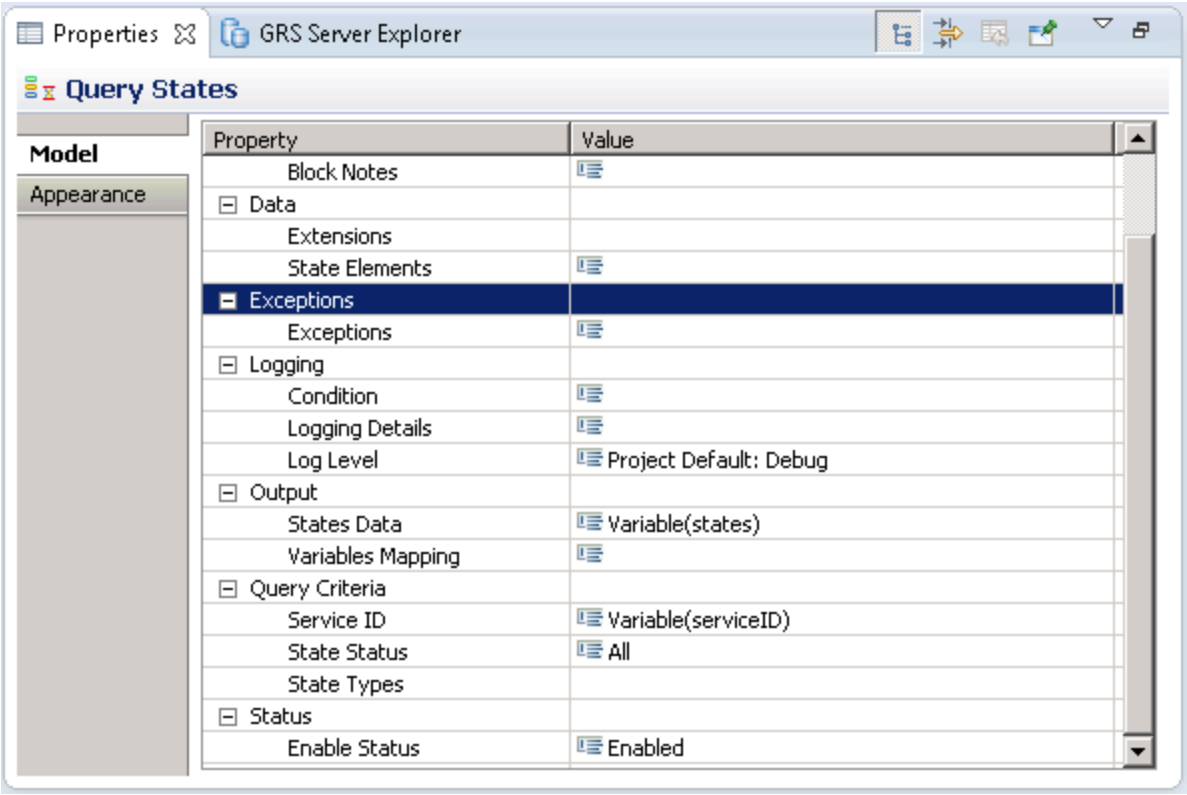


Notes:

- The result Output—`Service Data` is stored in user variable services.
- Query Criteria—`Identifier` is set to `customerID`, meaning we are querying services of the identified customer.
- Query Criteria—`Service Status` is set to `All` to ensure both conditions for active services and completed services can be correctly checked.
- Query Criteria—`Service Type` is not set, ensuring services of all service types are returned for service type-related rule conditions.
- No Query Date Range is set, which is needed for date checking service conditions.

Query States Block

The following is the **Query States** block.

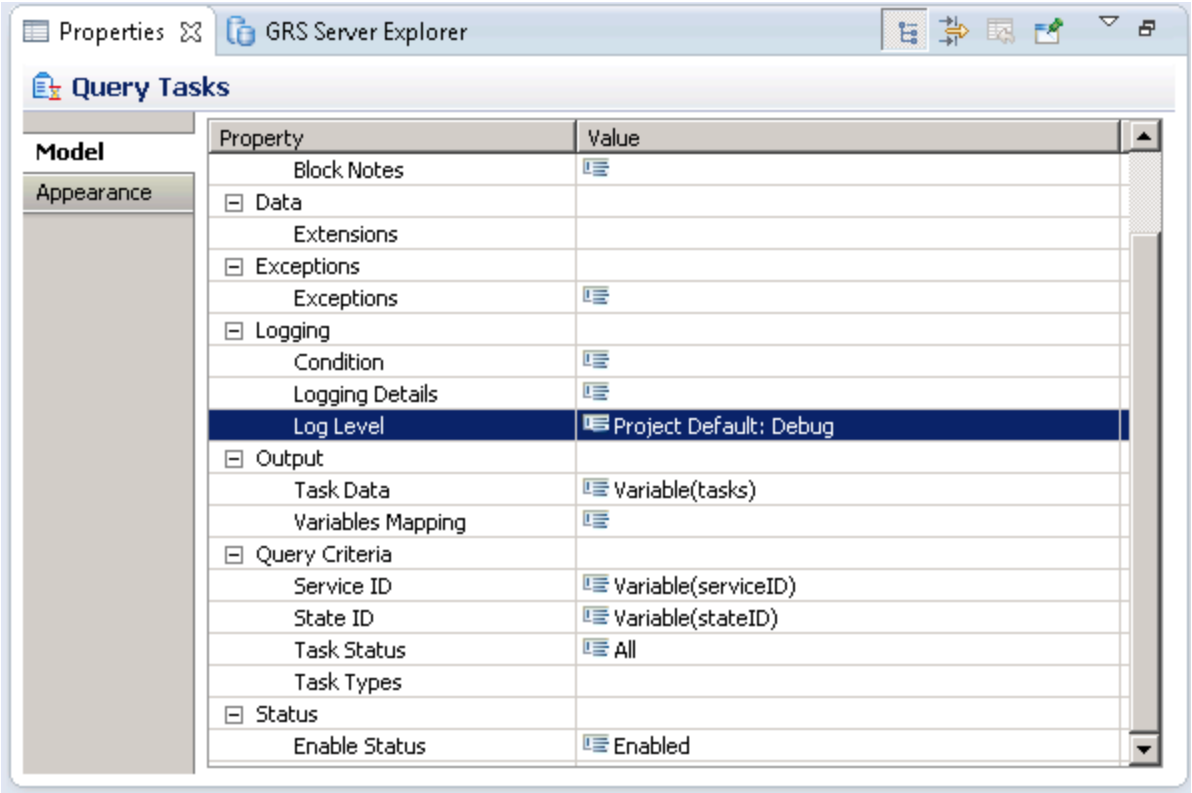


Notes:

- The result Output—`States Data` is stored in user variable states.
- Query Criteria—`Service ID` is set to `serviceID`, which should be previously extracted from **Query Services** results.
- Query Criteria—`State Status` is set to `All` to ensure both conditions for active states and completed states can be correctly checked.
- Query Criteria—`State Types` is not set, ensuring states of all state types are returned for state type-related rule conditions.
- Since Query States may only fetch states of a single service, the current CM template expects all states supplied are associated with a single service. If states of multiple services are aggregated to a CM rule package, the rules may not behave as expected.

Query Tasks Block

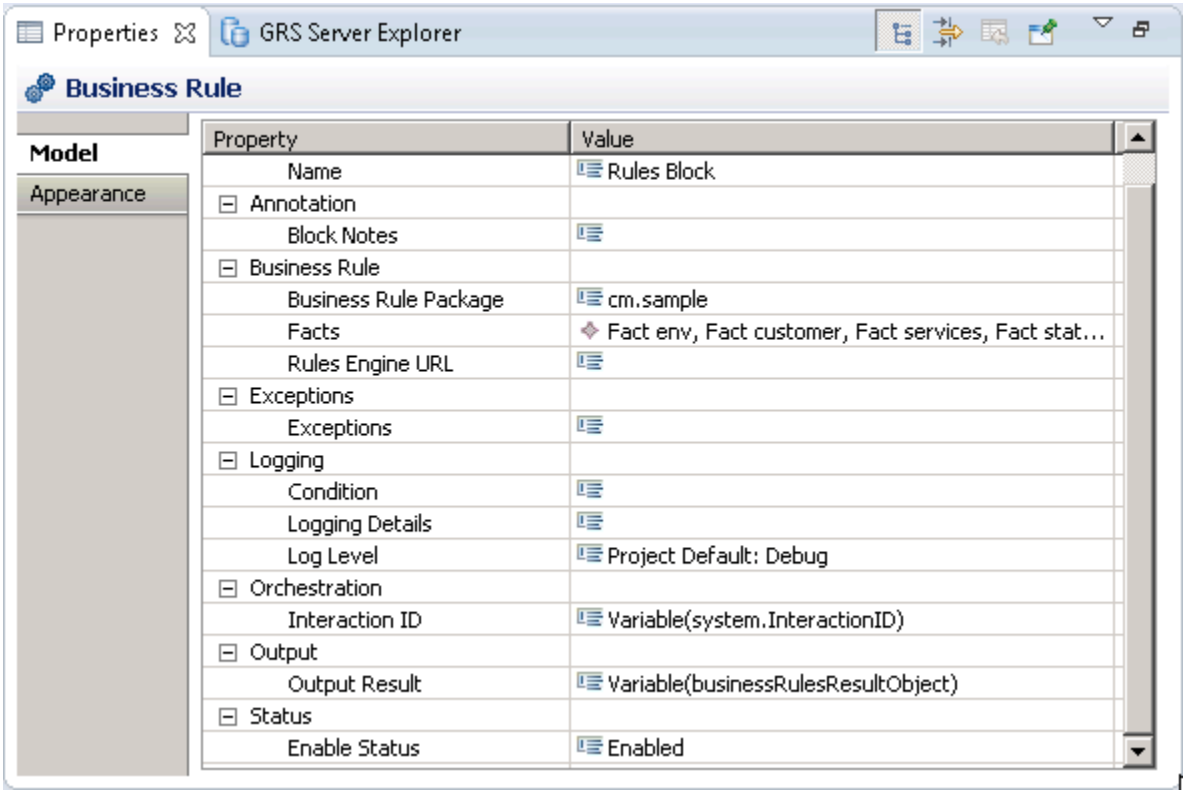
The following is the **Query Tasks** block.



Notes:

- The result Output—`Task Data` is stored in user variable tasks.
- Query Criteria—`Service ID` is set to `serviceID`, which should be previously extracted from **Query Services** results.
- Query Criteria—`State ID` is set to `stateID`, which should be previously extracted from **Query States** results. This field may be omitted when querying tasks associated with a service.
- Query Criteria—`Task Status` is set to `All` to ensure both conditions for active tasks and completed tasks can be correctly checked.
- Query Criteria—`Task Types` is not set, ensuring tasks of all task types are returned for task type-related rule conditions.
- Since **Query Tasks** may only fetch tasks of a single service or state, the current CM template expects all tasks supplied are associated with a single service or state. If tasks of multiple services/states are aggregated to a CM rule package, the rules may not behave as expected.

Business Rules Block

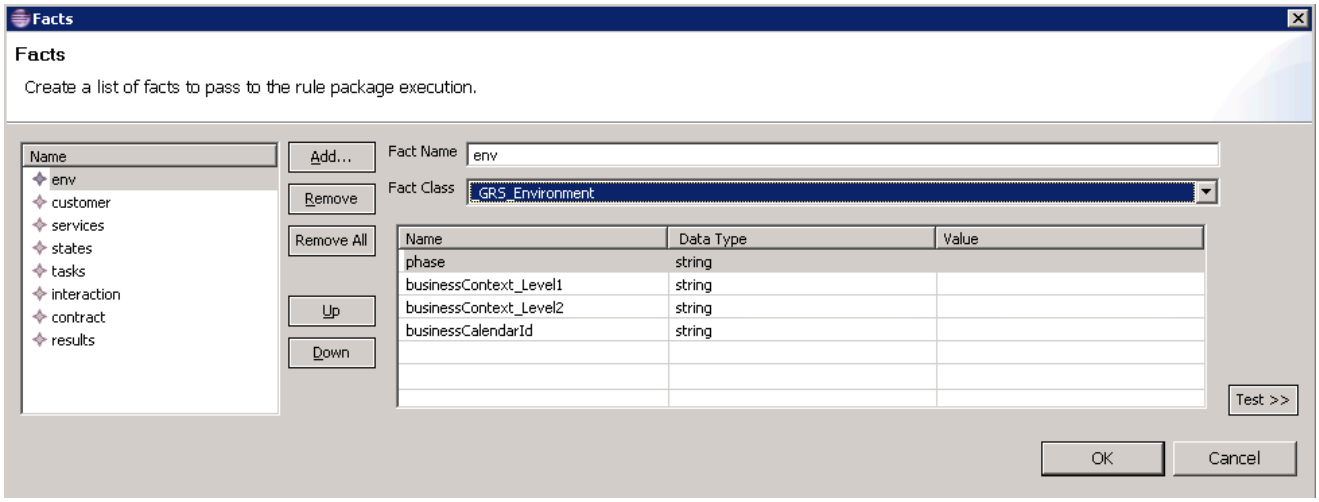


Notes:

- The **Business Rule** block is where GRS is invoked in the workflow/callflow. Before configuring this block, a rule package using the CM template should already be authored and deployed to GRE.
- `Business Rule Package` is the name of the rule package. This is chosen from a list of all deployed rule packages on the GRE specified in **Windows -> Preferences**.
- Output Result is copied to user variable `businessRulesResultObject`.
- Facts brings up a list of all facts to send to GRE. See below for further detail.

Facts

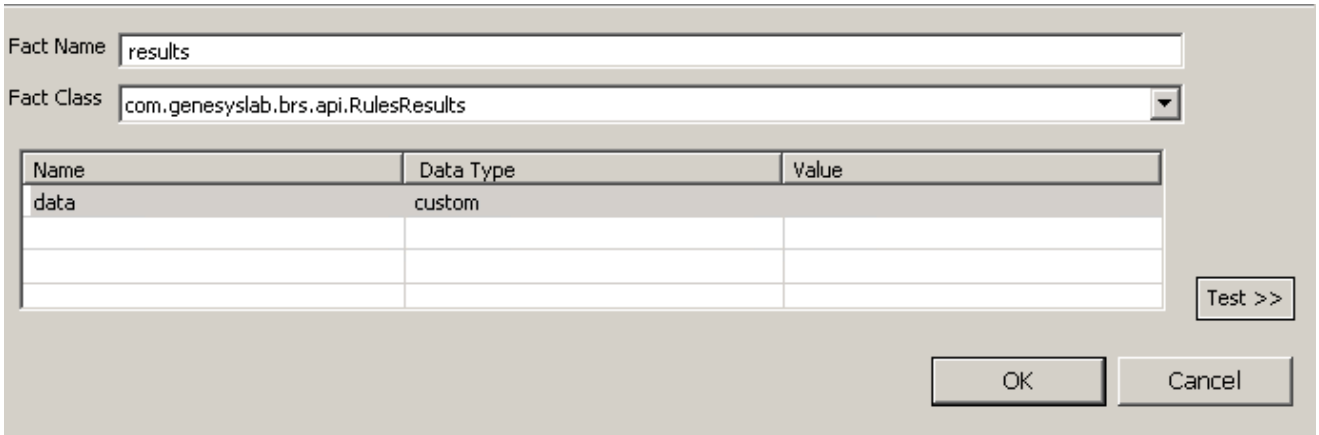
_GRS_Environment



The `_GRS_Environment` fact class is available to rules packages of all types, this fact class is for storing global environment variables.

com.genesys.brs.api.RulesResults

Fact class `com.genesys.brs.api.RulesResults` is a placeholder for storing results from GRE. It is required for CM rule packages. No value should be set for the data field.

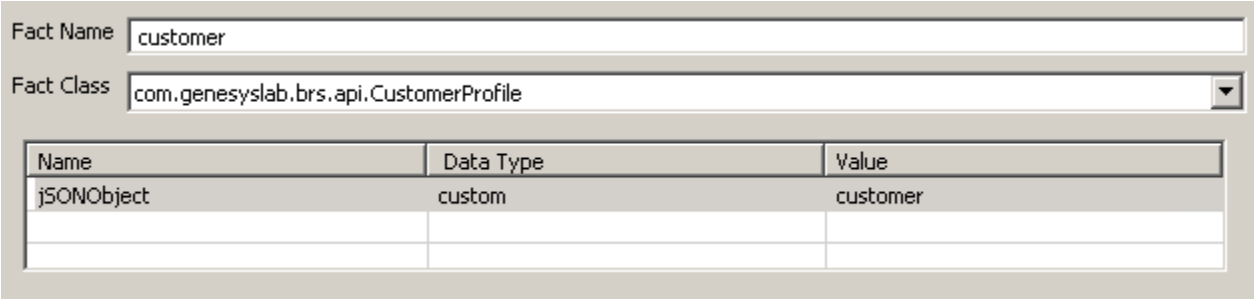


com.genesyslab.brs.api.CustomerProfile

Fact class `com.genesyslab.brs.api.CustomerProfile` provides the customer profile from **Query Customer** block to GRE. It is required only if customer-related conditions are used; otherwise it is ignored. If it is not provided, all customer-related conditions are considered failed.

In this example, the value of `jSONObject` is set to the user variable customer, which was previously

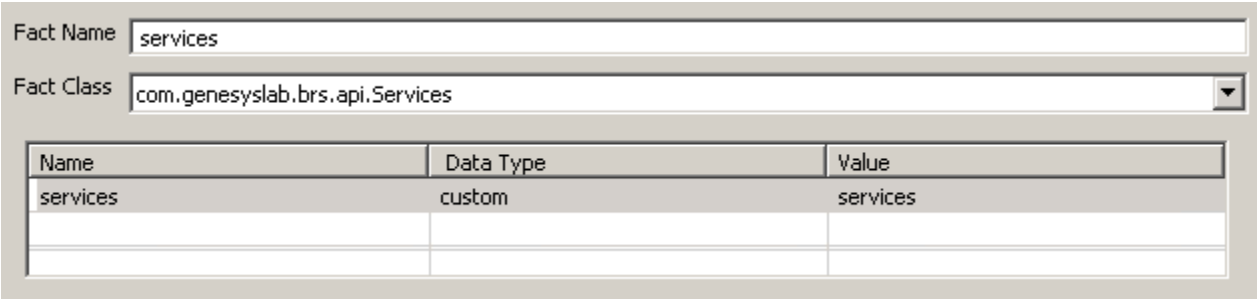
populated by **Query Customer** block.



com.genesyslab.brs.api.Services

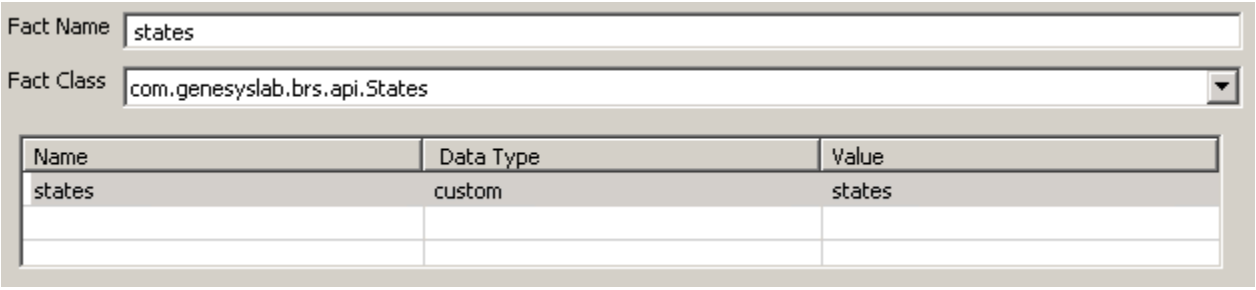
Fact class `com.genesyslab.brs.api.Services` provides services from **Query Services** block to GRE. It is only required if service-related conditions are used; otherwise it is ignored. If it is not provided, all service-related conditions are considered failed.

In this example, the services field is set to the user variable services, which was previously populated by **Query Services** block.



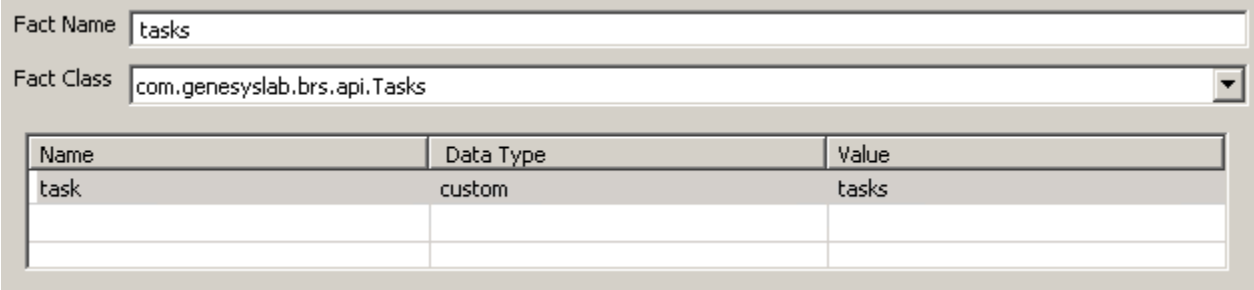
com.genesyslab.brs.api.States

Fact class `com.genesyslab.brs.api.States` provides the states from **Query States** block to GRE. It is required only if state-related conditions are used; otherwise it is ignored. If it is not provided, all state-related conditions are considered failed. In this example, the states field is set to the user variable states, which was previously populated by **Query States** block.



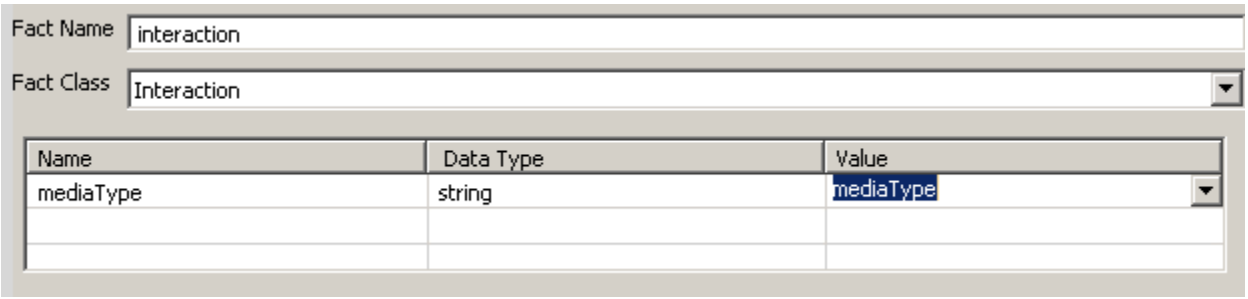
com.genesyslab.brs.api.Tasks

Fact class `com.genesyslab.brs.api.Tasks` provides the states from **Query Tasks** block to GRE. It is required only if task-related conditions are used; otherwise it is ignored. If it is not provided, all task-related conditions are considered failed. In this example, the tasks field is set to the user variable tasks, which was previously populated by **Query Tasks** block.



Interaction

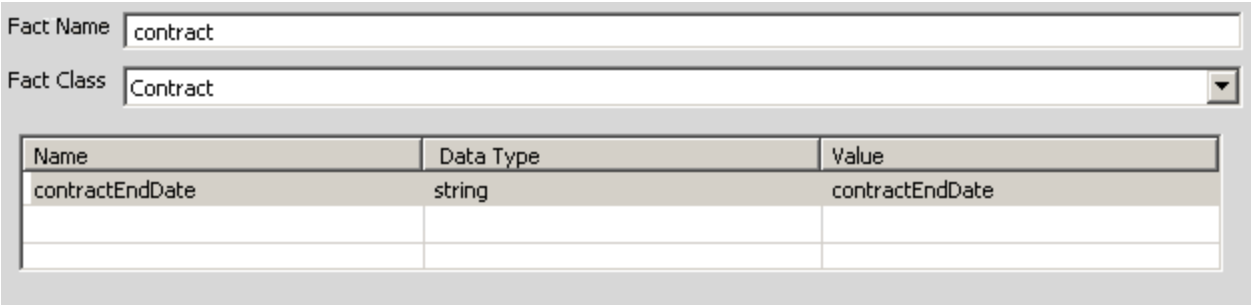
Fact class `Interaction` provides the media type of the current interaction to GRE. It is required only if media type-related conditions are used; otherwise it is ignored. If it is not provided, all media type-related conditions are considered failed. In this example, the `mediaType` field is set to the user variable `mediaType`, which is not actually populated in previous blocks in the current workflow. In normal usage, the user variable for providing media type to this fact should either be pre-populated in a previous block, or hard-coded based on the workflow (for example, if this workflow is only executed for voice).



Contract

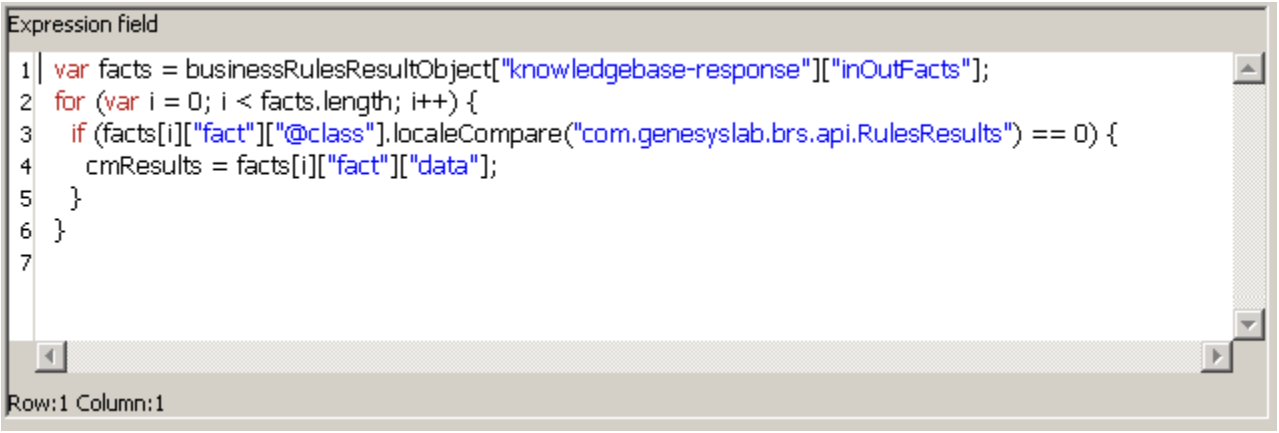
Fact class **Contract** provides the contract end date of the user to GRE. It is only required if contract-related conditions are used; otherwise it is ignored. If it is not provided, all contract-related conditions are considered failed.

In this example, the `contractEndDate` field is set to the user variable `contractEndDate`, which is not actually populated in previous blocks in the current workflow. In normal usage, the user variable for providing contract end date to this fact should should be pre-populated in a previous block (for example, calculated based on service start date and pre-configured contract length, fetched from external services).



ECMA Script Block

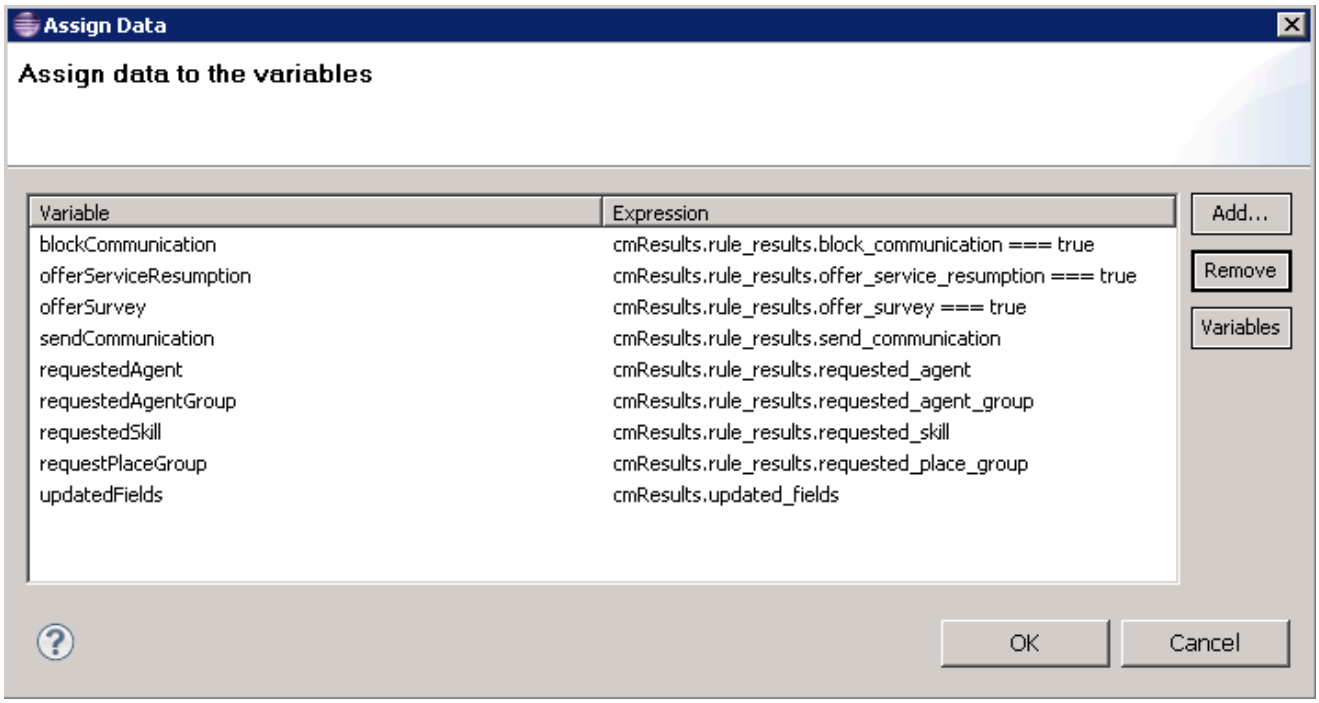
The purpose of this ECMA Script block is to extract the decisions/recommendations of the CM rule package from `businessRulesResultObject` (output of Business Rule block) into the user variable `cmResults`. It is not absolutely necessary, but as `businessRulesResultObject` contains all the input to the Business Rule block, and actual results of the CM rule package are buried in deep.



Assign Block

The purpose of this block is to demonstrate how to access all the different decisions of the CM rule package. Notice all results are in the `rule_results` field of `cmResults`, and the `updated_fields` field of `cmResults` stores any customer profile updates.

In this example, the Boolean values are assigned either true or false even if the rule package did not make any decision. If the "=== true" part is removed, the value would be undefined if no decision was made by the rule package one way or another, which, depending on the situation, could be a valid branch for a **Branching** block.


Assign Data
Assign data to the variables
Variable
Expression
blockCommunication
cmResults.rule_results.block_communication === true
offerServiceResumption
cmResults.rule_results.offer_service_resumption === true
offerSurvey
cmResults.rule_results.offer_survey === true
sendCommunication
cmResults.rule_results.send_communication
requestedAgent
cmResults.rule_results.requested_agent
requestedAgentGroup
cmResults.rule_results.requested_agent_group
requestedSkill
cmResults.rule_results.requested_skill
requestPlaceGroup
cmResults.rule_results.requested_place_group
updatedFields
cmResults.updated_fields
Add...
Remove
Variables
OK
Cancel