BUSINESS RULES

USER GUIDE

STEP Trailblazer 8.3

Table of Contents

Table of Contents **2**

Business Rules **6**

Setup Requirements 6

Additional Information 6

Global and Local Business Rules **8**

Using Business Rules in STEP **9**

Business Rules on Approval **11**

On Approve Process 12

Business Rules that Run on Approve 12

Business Rules with Conditional Attributes **14**

JavaScript business rule bind for Conditional Validity 14

JavaScript business rule bind of Workflow State 15

Business Conditions in Data Profiling **17**

Business Rules with GDSN **18**

Business action on inbound messages 18

Business Actions for GDSN 18

Execute command 19

Create CIC Object and Start Workflow 21

Set CIC Status for Publication 22

Update Package Hierarchy Status 23

'Evaluate JavaScript' with Binding to 'GDSN Publisher Product Validation' 24

Set Pending File 24

Execute Pending File 25

Business Rules in an Import Configuration **26**

Add a Business Rule to an Import 26

Applying Business Rules 28

Business Rule Limitations 29

Business Conditions in Web UI **30**

Configuring a Condition 30

Business Rules in Workflows **34**

Business Rule Common Use Cases **35**

Object Approval 35

Imports / Inbound Integration Endpoint 35

Exports / Outbound Integration Endpoint 35

Workflows 35

Reusable Helper Functions 35

Initial Set Up for Business Rules **36**

Configuration 36

Creating a Business Rule or Library **39**

Editing a Business Rule **41**

Editing a Business Library **47**

Testing a Business Rule **50**

Maintaining a Global Business Rule **55**

Business Rule Tab 55

Usage Tab 56

Statistics Tab 57

Log Tab 57

Status Tab 58

Deleting a Business Rule or Library **60**

Exporting Business Rule Definitions as Comments **62**

Running a Business Rule **63**
Business Actions **64**
Specifying a Business Action **68**
Business Action: Add Attribute Link **72**
Business Action: Add Reference **74**
Business Action: Add Referenced By **76**
Business Action: Automatic Classification **78**
Business Action: Claim **80**
Business Action: Execute JavaScript **82**
Adding a Bind **85**
Java vs. JavaScript **87**
JavaScript Binds **88**
Approve Context Bind **94**
JavaScript Methods 94
Business Condition 95
Business Action 96
Asset Importer Configuration Bind **97**
Attribute Related Binds **99**
Attribute 100
Attribute Group 100
Attribute Validated Parameter 101
Attribute Value 101
List of Values 102
Unit 103
Unit Group 103
Current Object Bind **104**
Data Container Binds **109**
Current Data Container Object 109
Pair of Attribute Values 110
e-Signature **112**
Using e-Signature in Web UI **113**
Applying e-Signature to a Transition in a Workflow **115**
Basic e-Signature Bind **118**
Event Binds **120**
Current Event Batch 121
Current Event Queue 121
Current Event Type 121
Event Queue 122
Event Type 122
Event Bind Examples **124**
Checking Current Event 124
Queuing a Derived Event 125
Gateway Integration Endpoint Bind **126**
GDSN Binds **129**
XPath Expressions 131
MongoDB Binds **133**
JSON Document 133
MongoDB Context 133
Object Aspects Binds **135**
ID 135
Name 136
Other Binds **137**
Conditionally Invalid Values 137
Import Change Info 138

Logger 138
Lookup Table Home 138
Mail Home 139
Business Rules with Conditional Attributes 140
JavaScript business rule bind for Conditional Validity 140
JavaScript business rule bind of Workflow State 141
Using JavaScript with Lookup Tables 143
Reference Type Bind 145
Secondary Object Bind 147
STEP Manager Bind 150
Survivorship Rule Source Objects Bind 152
Tree Object Binds 153
Asset 154
Classification 154
Entity 154
Product 154
User Binds 155
User 156
User Group 156
Workflow Binds 157
Current Transition 158
Current Workflow 159
Workflow Parameters 159
Workflow State 159
JavaScript Considerations 160
JavaScript Examples 161
Set an Attribute Value 161
Send Asset Content 162
JavaScript Exception Handling 164
Localized Messages for JavaScript Business Rules 166
Return Statement 166
Throw Statement 166
Adding a Localized Business Rule Message 167
Business Action: Generate Match Codes 170
Business Action: Initiate Items In STEP Workflow 171
Business Action: Merge Attribute Values 173
Business Action: Overlap Analysis 175
Business Action: Reference Other Business Action 176
Business Action: Remove Attribute Link 177
Business Action: Remove Object from STEP Workflow 179
Business Action: Remove Reference 181
Business Action: Send Email 183
Business Action: Send Republish Event 188
Business Action: Set Attribute Value 190
Business Action: Set Name 193
Business Action: Set Object Type 195
Business Action: Set Product to Classification Link Type 197
Business Action: Set Workflow Variable 199
Business Action: Standardize Address 202
Address Standardization Modes 203

Renew Address Validations 204

CASS Validation 205

Business Action: Trigger STEP Workflow Event 206

Business Conditions 208

Specifying a Business Condition 210

Business Condition: Attribute Value Comparison 214

Business Condition: Evaluate JavaScript 216

Business Condition: Function 219

Business Condition: LOV Cross-Validation 223

Business Condition: Reference other Business Condition 226

Business Condition: Validate Product Variant 228

Business Condition: Valid Hierarchies 229

Business Functions 231

Calling a Business Function from a JavaScript Business Rule 232

JavaScript Function 236

User-Defined Functions 240

Business Libraries 241

Business Rules

Business rules are units of business logic that are stored as objects in System Setup. Business rules are used for many different purposes in STEP and come in three variants:

	Input	Output	Side effects allowed
Business actions	Current object, current event batch, etc. provided by the context in which the action is executed. For more on business actions, see the **Business Actions** topic.	None	Yes
Business conditions	Current object, current event, etc. provided by the context in which the condition is evaluated. For more on business actions, see the **Business Conditions** topic.	Boolean result of evaluating the condition and a message for the user	No
Business functions	Input parameters defined by the function and provided by the functionality evaluating the function. For more on business functions, see the **Business Functions** topic.	Result of evaluating the function	No

A fourth type of business rule, **business library**, allows users to define JavaScript library functions that can be called from other JavaScript-based business rules. For more information on business libraries, see **Business Libraries** topic documentation.

For more information on differentiating between scenarios where one business rule is more useful than another, see the **Business Rule Common Use Cases** topic in the **Business Rules** documentation.

Setup Requirements

The following set up is required to use business rules:

1. Perform the one-time setup steps described in **Initial Set Up for Business Rules**.
2. Create a business rule as described in **Creating a Business Rule or Library**.
3. Edit the business rule as described in the **Editing a Business Rule**, or **Editing a Business Library**.
4. Test the business rule as described in **Testing a Business Rule**.

Additional Information

The following information is useful prior to creating and after a business rule is set up:

1. Review the most common ways to use business rules as described in **Using Business Rules in STEP**.
2. Review the decisions that affect how a business rule runs as described in **Running a Business Rule**.
3. Maintain or modify global business rules as described in **Maintaining a Global Business Rule**.
4. To understand more about the business functions and how to use them, see **Calling a Business Function from a JavaScript Business Rule**.
5. Delete a business rule as described in **Deleting a Business Rule or Library**.
6. Maintain or modify local business rules as described **Business Rules and Workflows**.
7. Understand the differences between global and local business rules as described in **Global and Local Business Rules**.
8. Export business rule definition for comparison purposes in an external source control system as described in **Exporting Business Rule Definitions as Comments**.

Global and Local Business Rules

The scope of a business rule can be either local or global.

- Global business rules are reusable and can have multiple functions, for example, in different workflows, during object approval, and in imports and bulk updates. Global business rules are available when you open a business rule selector.
- Local business rules are specific to one process such as when a specific workflow enters a specific state. Local business rules are not available from a business rule selector. Global business rules can be added to a workflow and then copied for local use. Local business rules are typically located below the workflow where they are used.

Icons are used to differentiate between conditions and actions. A hand indicates the business rule is global.

	Condition	Action
Global		
Local		

Local and global business actions can be executed when entering a workflow state (On Entry), when exiting a workflow state (On Exit), when a deadline is met and the Escalation background process is run, and when the workflow transitions from one state to another (On Transition).

Local and global conditions can be tested on transitions. A transition cannot take place if the associated business action evaluates to false.

For more information about local business rules, see the **Business Rules and Workflows** topic in the **Workflows** documentation.

Using Business Rules in STEP

Wherever business rules are used within STEP, they are always tested or executed in relation to one object at a time, and in a specific context / workspace. The most common places to use business rules are included in the list below.

- **Approvals** - Conditions can be tested when approval is attempted on an object under revision control, and the condition can allow or prevent the approval. Actions executed on approval and can modify data in STEP (typically data on the object being approved), send emails etc. related to the approval.

  For more information, see **Business Rules on Approval** documentation.

- **Automatic Classifications** - Actions can be executed to automatically classify objects, and can be applied, for example, on approval, during an import, or as a part of a workflow.

  For more information, see **Using Automatic Classification with Business Actions** documentation.

- **Bulk Updates** - Conditions can be tested as a precondition for executing an action. Actions can be executed.

  For more information, see **Run Business Rule Operation for Bulk Updates** documentation.

- **Conditional Attributes** - The JavaScript business action 'Conditionally Invalid Values' bind resolves to a set of all values.

  For more information, see **Business Rules with Conditional Attributes** documentation.

- **Data Profiles** - Conditions can be tested against all objects in a category (for example, a part of the Product hierarchy), and the result of the tests can be displayed on the Profile Dashboard.

  For more information, see **Business Conditions in Data Profiling** documentation.

- **Gateway Integration Endpoints** - Accessed from JavaScript in business rule conditions and actions, the bind can work with a variety of the REST methods.

  For more information, see the **Gateway Integration Endpoint Bind** documentation.

- **GDSN** - Actions can be used in a variety of ways in the GDSN component, for example, to send out messages, update status, and start workflows. The following describes how to use GDSN specific business actions and how to run business actions when receiving messages from GDSN.

  For more information, see **Business Rules with GDSN** documentation.

- **Imports and Inbound Integration Endpoints** - Conditions can be tested during imports, and the condition can allow or prevent the creation or update of objects. Actions executed during import can modify the objects being imported, apply actions to objects being imported, send emails, and start workflows, etc.

  For more information, see **Business Rules in an Import Configuration** documentation.

- **Matching, Linking, and Merging** - Actions and Conditions can be used to normalize the data for comparison to identify the duplicate products in STEP. Actions can also be used in relation to Golden Records Survivorship Rules.

For more information, see **Matching, Linking, and Merging JavaScript Binds** documentation or **Golden Records Survivorship Rules** documentation.

- **Outbound Integration Endpoint** - Conditions can be used as event filters for an event-based OIEP. Actions can be executed via a pre-processor for any OIEP, or as a event generator to generate derived events to export or publish for an event-based OIEP.

  For more information, see **OIEP - Event-Based - Event Triggering Definitions Tab** documentation or **OIEP - Pre-Processor - Business Action** documentation.

- **Web UI** - Actions can be executed using 'Run Business Action' component to update the object. Conditions can be evaluated using 'Initiate Business Condition Action' component.

  For more information, see **Business Conditions in Web UI** documentation.

- **Workflows** - Conditions can be tested within workflows to allow or prevent transitions from one state to another. Actions can be executed when entering a state, when leaving a state, when performing a specific transition, and when a deadline is met. Actions can also modify the object being tracked by the workflow, modify other objects in STEP, modify the workflow behavior, send email, start other workflows, etc.

  For more information, see **Business Rules in Workflows** documentation.

Business Rules on Approval

Global business rules valid for the node being worked can be evaluated or executed during the approval process.

- When a condition evaluates to false, the approval fails, an error message is displayed, and no actions are executed.
- When a condition evaluates to true, the selected actions are executed.

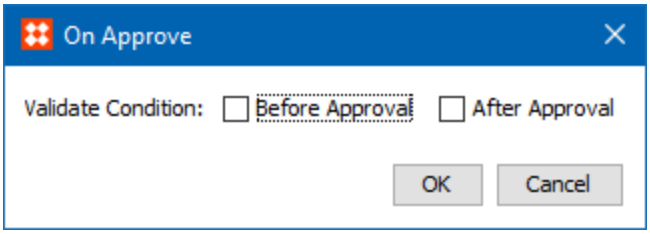
Global business rules are set to run on approval in the Business Rule editor. For more information, see the **On Approve** section of the **Editing a Business Rule** topic.

For information on running business rules during the import process, see the **Business Rules in an Import Configuration** topic.

Evaluate Condition

On conditions, specify to validate the condition before and/or after approval.

- **Before Approval** tests are run in the Main workspace.
- **After Approval** tests against the object as it would appear int he Approved workspace, assuming the approval succeeds. Technically, the object is approved, the condition is tested, and if the condition fails, the approval is rolled back. This allows a condition to test data that is not owned by the object being approved, such as inherited attribute values and references, referenced objects, children, etc., which could be different or non-existent in the Approved workspace. Since approved data is typically what is made available to downstream systems, this is a valuable option.



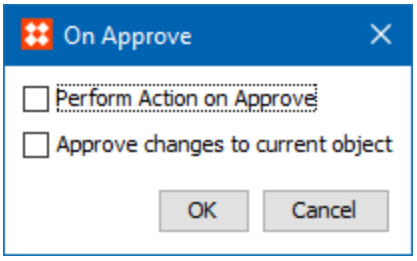
Note: The condition is only guaranteed to be met for a given object at the moment it is tested. If the condition depends on data owned by other objects, those objects can change and be approved without the condition being reevaluated. Therefore, it is common setup to test data on the objects that owns the data.

For details on adding a condition, see the **Specifying a Business Condition** topic.

Execute Action

On actions, specify to execute the action on approve and/or to approve changes as well. Execution always takes place in the main workspace prior to the actual approval.

- **Perform Action on Approve**
- **Approve changes to current object** allows changes made to the object being approved to be included in the approval.



For details on adding a condition, see the **Specifying a Business Action** topic.

On Approve Process

When objects are selected for approval, and business rules are set to run On Approve, the steps below are followed:

1. If there is nothing to approve, exit without running business rules.
2. If there is data to approve, run actions configured to be executed on approval.
3. Test all conditions configured to be validated before approval (for sub-conditions, run only until one of them fails). If one or more fails, go to step 8.
4. If using an event-based OIEP, run event filter conditions and event generator actions. Exit and roll back if mandatory data is missing.
5. Perform the approval, including standard dependency checks. If one or more fails, go to step 8.
6. Run all conditions configured to be validated after approval (for sub-conditions, run only until one of them fails), and perform standard mandatory checks. If one or more fail, go to step 8.
7. Run mandatory checks in Approved workspace. Exit and roll back if mandatory data is missing.
8. Commit changes if all previous steps were successful, or roll back changes if any conditions failed.

Due to the order of the On Approve Process, note the following:

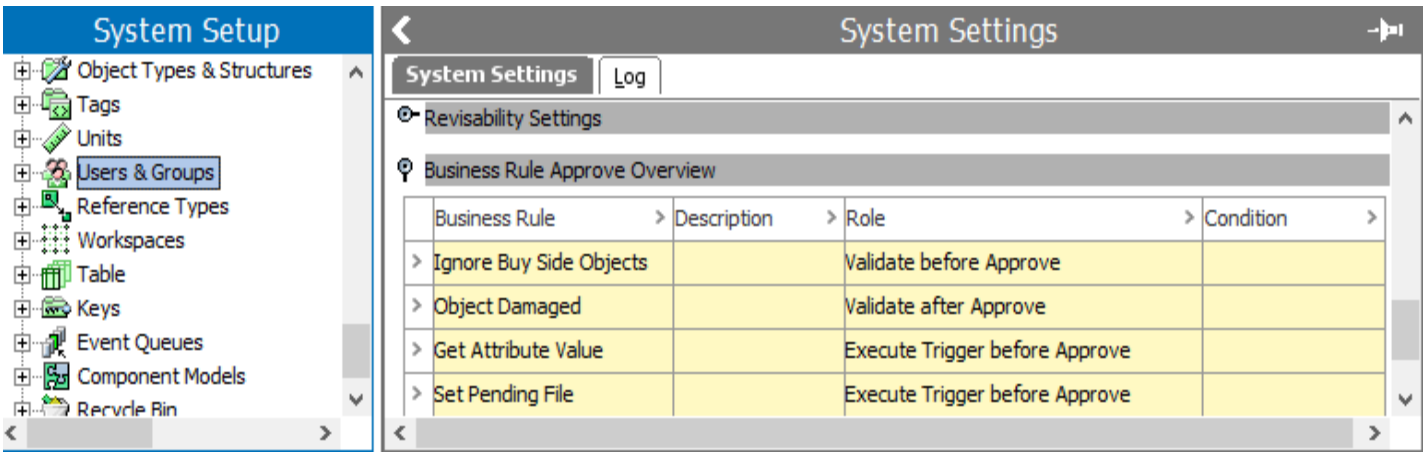
- Changes made by actions are not kept when conditions before or after approval are false.
- If the 'Approve changes to current object' option is not checked, and an action makes a change to the object being approved, a complete approval cannot be achieved.
- Actions can make changes that cause a condition to become true. For example, a condition could test for the existence of a reference that is added by an action during approval.

Note: If multiple conditions and actions are tested and/or executed on approval, the order in which the conditions are tested and the actions are executed is not guaranteed.

Business Rules that Run on Approve

Follow the steps below to display a list of all business rules set to run on approval.

1. In System Setup > Users & Groups > click the System Settings tab.
2. Open the **Business Rule Approve Overview** flipper to display business rules configured to be evaluated or executed On Approve.


System Setup
Object Types & Structures
Tags
Units
Users & Groups
Reference Types
Workspaces
Table
Keys
Event Queues
Component Models
Recycle Bin
System Settings
System Settings
Log
Revisability Settings
Business Rule Approve Overview
Business Rule
Description
Role
Condition
Ignore Buy Side Objects
Validate before Approve
Object Damaged
Validate after Approve
Get Attribute Value
Execute Trigger before Approve
Set Pending File
Execute Trigger before Approve

Business Rules with Conditional Attributes

For information on setting up a conditionally valid attribute, see the **Conditional Attribute Display** documentation in the **System Setup / Super User Guide**.

For information on adding a bind, see the **Adding a Bind** topic.

JavaScript business rule bind for Conditional Validity

It is possible to make use of the Conditionally Invalid Values bind in a JavaScript business action. This will resolve to a (possibly empty) set of values, and this set will contain all values which has value conditions that for the current node evaluates to false.

Note: Note that this binding is unrestricted in the sense that it is always available, not just from workflows or imports, etc.

This is a JavaScript example for clearing conditionally invalid values.

Required binds:

- invalid:"Conditionally Invalid Values"-binding
- logger:"Logger"-binding

```
//acquire an iterator - easiest way to deal with Sets
var iter = invalid.iterator();
//iterate over conditionally invalid values
while (iter.hasNext()){
//get the value
var val = iter.next();
//test if there is a value and if it is local
var clean = val.isLocal() && val.getSimpleValue();
//do some chatting to the log
var msg
if (clean)
msg = "Cleaning up";
else
msg = "Ignoring";
msg += " conditionally invalid ";
if (val.isLocal())
msg += "local value";
else
msg += "non-local value";
```

```
msg+= val.getAttribute().getID() + " [" + val.getSimpleValue() + "]";

logger.info( msg );

//in case of local value - delete it

if (clean) val.setSimpleValue("");//empty string handled as null and deletes
local values

}
```

JavaScript business rule bind of Workflow State

It is possible to make use of the workflow state bindings for conditions and actions that, on the import, can resolve and be used for general evaluation and handling of the product being imported.

This is an example business condition that be applied on import of a maintenance Smartsheet, exported from the Web UI Task List.

Required binds:

- state:"Workflow state"-binding
- node:"Current Object"-binding
- logger:"Logger"-binding

```
//up front test if the import carries expectation of specific workflow state

if (!state) {

logger.info("No import state provided");

//could have instead return some rejection to say no import if a state is not
supplied

return true;

}

var instance, task, msg;

//acquire the workflow instance for the node and workflow

instance = node.getWorkflowInstance(state.getWorkflow());

if (!instance) {

//simply reject since workflow has been terminated or never started

msg = "Rejected : Workflow " + state.getWorkflow().getTitle() + " not started
for " + node.getTitle();

logger.info(msg);

return msg

}

//now look for actual task for node in supplied state

task = instance.getTask(state);

if (!task){
```

```
//no task also causes rejection
msg = "Someone might have changed data " + node.getTitle() + " has no task for
state " + state.getID() + " in worflow " + state.getWorkflow().getTitle();
logger.info(msg);
return msg;
}
logger.info('Accepted: we have a task and will now continue with import');
return true;
```

Business Conditions in Data Profiling

You can test conditions as part of a data profiling process and the results can be visualized in a widget. Conditions can be tested on entire hierarchies as part of the profiling process.

For example, a user wants to know the number products without a selling price from specific vendor. A business condition can filter for the vendor during profiling , which then can check the selling price value.

For more information, see the **Data Profiling** user guide.

Using a business condition within data profiling involves the following setup:

- Ensuring the business rule is valid for the object type being profiled.
- Setting the object type of the node to allow profiling via the Enable Profiling parameter. For details, see the **Data Profiles** topic in the **Data Profiling** documentation.

Displaying a data profile widget involves the following configuration:

- Configuring a data profile widget as defined in the **Profile Configuration** topic.

Business Rules with GDSN

Business rules can be used in a variety of ways in combination with the GDSN component. They can, for example, be used to send out messages, update status, and start workflows. The following describes how to use GDSN specific business actions and how to run business actions when receiving messages from GDSN.

Business action on inbound messages

In the inbound format configuration you can configure which business actions should be run when different message types are received. You can use this to update status, transition workflows or send out notification e-mails.

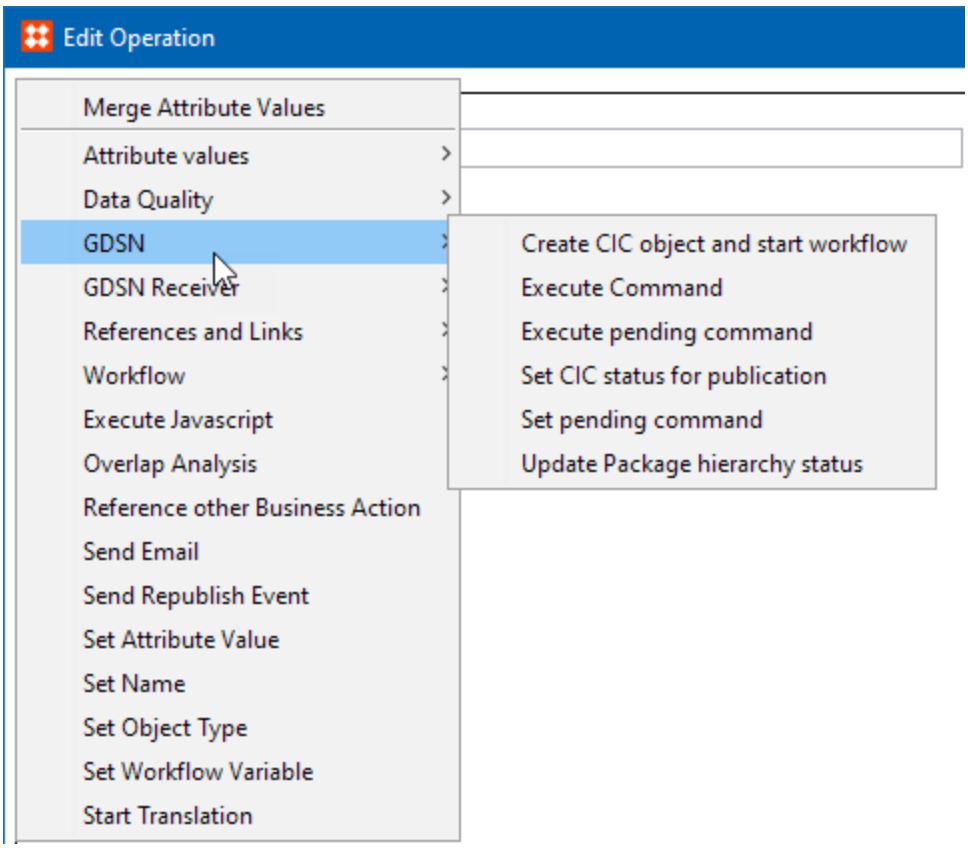
In the inbound format configuration, you can configure a map from strings to XPaths. When a message is received, the XPaths are evaluated and a map is created from the string keys to the evaluated values. This map can be bound into a JavaScript business action so that the business action can access parts of the incoming message.

For general information about how to set up business actions, see **Business Actions** in the **Business Rules** documentation.

Business Actions for GDSN

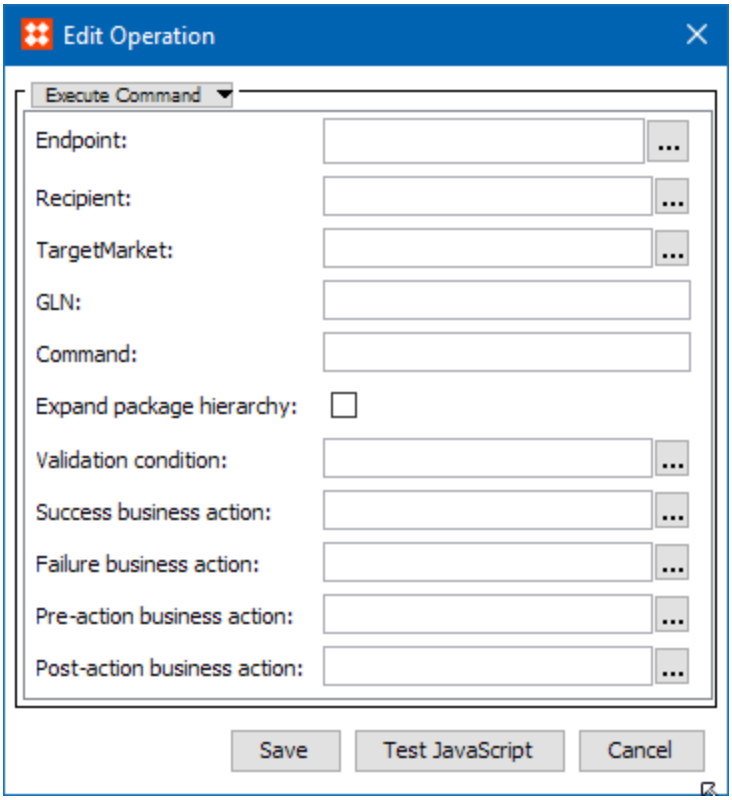
1. In **System Setup**, expand **Business Rules**, and then select the relevant action.
2. On the **Business Rule** tab, in the lower left corner, click **New Business Action**.
3. Enter ID and Name in Create Business Action window
4. Click on Create button.
5. Once it is created, right click on it and select Edit Business Rule.
6. In the field that opens, click **Add new Business Action** link and the **Merge into** icon (**Edit operation** icon) will appear in the empty box above.
7. Click the **Edit Operation**  icon.
8. In the **Edit Operation** dialog, select GDSN from the dropdown.

The following actions are available:



Execute command

This action sends out a message to a data pool.

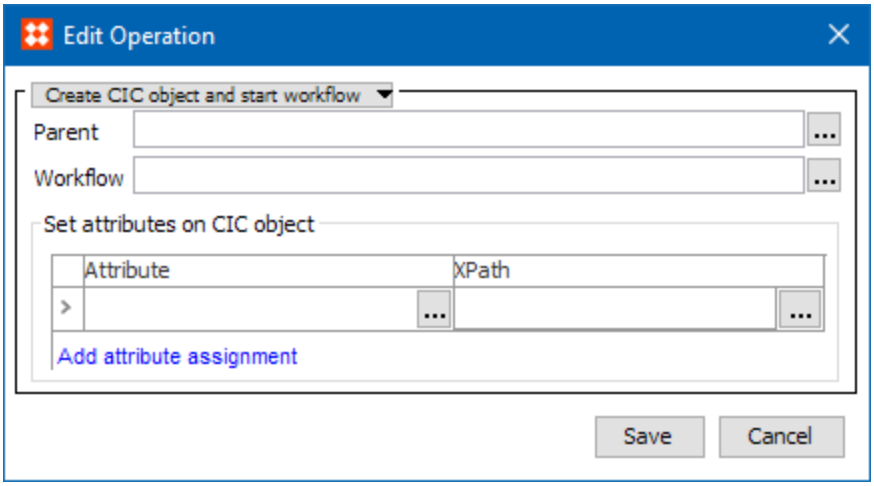


Field	Description
Command	Enter the command parameter that matches the command from the outbound configuration of the message type to be sent out.
Endpoint	Specify the outbound endpoint to be used to send out the message. A list of available endpoints displays when you click the ellipsis button **(…)**.
Expand package hierarchy	When checked, selecting a hierarchy top node exports the entire hierarchy below the node.
GLN	Specify the GLN the products must be sent from. This is the GLN the MyGLN parameter tag resolves to.
Recipient	Optional For a publish template, specify the recipient you want to publish the products to. The Recipient parameter tag will contain the GLN of the recipient.

Field	Description
Target Market	Specify the target market the message should be sent in. The target market parameter tag will contain the target market attribute value of the selected target market.
Validation Condition	Business condition that should be validated successfully before sending message.
Success business action	Specify which business action to run if the Validation condition is successful.
Failure business action	Specify which business action to run if the Validation condition fails.
Pre-action business action	Specify the Business Action that runs first (before any other Business Rules) when sending message.
Post-action business action	Specify the Business Action that should run after sending message.

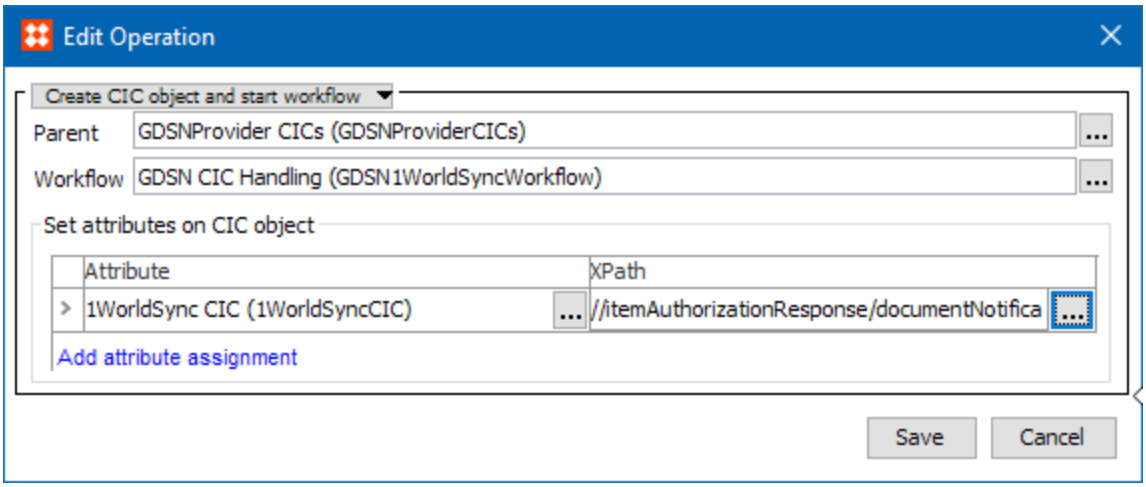
Create CIC Object and Start Workflow

This option creates a CIC entity and starts it in a workflow. It is connected with the product the CIC applies to and with the recipient that sends the CIC. Furthermore, using XPaths, it is possible to place parts of the CIC message in an attribute on the CIC object.



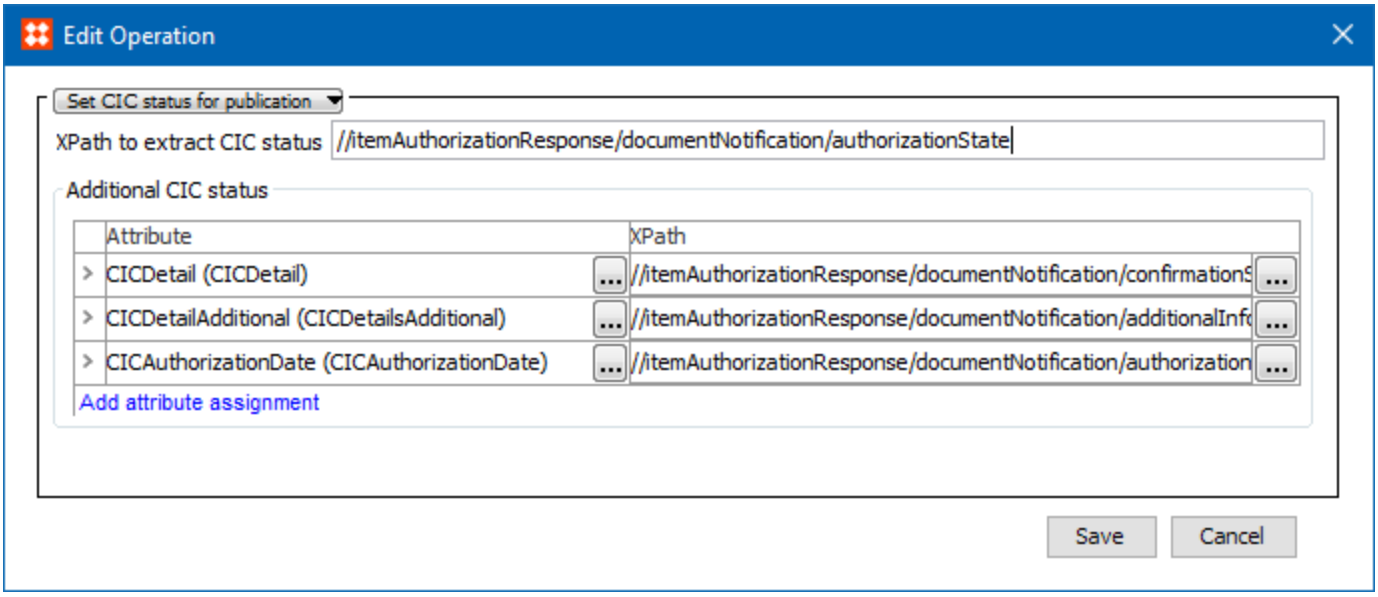
Field	Description
Parent	The CIC entity is created below the entity specified.
Workflow	The CIC entity is started in the workflow specified.
Set attributes on CIC object	Click the **Add attribute assignment** link. Select the attributes to be set on the CIC object and the XPaths to be evaluated to find the values from the inbound message.

The image below shows, example of the 'Create CIC object and start workflow' business action.



Set CIC Status for Publication

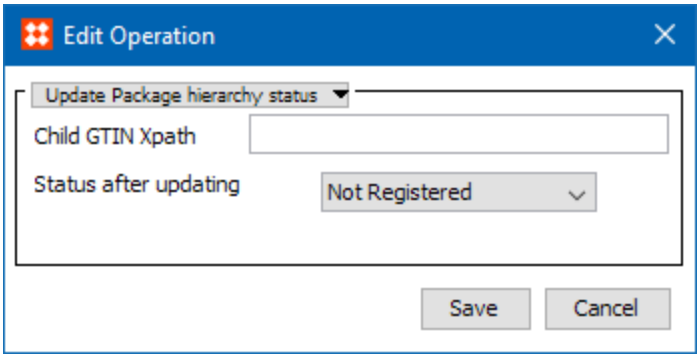
Sets the CIC status for a publication to a recipient.



Field	Description
XPath to extract CIC status	Mandatory Specify an XPath that points out the CIC status in the inbound message.
Additional CIC status	Optional Specify an XPath to additional CIC information and point the attribute that should be used to store this information. For example, the additional CIC information could be text values that go along with the CIC999 status or it could be the date CIC status date.

Update Package Hierarchy Status

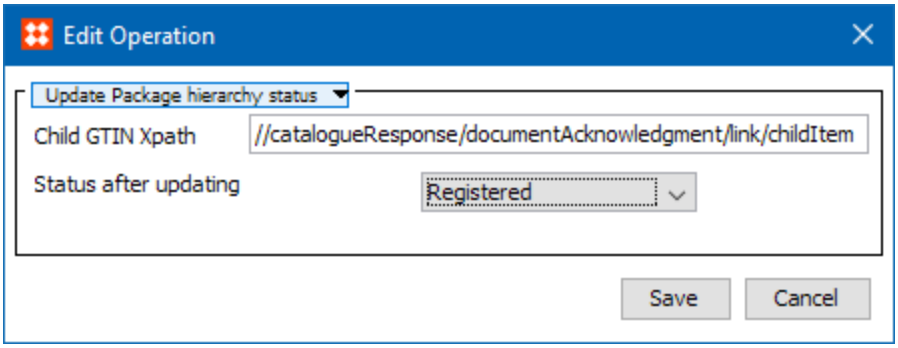
The option sets the registration status of a package hierarchy link.



In the **Child GTIN XPath** field, specify an XPath that points out the GTIN of the child product in the inbound message.

From the **Status after Updating** list, select the status that you want the link to have after the action has been run.

The image below shows, example of the 'Update Package hierarchy status' business action.



'Evaluate JavaScript' with Binding to 'GDSN Publisher Product Validation'

The 'Evaluate JavaScript' plugin in connection with the 'GDSN Publisher Product Validation' binding provides the option to write JavaScript business rules that can access information about a product instance that is registered for a given data pool.

For example, this could be used to write a script that can validate if all required attributes for a given data receiver have been populated on a given product. For more information about the Validation Condition option, see **GDSN Publish Action Button** or **GDSN Register Action Button**.

Set Pending File

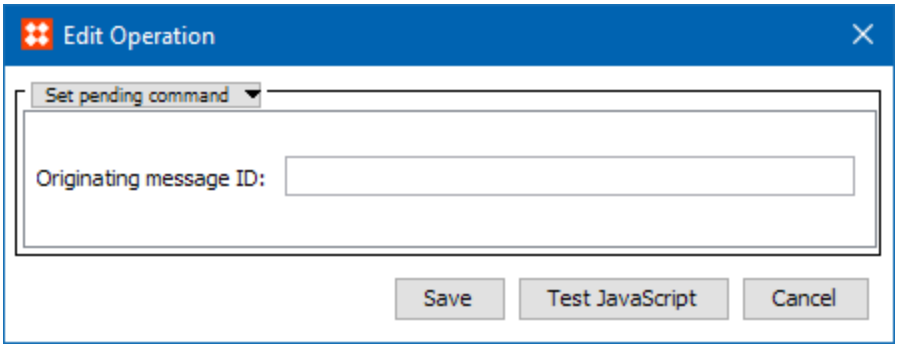
This Business Action is used for the 1WorldSync data pool in conjunction with using the 'Pre command' option in the Web UI (**GDSN Web UI Buttons**). When a pre-command is executed, the system will await a response from the data pool and GDSN before the real command is executed.

For the 1WorldSync data pool, these responses are not combined in one file and will thus not be received at the same time. Instead the responses are received asynchronously, one response message will be received from the 1WorldSync data pool and another response message will be received from the GDSN. Only when both responses have been received, is the real command (the pending command) executed.

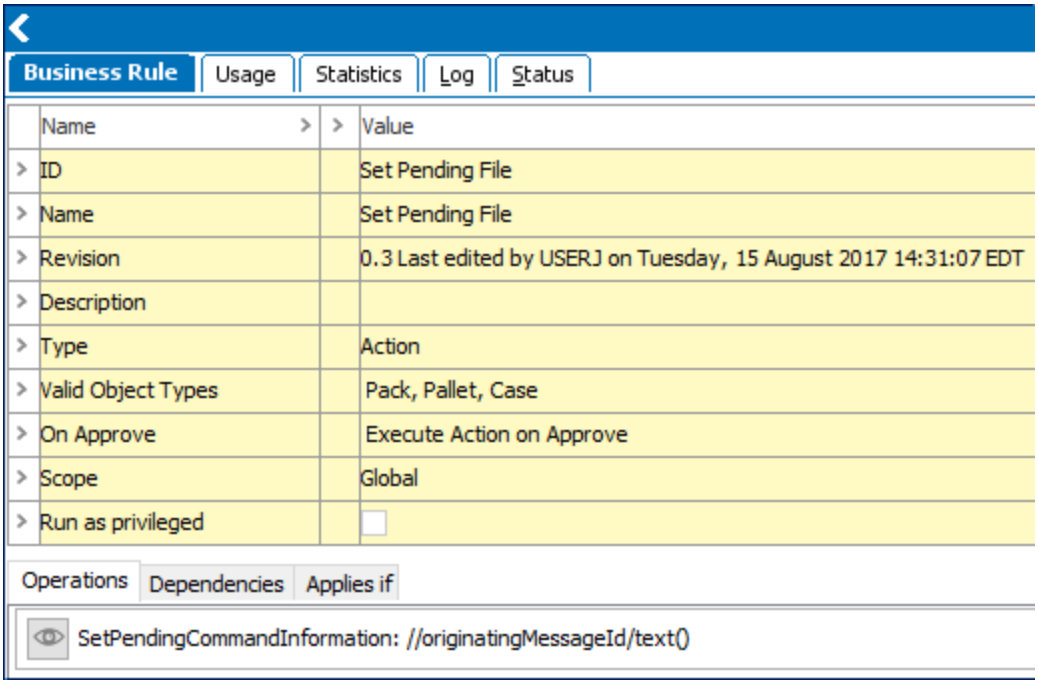
The 'Set pending command' operation for this business action tracks which command should be executed when the response messages have been received for the pre-command, the pending command will be stored in a description attribute that is valid for the registration object (see **GDSN Provider Component Model Elements**).

The Set Pending File business action is triggered via the Inbound format configuration with the Type Key 'catalogueResponsedocumentAcknowledgementitemADD'.

Originating Message ID: Specify the XPath for the Originating Message ID. The default value is //originatingMessageId/text().



The image below shows, example of the 'Set Pending File' business action.



Execute Pending File

This Business Action will execute the pending command that is stored in a description attribute on the registration object. For further information, see Set Pending File above.

The Execute Pending File business action is triggered via the Inbound format configuration with the Type Key 'gdsnItemRegistryResponsedocumentAcknowledgement'.

Business Rules in an Import Configuration

Business rules can be applied during the import process, enabling validation of data while it is being imported. Business actions allow changes to data during import, and can cause other changes as well.

The business rules available depend on your system setup. New business rules must be created in System Setup, which also shows if a business rule is used by an import configuration.

To determine if a business rule is used in an Import Configuration:

1. In System Setup > expand the Business Rules node.
2. Select the relevant business rule, and click the Usage tab.

   If a business rule is used by an import configuration, you cannot delete the rule.

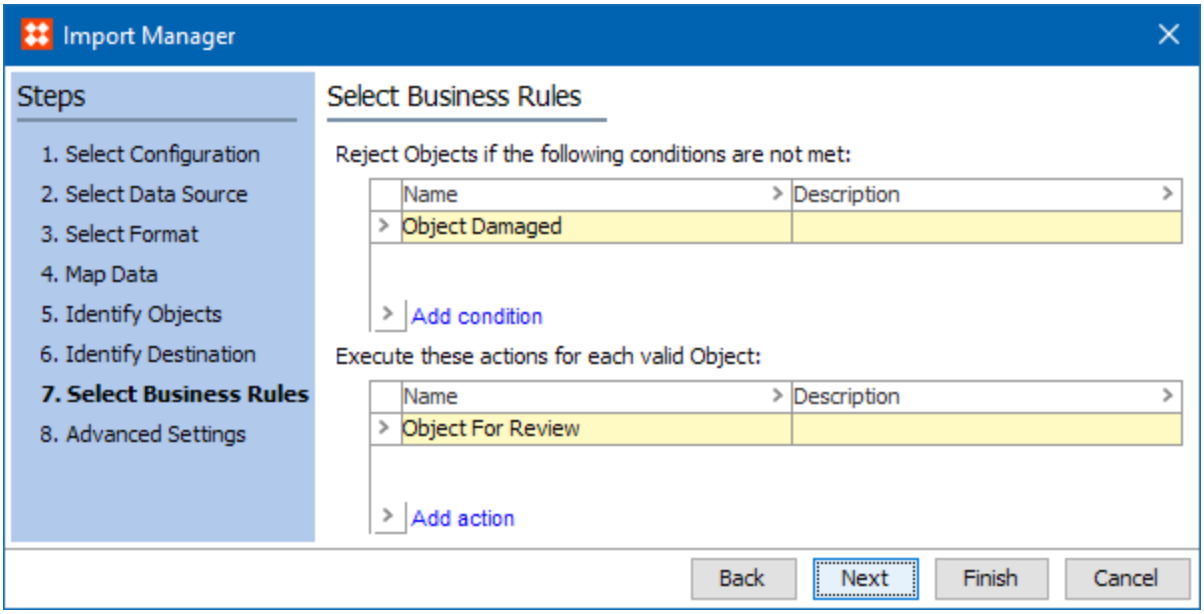
Conditions can be used to reject object creations and modifications to existing objects. Conditions are tested against data as it would appear in STEP assuming the import succeeded. Thus, the test is not limited to data in the import file. If a condition fails the import will be rolled back.

Actions will only be executed during import if all conditions evaluate to true. Thus, contrary to the approval process described in the **Business Rules on Approval** section, actions cannot make changes that make the conditions become true.

Add a Business Rule to an Import

Import Manager

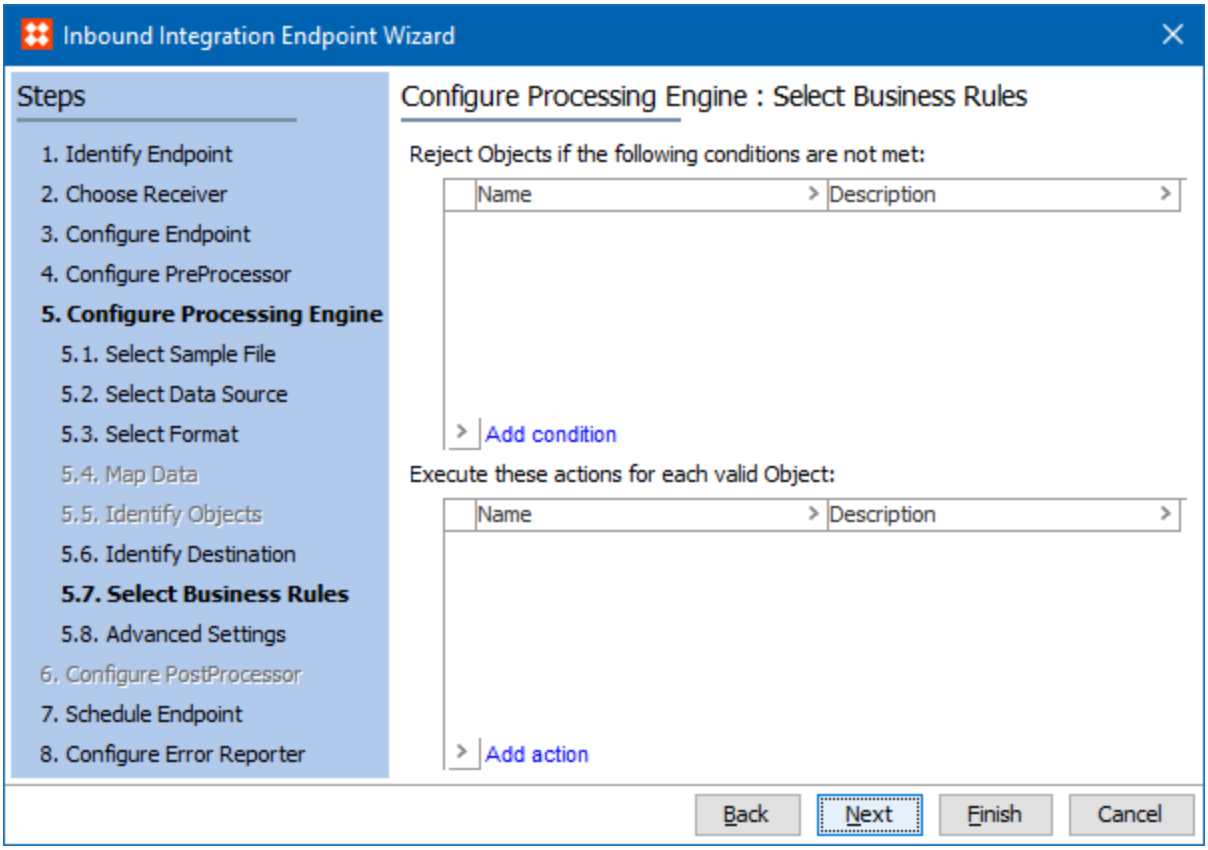
Business rules are configured on the Select Business Rules step of the Import Manager wizard. For more information, see **Import Manager - Select Business Rules** topic in the **Data Exchange** documentation.



The saved import configuration includes any selected business rules. For more information, see **Maintaining a Saved Import Configuration** topic in the **Data Exchange** documentation.

IIEP

Business rules are configured for the STEP Importer processing engine, on the Select Business Rules step of the IIEP wizard. The selected business rules can be modified by editing the IIEP configuration. For more information, see **IIEP - Configure Processing Engine** topic in the **Data Exchange** documentation.



Applying Business Rules

Business rules can be applied to products, classifications, entities, and assets. If business rules are applied to other object types, the rules are not evaluated.

The order of business rule execution for each imported object is: import, validation, actions, and auto-approve (if enabled).

- **Validation** - During the import, both changed and unchanged objects are validated against the business conditions. If the validation fails, the change is rejected. The changes are not imported, and an error message is logged in the Import Execution Report.

  Note: Objects are always validated, even if they are unchanged, to prevent Actions from running. The business rule itself can react on whether the object is detected as unchanged by the Import (e.g. to avoid rejecting unchanged data). Conditions cannot access the object as it was created prior to the import change.

- **Actions** - Actions primarily modify the imported objects. Ensure that the actions do not affect other objects. Actions are executed in the order specified in the import configuration.

For performance reasons, the Import Manager sometimes evaluates the same business rule twice for the same object.

Errors

If a business rule results in an error, the import skips the object. Errors are logged in the Execution Report. If too many errors occur, it may stop the import process. It is therefore important that business rules are created to handle expected exceptions.

Detecting Changes

JavaScript business rules access the imported objects using the 'Current Object' bind and the special 'Import Change Info'.

For each object, the import process detects if there are any changes. This information is provided to the business rules, and the business rules react based on this information.

In JavaScript the 'changeInfo' object contains two members:

- 'isUnmodified()' is true when object is not changed during the import.
- 'getChanges()' allows the business rule to retrieve a list of attribute IDs using 'getAttributes()' where the values have changed.

Business Rule Limitations

- Objects must be imported one at a time for the validation mechanism to work. It is therefore not possible to import nested STEPXML documents when you are using business rules.
- If one or more business rules are selected, the Import Manager uses domain mode.
- All selected business rules are run in sequence, from top to bottom.
- Changes are not detected for references and links. Objects with references in the import are therefore always reported as changed.
- Sometimes imported references are deferred if they depend on objects that have not been imported yet. As a consequence, the deferred parts of an imported object are not present while it is being validated. Therefore, use caution when using business conditions to validate references.
- When actions are used, the 'Approve Import Changes' on Advanced step is disabled - importer cannot automatically auto approve imported data because side effects from business actions are unknown. However it is still possible to approve imported objects via a business action.

Business Conditions in Web UI

Conditions can perform validation in Web UI while the user enters data, and notify immediately when a condition is not fulfilled. For example, to ensure that the attribute 'Sell Pack Depth' should be greater than or equal to 10 cm, a condition is created to evaluate the value entered and added to the appropriate screens. When the value is less than 10 cm, a warning is displayed as the user exists the field.

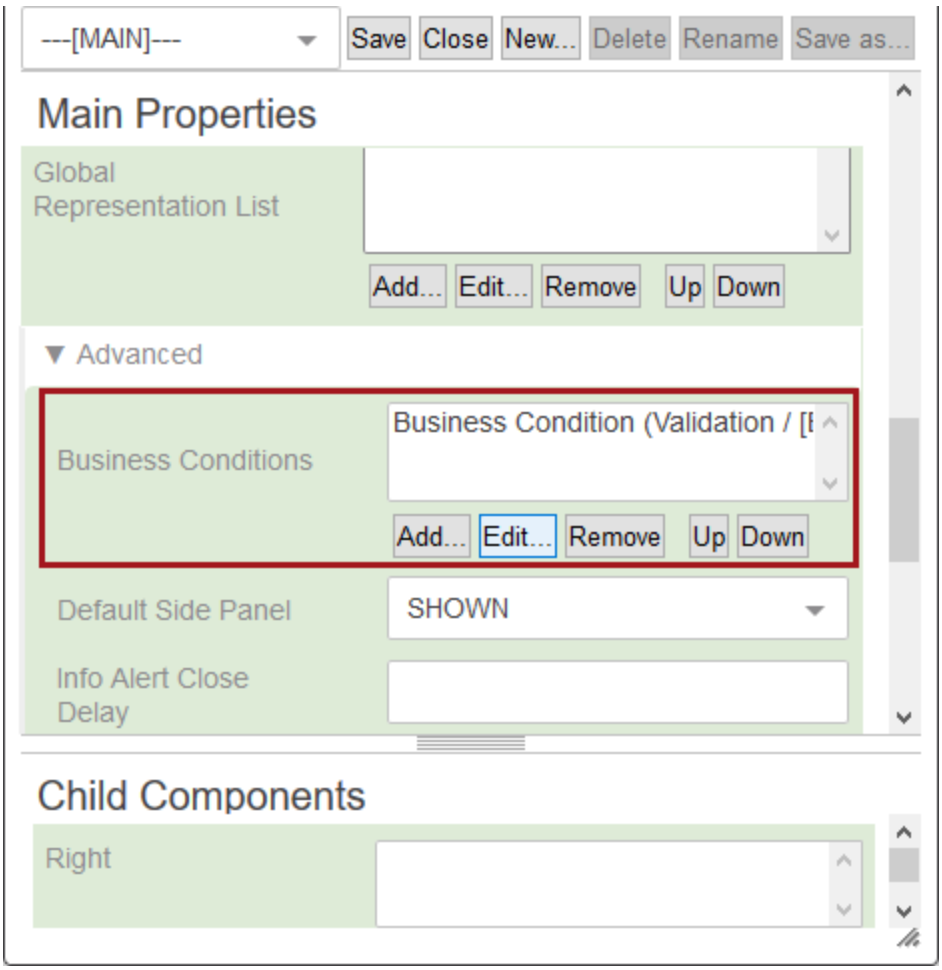
Conditions can also be used to filter Lists of Values. This function makes it impossible to select a value that violates the condition.

Configuring a Condition

1. Create a business condition in workbench.
2. Log in to Web UI, and click the gear wheel icon (⚙) to enter design mode.
3. Add the condition on the appropriate level (further defined below):
   - On the main screen level, the condition is checked on all screens in Web UI.
   - On the individual screen level, the condition is only checked on the current screen.

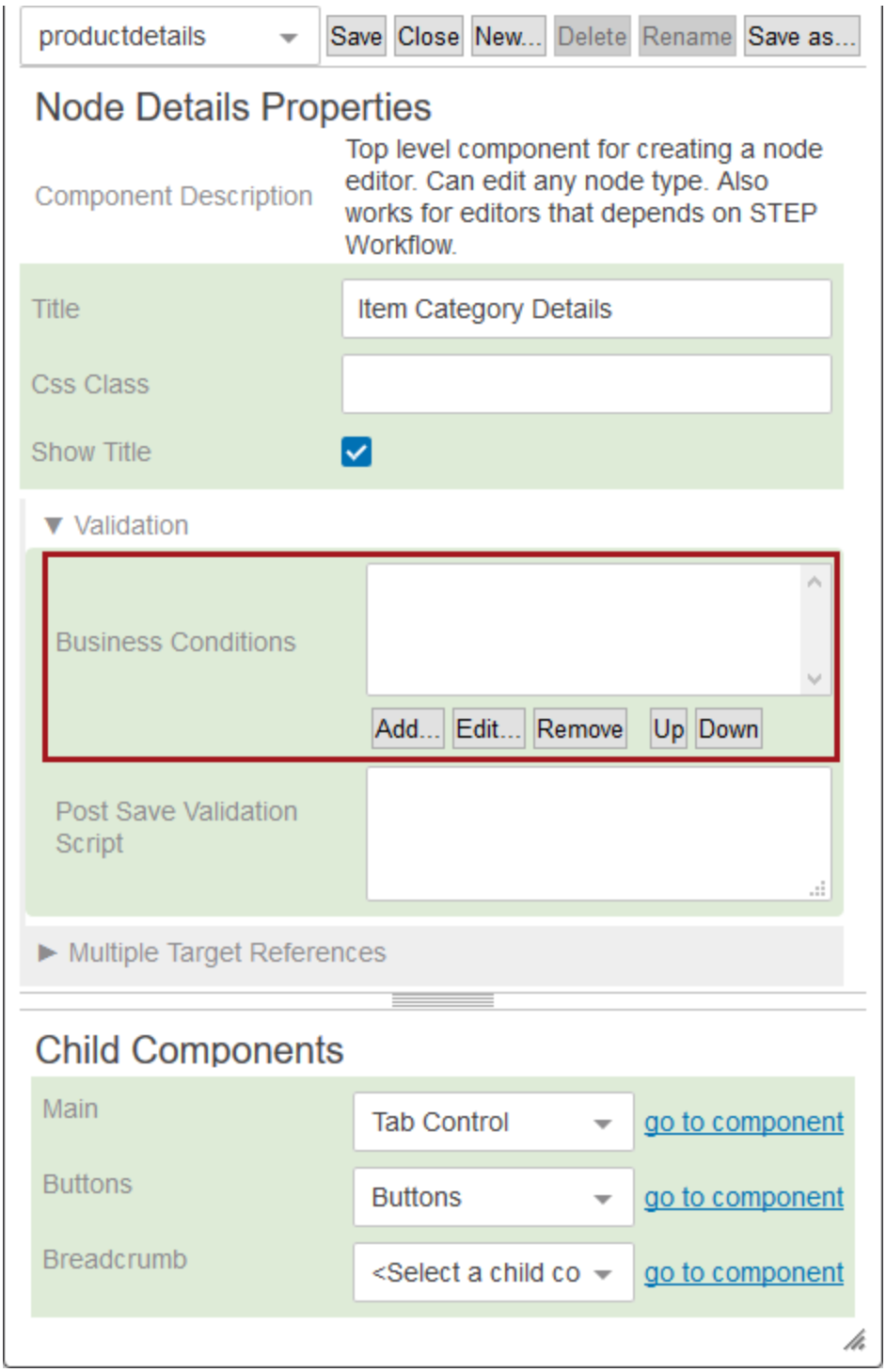
Main Screen Level

The following example shows the configuration of the Web UI screen called MAIN. This allows you to add, edit, remove, and reorder conditions for all screens. See the **Business Condition Properties** section below for details on adding or editing business conditions.



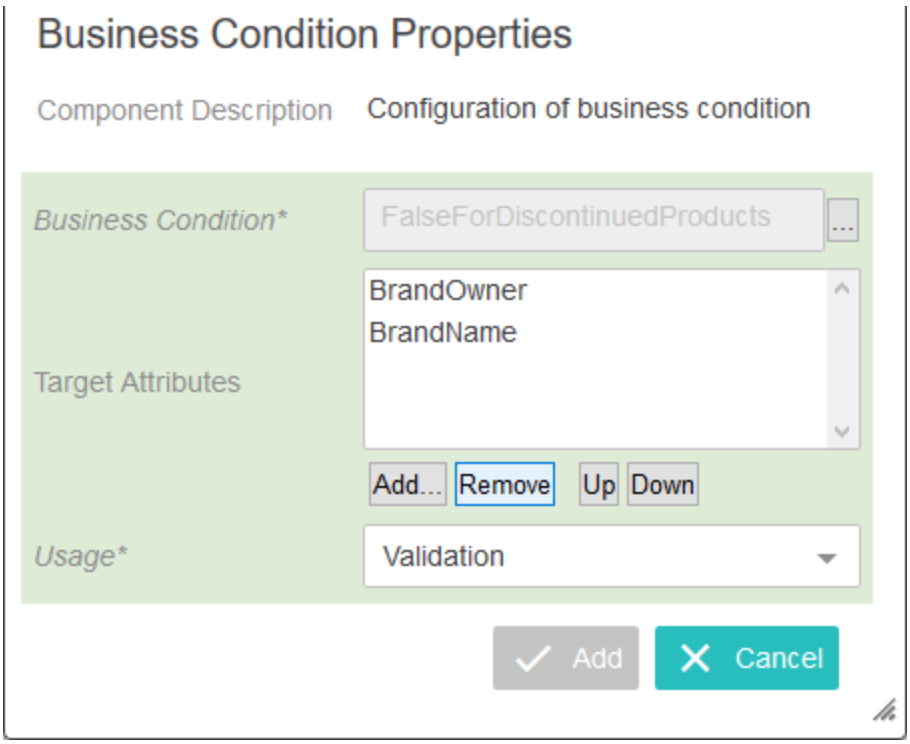
Individual Screen Level

The following example shows the configuration of a Web UI screen called ProductDetails. This allows you to add, edit, remove, and reorder conditions for this screen. See the **Business Condition Properties** section below for details on adding or editing business conditions.



Business Condition Properties

Click the **Add** or **Edit** button on the Business Conditions parameter to display the following configuration dialog.



Set the following options:

- **Business Condition**: Click the ellipsis button **(...)** to display the Select Nodes dialog, which inlcudes all business conditions available in STEP. Select the business condition to evaluate on the screen.

  Note: JavaScript conditions that use the current object bind cannot be used in Web UI.

- **Target Attributes**: Add the attributes that should be marked as wrong if the condition is not met.
- **Usage**: Select the desired usage from the following options:
  - **LOVFilter** filters illegal values out when the target attribute is a list of values attribute.
  - **Validation** marks the target attribute as 'wrong' when the condition fails.

Business Rules in Workflows

Business rules can be used in workflows to control the flow logic, make automatic changes to the objects being tracked with the workflow, or to notify users of task assignments, etc.

Business conditions are used to define the transitions that are legal in the workflow. Conditions can be tested on transitions between States, allowing and/or preventing the transition from being performed.

Business actions can be executed when:

- An object enters a state (On Entry)
- An object leaves a state (On Exit)
- A deadline is met and the deadline checker Background Process is run.
- A transition is performed (On Transition)

For more details and an example, see the **Business Rules and Workflows** topic in the **Workflows** documentation.

Additional information on using business rules in workflows, including detailed configuration instructions to accomplish specific use cases can be found throughout the **Advanced Workflow Topics** section of the **Workflows** documentation.

The public Java API available from JavaScript-based business rules offers functionality specifically for working with workflow logic–primarily the WorkflowableNode, WorkflowInstance, Task and Workflow interfaces. Using this functionality however requires in-depth knowledge about workflows.

Business Rule Common Use Cases

The following list is some of the most common uses for different types of business rules.

Object Approval

Business actions can be triggered by an approval on an object and can make changes to the object being approved, change other date in STEP, or external systems.

Business conditions can be triggered by an object approval and function as gates, either allowing or preventing approval of the object.

Imports / Inbound Integration Endpoint

Business actions can be executed in relation to objects being created or updated via the import.

Business conditions can serve as a filtering tool that can reject object creations as well as any importing data that modifies existing objects.

Note: Business actions will only be executed if all business conditions evaluate as true.

Exports / Outbound Integration Endpoint

Business actions can be evaluated for each generated event with the purpose of producing derived events.

Business conditions can be evaluated for each generated event, either allowing or preventing the event from being added to the associated event queue.

Workflows

Business actions are be evaluated executed when an object enters a state (On Entry), an object leaves a state (On Exit), a deadline is met, or when a transition is performed (On Transition).

Business conditions are tested on the transitions between workflow states, which will serve as a gate as to whether or not an object is moved between states.

Reusable Helper Functions

Business functions serve as a means of helping other business rules. Business functions are the only type of business rules that produce a result based on inputs without changing those inputs or any other data in STEP. Business functions allow users to write JavaScript logic in one place that can be called from other business rules.

Initial Set Up for Business Rules

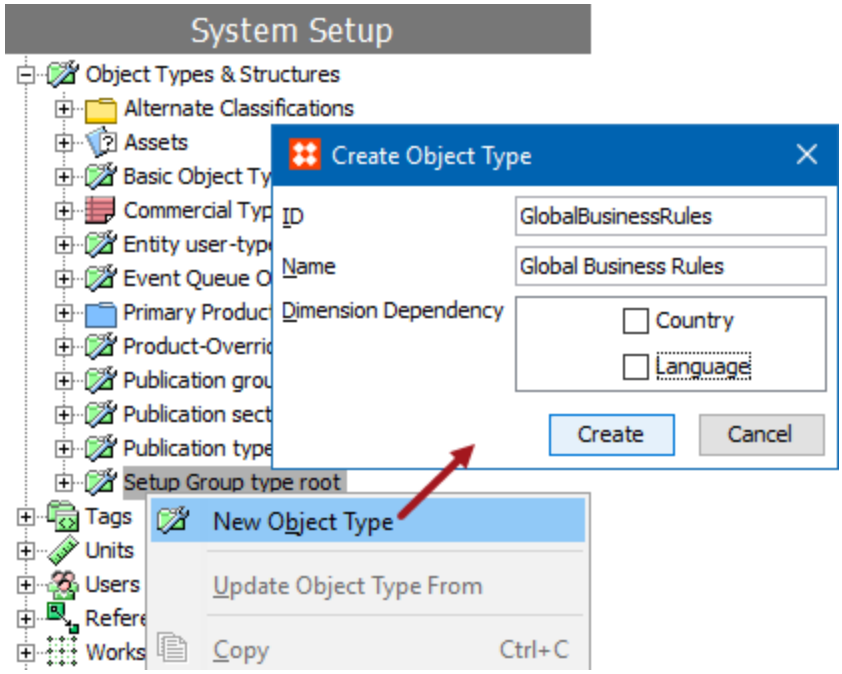
Before business rules can be created in STEP, a one-time configuration must be completed that includes:

- Create System Setup object types to hold the business rules (Global Business Rules, Business Actions, Business Conditions, Business Functions, and Business Libraries)
- Assign a Setup Group parent for each object type

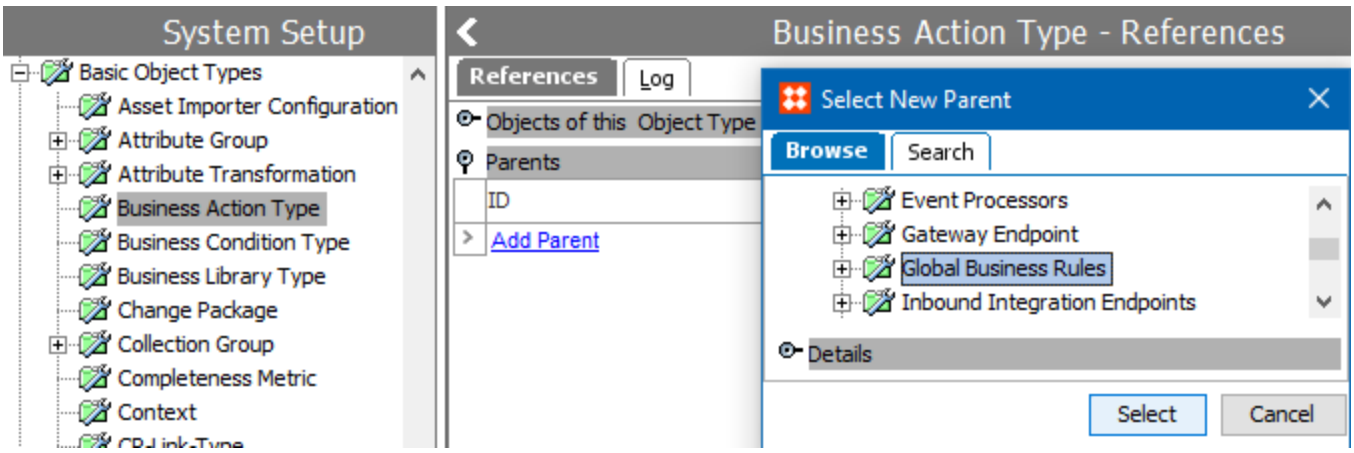
Each part of this setup is covered in the steps below. These steps can be carried out multiple times to separate different business rule types for organizational purposes.

Configuration

1. In System Setup, expand Object Types & Structures.
2. Right-click Setup Group type root and click **New Object Type** to display the Create Object Type dialog.

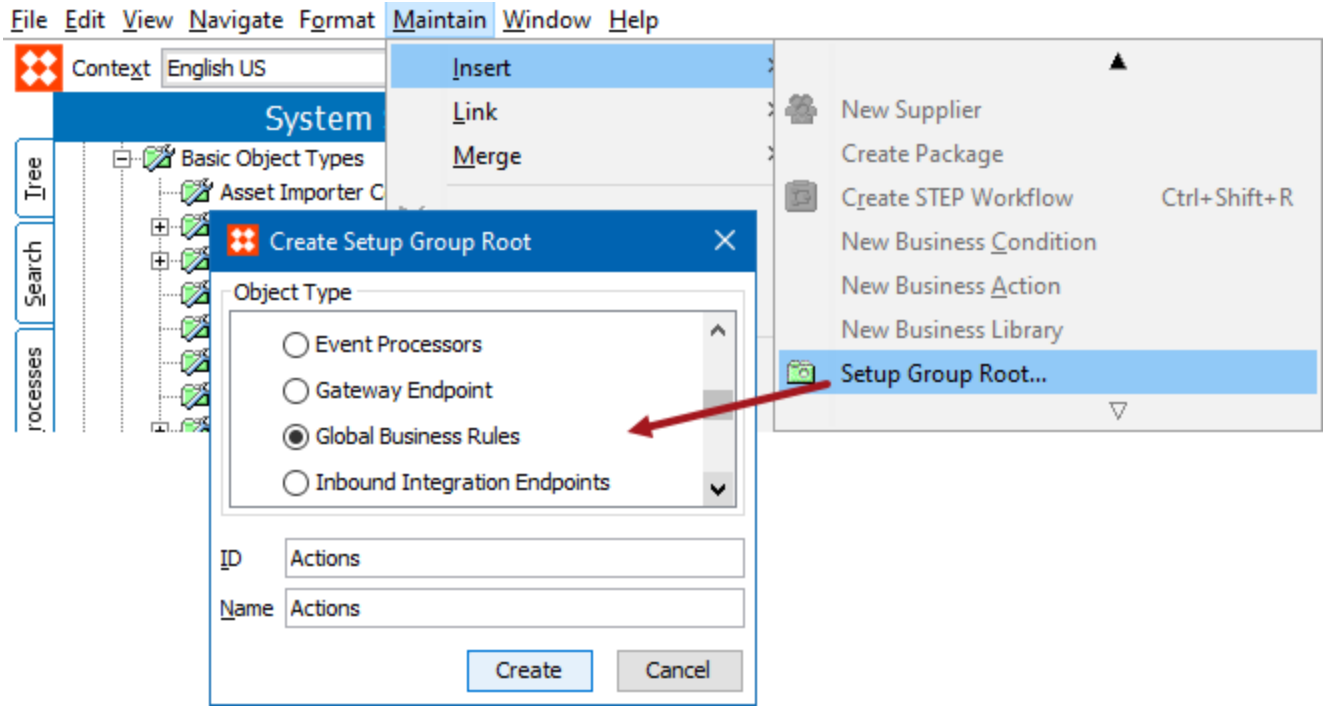


3. Enter an **ID** and a **Name** (such as Global Business Rules) and then click **Create**.
4. Optionally, if additional levels of organization are required to separate business conditions, business actions, and business libraries, right-click the object type you just created, and add the necessary additional object types. For example, name them Business Action Type, Business Condition Type, and Business Library Type.
5. In **Object Types & Structures**, expand Basic Object Types, and then select one of the Business Action Types you just created (Business Action Type, Business Condition Type, or Business Library Type).



- On the **References** tab, open the Parents flipper and click the **Add Parent** link.
- In the **Select New Parent** dialog, select the setup group you just created.
- Click the **Select** button to make it a valid parent.
- Repeat this step for each object type previously created.

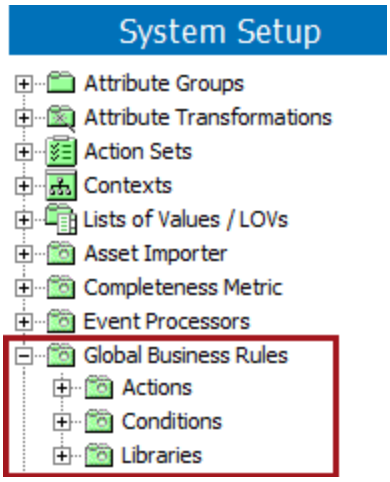
6. From the **Maintain** menu, point to **Insert**, and then choose **Setup Group Root** to display the Create Setup Group Root dialog.



- Select the setup group object type you just created, enter an **ID** and a **Name** for one of the object types you created, and then click **Create**. An instance of the object type is created in System Setup.

- Repeat this step for other object types (Business Action Type, Business Condition Type, Business Function Type, or Business Library Type).

The following hierarchy shows a setup group object type instance named Global Business Rules that is configured to hold actions, conditions, and libraries.

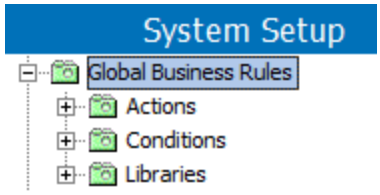


For the next step, creating business rules, see the **Creating a Business Rule or Library** topic.

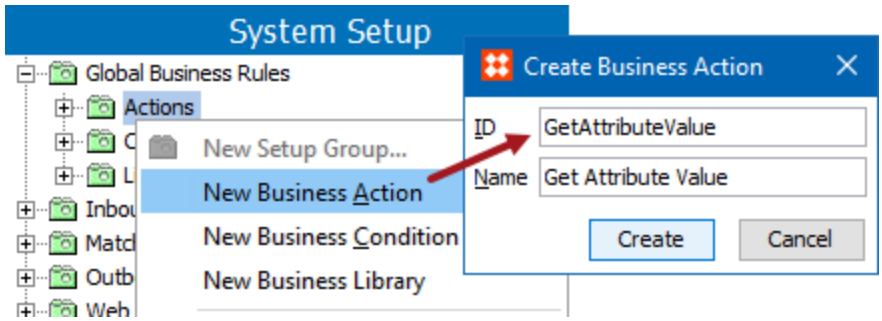
Creating a Business Rule or Library

After completing the one-time configuration, business actions, business conditions, business functions, and business libraries can be created in System Setup. For details about the required initial setup, see the **Initial Set Up for Business Rules** documentation.

1. In System Setup, identify the setup group object type instance configured to hold business rules. In this example, the Global Business Rules instance is configured to hold actions, conditions, functions, and libraries.



2. Right-click on the desired setup group to display a menu for creating the type(s) allowed.

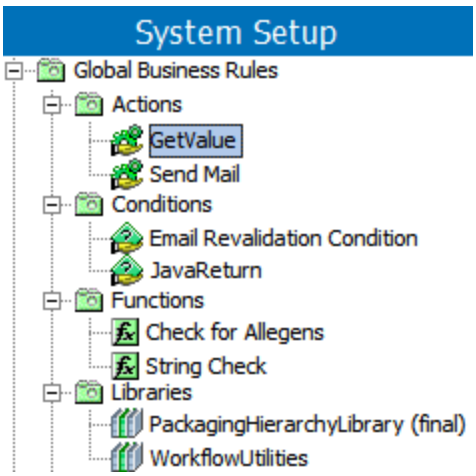


3. Enter an **ID** and a **Name** (such as Get Attribute Value) and then click the **Create** button. The selected business rule is created.

Note: Business rule objects are shown as read-only when viewed in the Approved Workspace.

4. To configure a new action, condition, or functions, use the steps defined in **Editing a Business Action, Condition, or Function,**, or to configure a new library, use the steps defined in **Editing a Business Library**.

The following hierarchy shows a setup group object type instance named Global Business Rules that is configured to hold actions (for example, Get Attribute Value), conditions (for example, Object Damaged), and libraries (for example, Workflow Utilities).

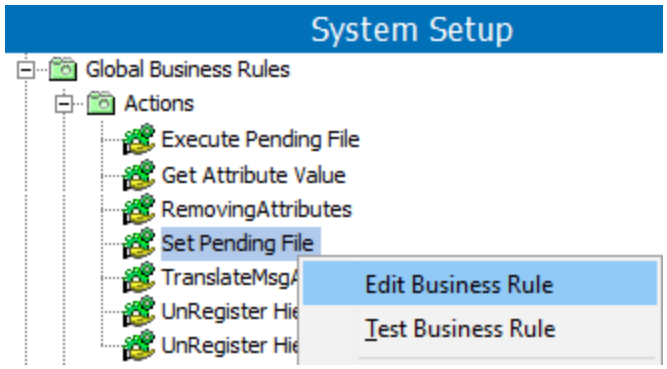

System Setup
Global Business Rules
Actions
GetValue
Send Mail
Conditions
Email Revalidation Condition
JavaReturn
Functions
Check for Allegens
String Check
Libraries
PackagingHierarchyLibrary (final)
WorkflowUtilities

Editing a Business Rule

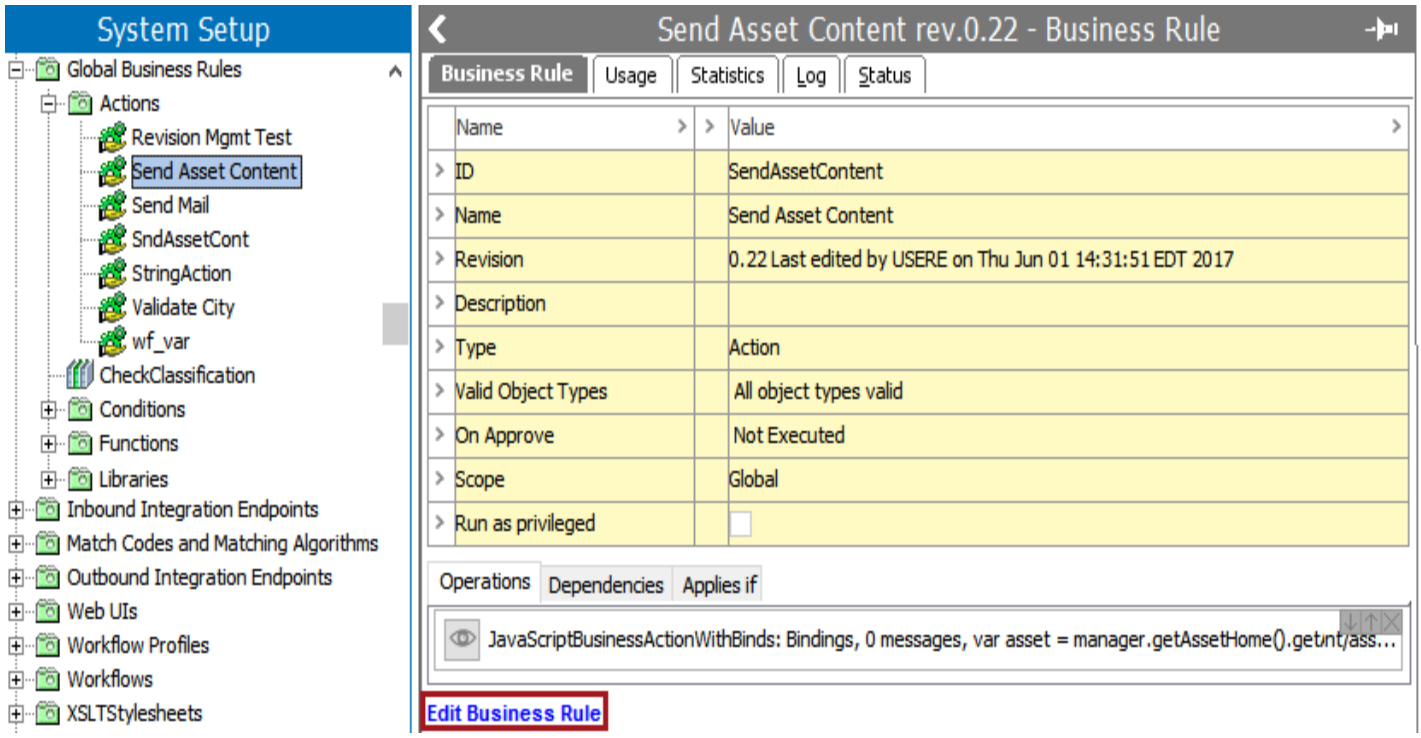
The Business Rule editor allows you to edit settings for a business rule and handles Global and Local business rules differently. For more information, see the **Global and Local Business Rules** section of the **Business Rules** documentation.

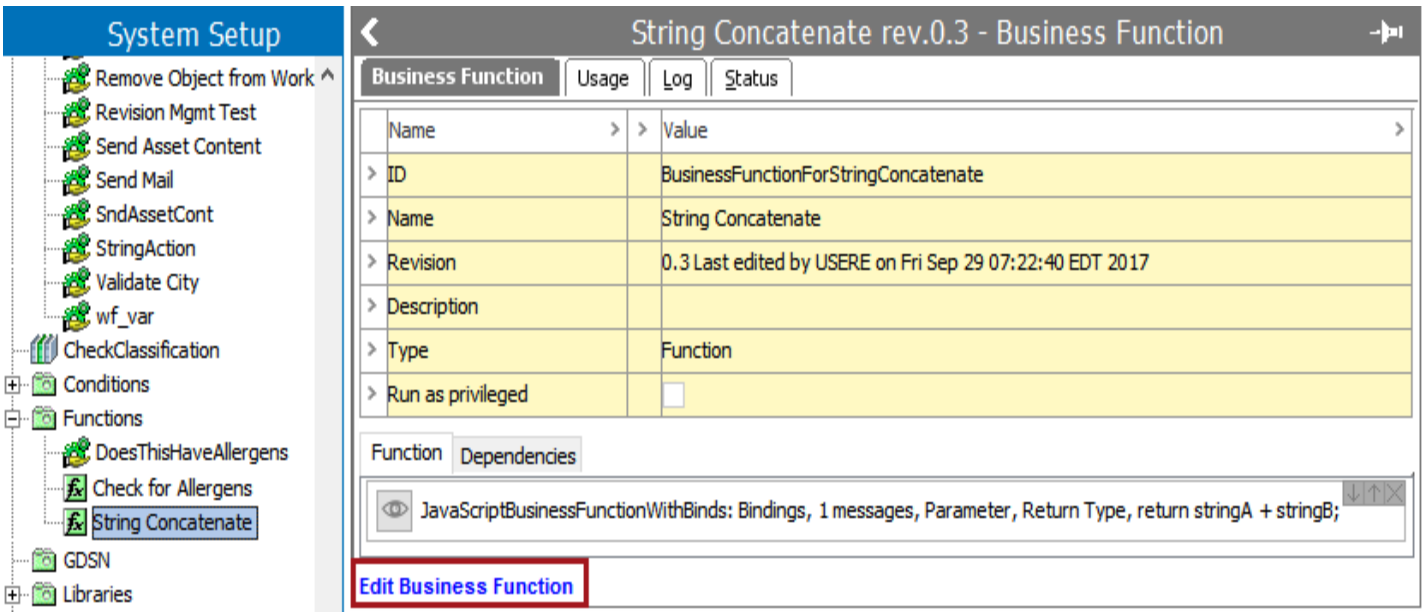
There are two ways to open the Business Rule editor in System Setup:

- Select the business rule, right-click, and select **Edit Business Rule** from the menu.

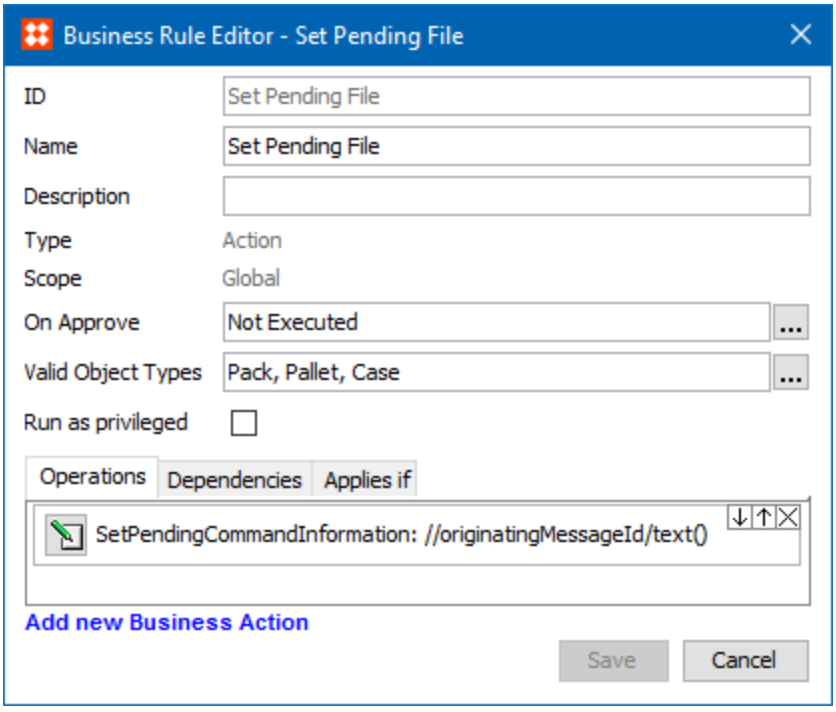


- Select the business rule and select the **Edit Business Rule** link or the **Edit Business Function** link on the Business Rule tab.

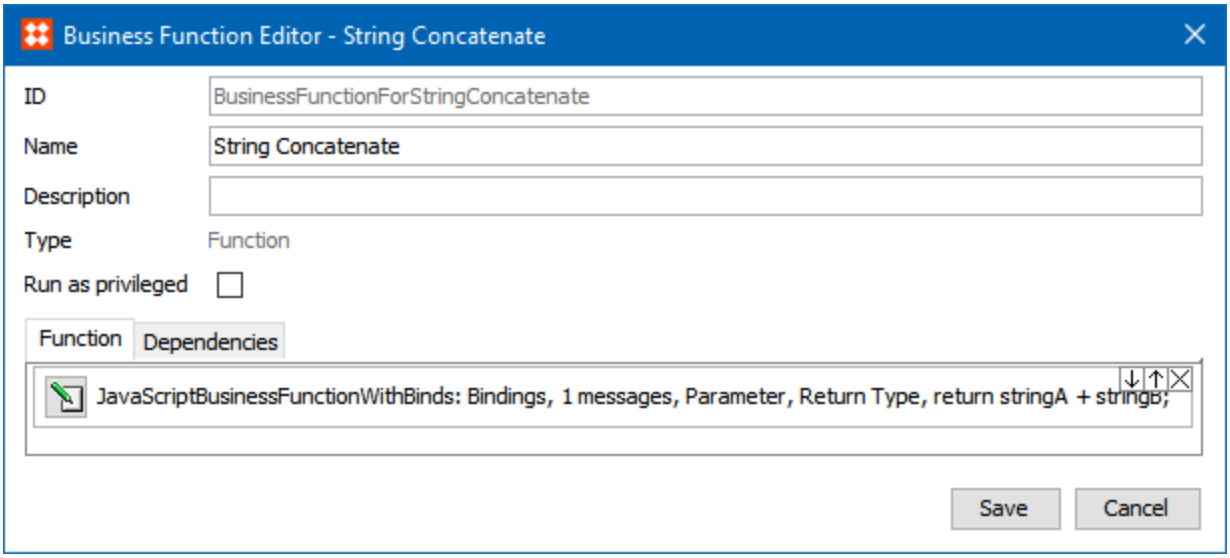




The Business Rule Editor for actions and conditions allows you to modify most parameters on a business rule. However, the ID, Type, and Scope, which are all shown with read-only text below, cannot be edited.



The business function editor has few options to specify since all of the work is done via the included JavaScript editor. For more information on setting up business functions, see **Calling a Business Function from a JavaScript Business Rule** topic in the **Business Function** documentation.



Note: Global business rules can also be edited via workflows where the same global Business Rule Editor is used. For more information, see the **Business Rules and Workflows** topic in the **Workflows** documentation.

The Business Rule Editor parameters are defined below and can be set on a condition, an action, or a function business rule:

ID Parameter

Read-only display to indicate the business rule ID assigned while creating the business rule.

Name Parameter

Displays the business rule name assigned while creating the business rule and can be modified.

Description Parameter

Displays the business rule description assigned while creating the business rule and can be modified.

Type Parameter

Read-only display to indicate the type of business rule being edited. This can be action, condition, a function, or a library.

Scope Parameter

Read-only display to indicate if the business rule is global or local.

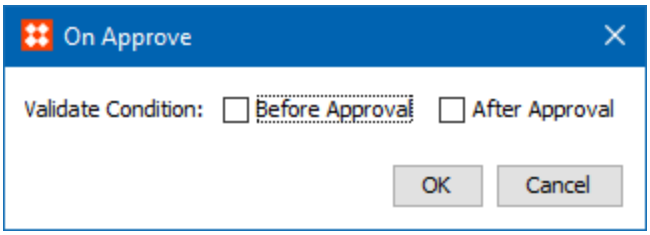
On Approve Parameter

In the On Approve field, specify if the business rules should be run during an approval process. This only applies to global business rules. For local business rules, the setting of the workflow determine this parameter.

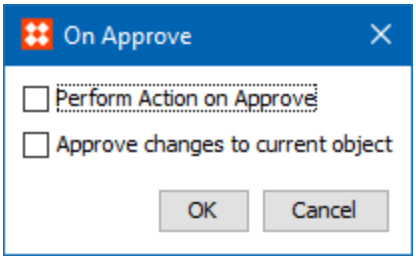
For details on these options, see **Business Rules on Approval**.

Click the ellipsis button **(...)** to display the On Approve dialog. The 'Valid Object Types and 'Applies if' settings determine if the condition will be tested when attempting to approve a revisable object.

- On conditions, specify to perform the validations before or after approval.



- On actions, specify to execute the action on approve, and whether to approve changes as well.



Valid Object Types Parameter

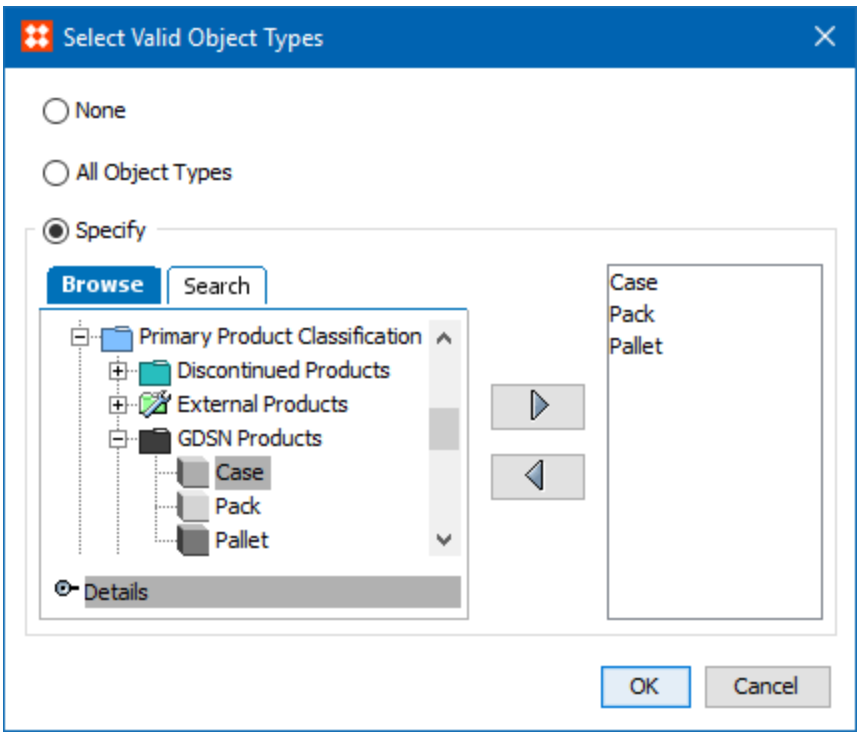
Specify the object type or types for which the business action or condition is valid. This only applies to global business rules. For local business rules, the setting of the workflow determine this parameter.

Note: Business functions do not require a Valid Object Type.

Click the ellipsis button **(...)** to display the Select Valid Object Types dialog. Select one of the following options:

- **None** - The business rule will not be executed or tested on any object type's instances. Use this option to deactivate the business rule.
- **All object types** - The business rule will be executed or tested on all object type's instances. It is not recommended to use this option since the rule will be executed regardless of the object type.
- **Specify** - The business rule will be executed or tested on the specified object type's instances. This is the common setup since the rule will be executed only on specified object type instances.

  Use the Browse or Search tab to identify and select the desired object type, Click the move right button ( ) until all object types are displayed in the panel on the right. Use the move left button ( ) to remove an object type from the list.

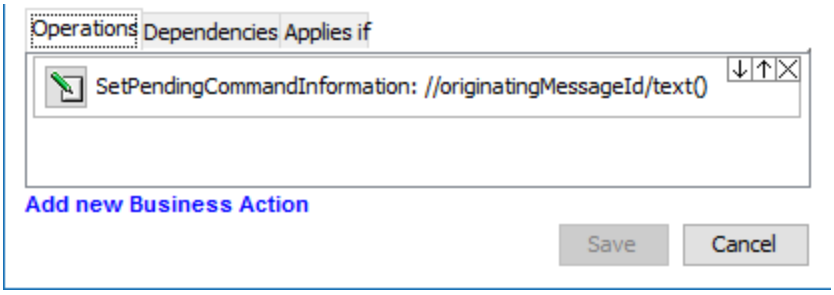


Run as Privileged Parameter

Check the **Run as privileged** checkbox to allow the business rule to execute functions that the user is not authorized to execute.

Operations Tab

The Operations tab allows you to add or modify the actions or conditions that apply to a business rule.



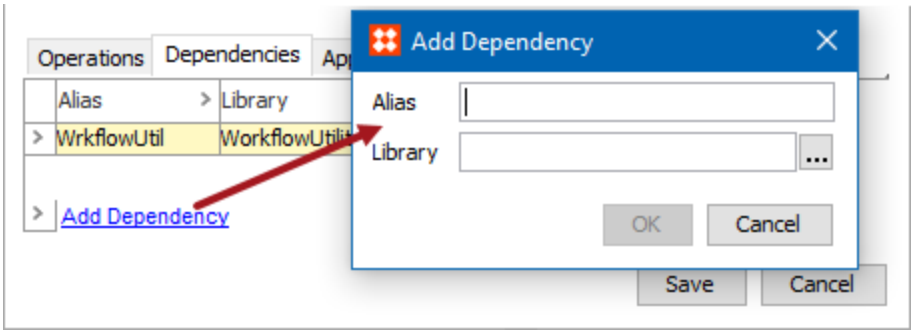
- To add a new condition or action, click the **Add new Business Action** or **Add new Business Condition** link. A new row is added to the list.
- To edit an condition or action, click the **Edit Operation** ( ) button.
- To change the order of the business rules, use the up or down arrow buttons.
- To delete a business action or condition, click the X button.

For more information about actions, conditions, and functions, see the **Business Actions** topic, the **Business Conditions** topic, and the **Business Functions** topic.

Dependencies Tab

The Dependencies tab allows you to reference business rules to business libraries.

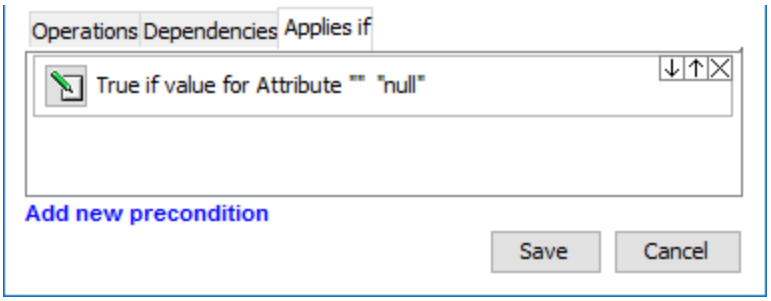
- To add a dependency, click the **Add Dependency** link, and type an alias. Then click the ellipsis button **(…)** and select a business library that contains JavaScript, and click **OK**.



For more information, see **Business Libraries**.

Applies If Tab

The **Applies if** tab allows you to specify a precondition (a condition) to be evaluated before the business rule is applied.



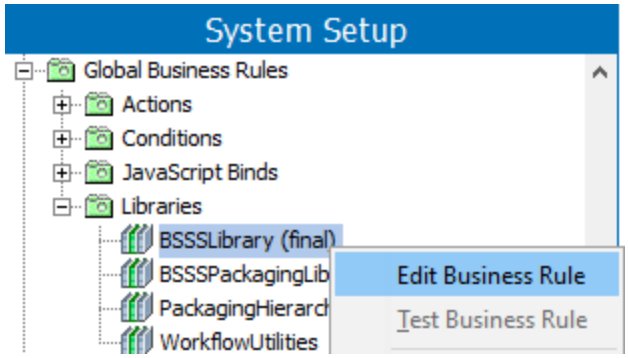
- To add a new condition, click the **Add new precondition** link. A new row is added to the list.
- To edit a condition, click the **Edit** ( ) button.
- To change the order of the conditions, click the up or down arrow buttons.
- To delete a condition, click the **X** button.

For more information about conditions, see the **Business Conditions** topic.

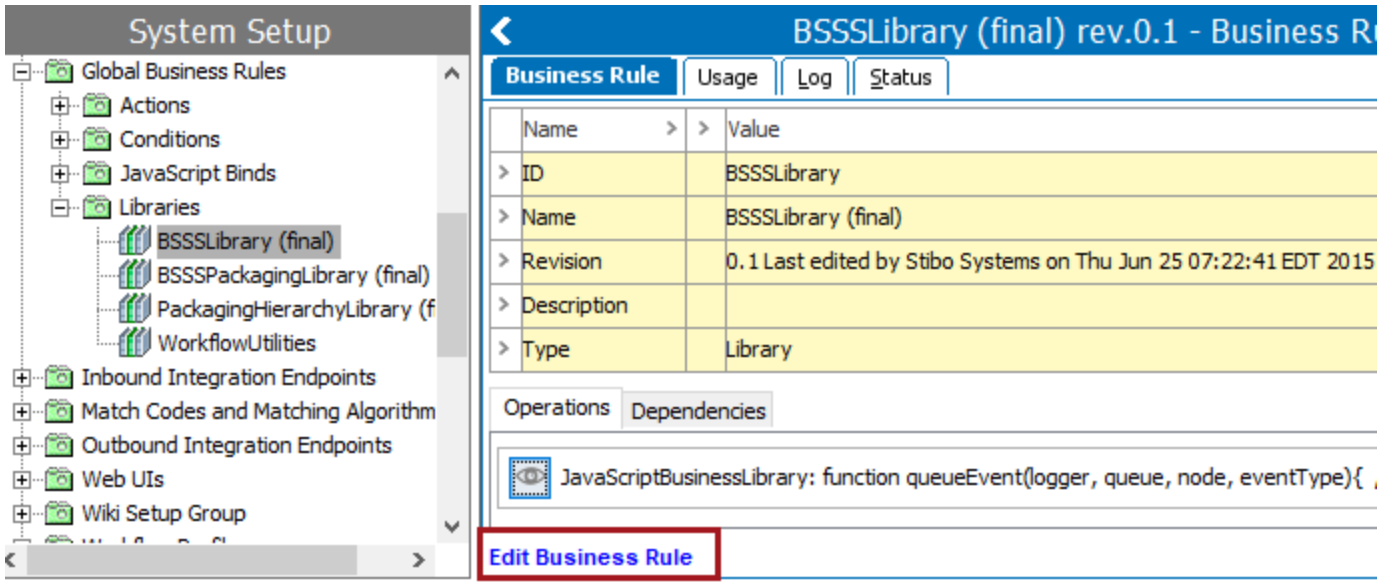
Editing a Business Library

There are two ways to open the Business Rule editor in System Setup:

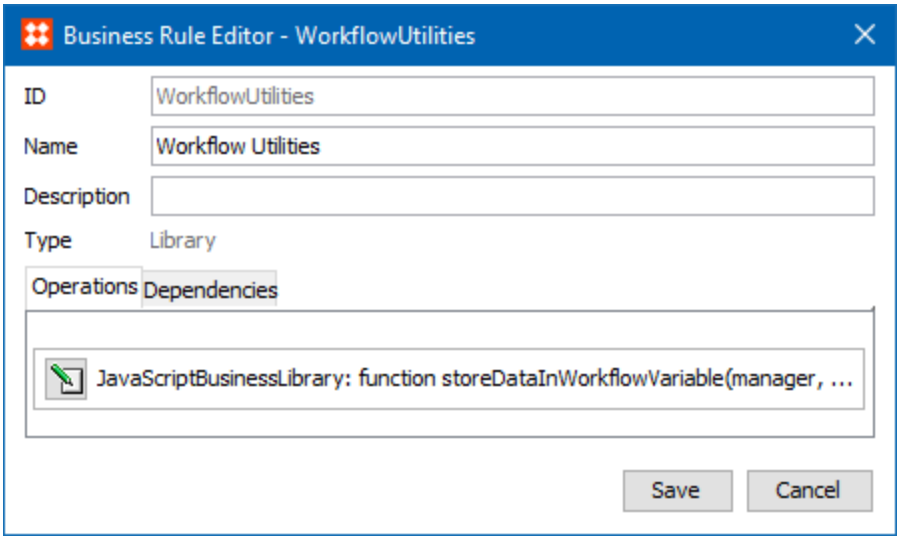
- Select the business rule, right-click, and select **Edit Business Rule**



- Select the business rule and select the **Edit Business Rule** link on the Business Rule tab.



The Business Rule Editor for libraries allows you to modify most parameters on a business rule. However, the ID, Type, and Scope, which are all shown with read-only text below, cannot be edited.



The following parameters can be set on a library business rule: Name, Description, Operations, and Dependencies.

ID Parameter

Displays the business rule ID assigned while creating the business rule and cannot be modified.

Name Parameter

Displays the business rule name assigned while creating the business rule and can be modified.

Description Parameter

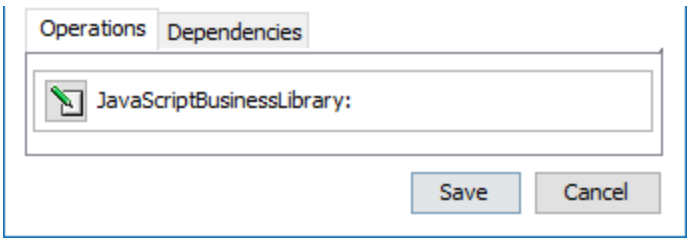
Displays the business rule description assigned while creating the business rule and can be modified.

Type Parameter

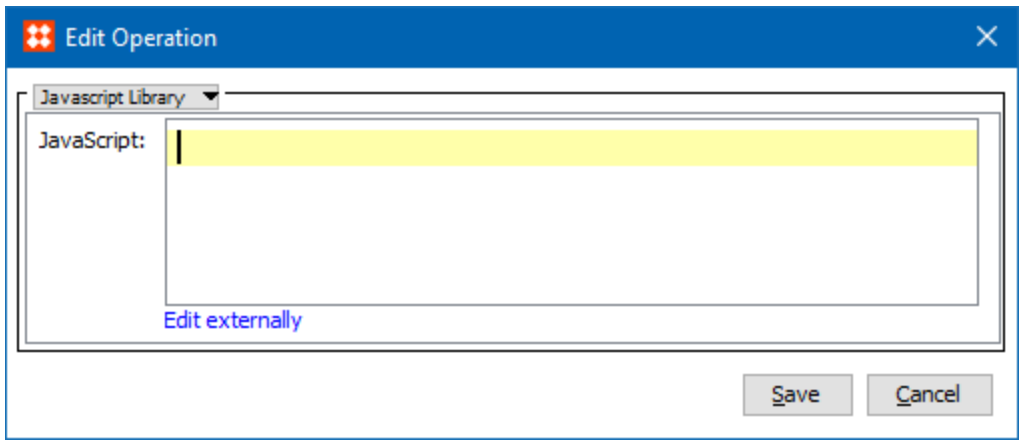
Displays the type of business rule being edited. This can be action, condition, a function, or a library.

Operations Tab

The Operations tab allows you to modify the JavaScript code for the business rule. By default, a JavaScriptBusinessLibrary operation is added to a new business library object. All required JavaScript for the library will be added to this operation.



- To edit the library, click the **Edit Operation** ( ) button to display the Edit Operation dialog.

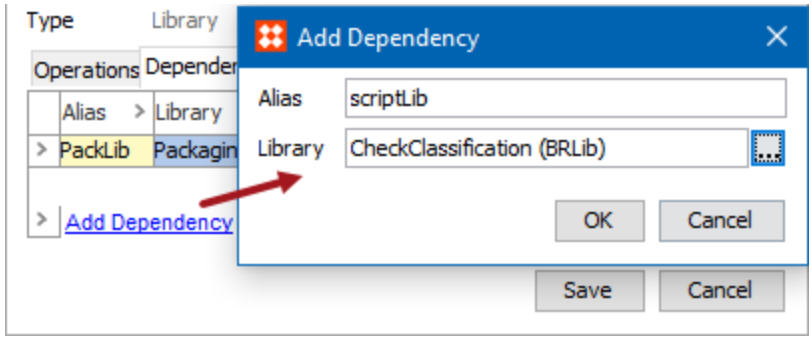


Dependencies Tab

The Dependencies tab allows you to to reference business rules to business libraries.

Important: A dependency is required to the library from each business action, business condition, or library where you want to be able to use the library.

- To add a dependency, click the **Add Dependency** link, and type an alias. Then select a business library that contains JavaScript. Click **OK** to add it to the list of Alias / Library entries.



Once the dependency has been declared, functions in the library can be called from JavaScripts.

In the following example, an action has a reference to the library with the alias scriptLib and the function CheckClassification.

```
if(scriptLib.CheckClassification(node)) {
 //Action to perform in case "CheckClassification" returns true
}
```

Testing a Business Rule

Typically, after creating a business rule, you can test it.

Important: Business rules using context-specific JavaScript binds cannot be tested using this method. Therefore, JavaScript that uses the import change info or approve context binds cannot be tested. Similarly, business actions where the Execute JavaScript function is used to create parametrized business rules cannot be tested.

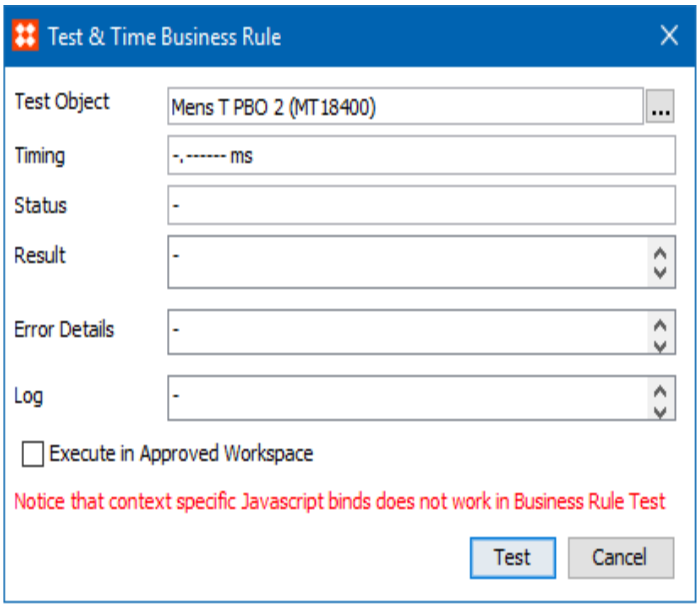
For an overview of the decision points involved in running a business rule, see the chart in the **Running a Business Rule** topic.

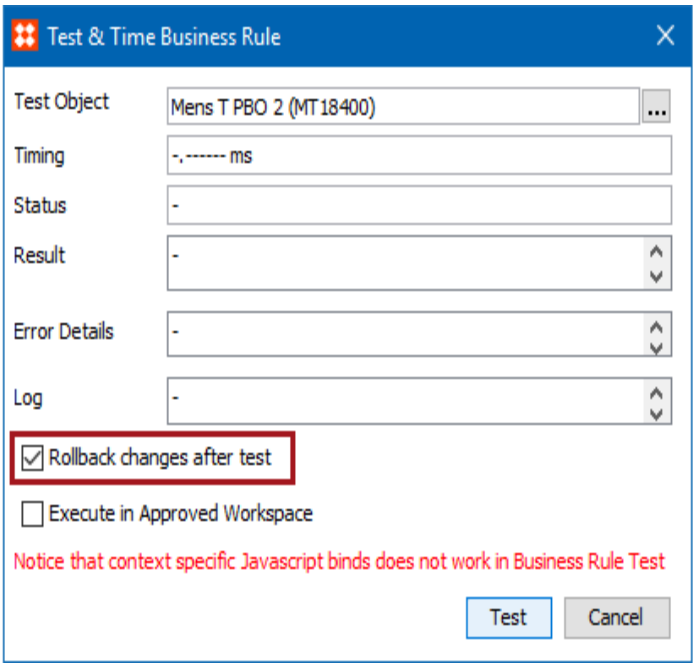
Monitoring Business Rules

Using the **BusinessRule.Warning.Threshold** configuration property in the sharedconfig.properties file, allows you to specify a threshold in milliseconds for business action execution. If it takes longer to execute or test a given business action, a warning is posted in the main STEP log file available from the STEP System Administrator button on the WebStart. For information on the log, see the **Logs** topic in the **Administration Portal** documentation.

Test a Business Rule

1. In System Setup, expand the business rules node, and then select the business condition or action that you want to test.
2. Right-click the condition or action, and then choose **Test Business Rule** to display the Test & Time Business Rule dialog.

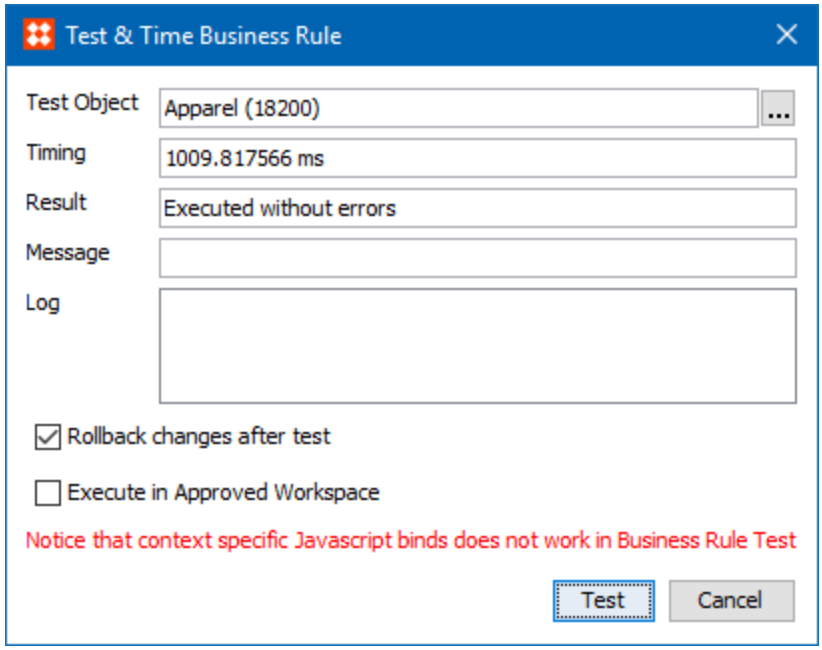




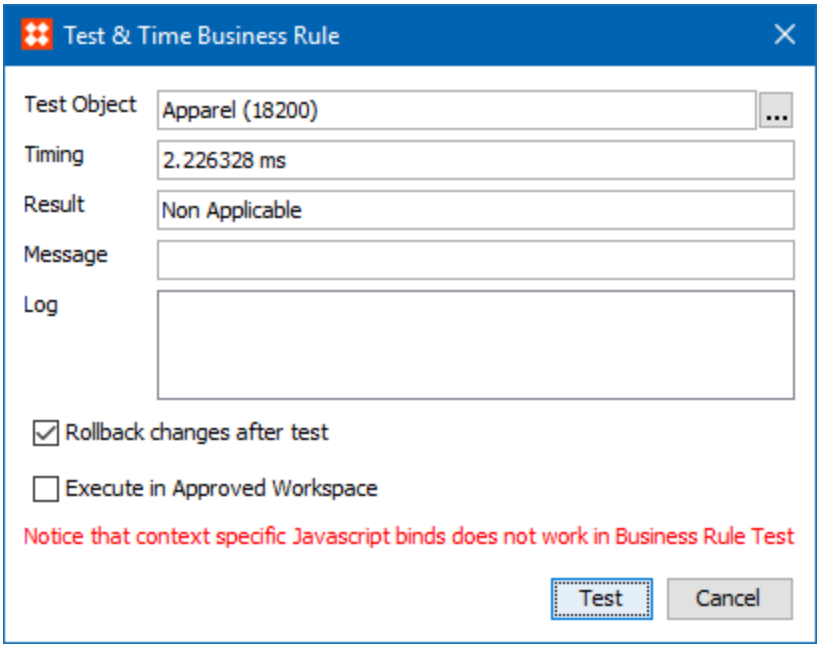
3. In the **Test Object** parameter, click the ellipsis button **(…)** to display the Select Test Object dialog. Use the Browse or Search tab to identify and select the object used to test the business rule.
4. When testing an action, the **Rollback changes after test** checkbox is available. Check the checkbox if you want the changes caused by the action to be rolled back after the action has been executed.
5. The **Execute in Approved Workspace** is unchecked by default and the test is performed in the Main Workspace. Check the checkbox if you want the action to be performed on the selected object in the Approved Workspace.
6. Click the **Test** button.
   - **Result** for action - When testing an action, this parameter displays either **executed without errors** or **failed**. If the action fails, an error message is displayed. See the **Action Test Result Examples** section below.
   - **Result** for condition - When testing a condition, this parameter displays true, false, or failed. If a condition evaluates to false, the reason is displayed. If a condition fails, an error message is displayed. See the **Condition Test Result Examples** section below.
   - **Timing** - This parameter displays the time taken to execute the business rule on the selected node.
   - **Log** - This parameter displays log message for debug and/or testing purpose. In JavaScript, use **logger.info("Message");** in the code to log a message.

Action Test Result Examples

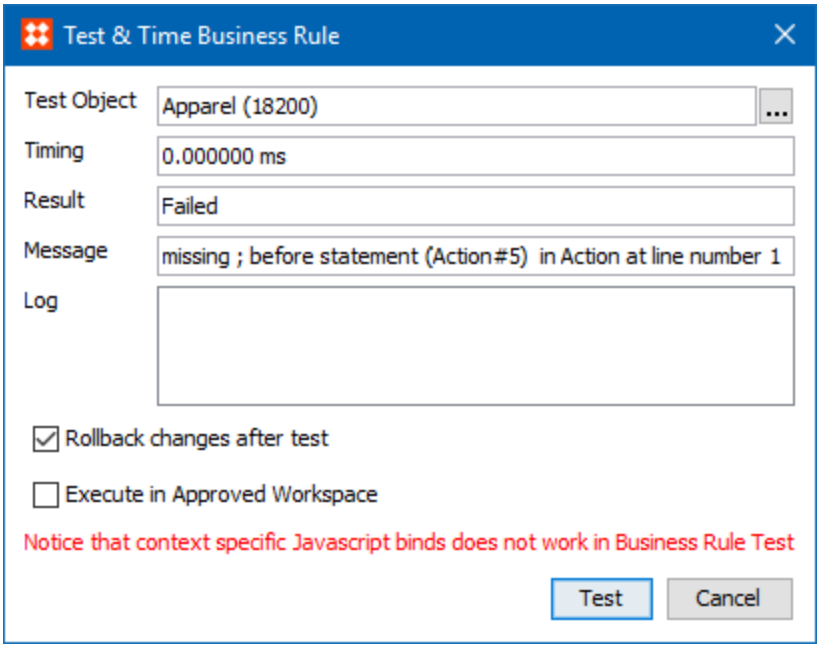
- **Executed without error** - This result is displayed when the business action is executed successfully as illustrated in the below image.



- **Non Applicable** - This result is displayed when the test object is not match with the business action valid object types as illustrated in the below image.

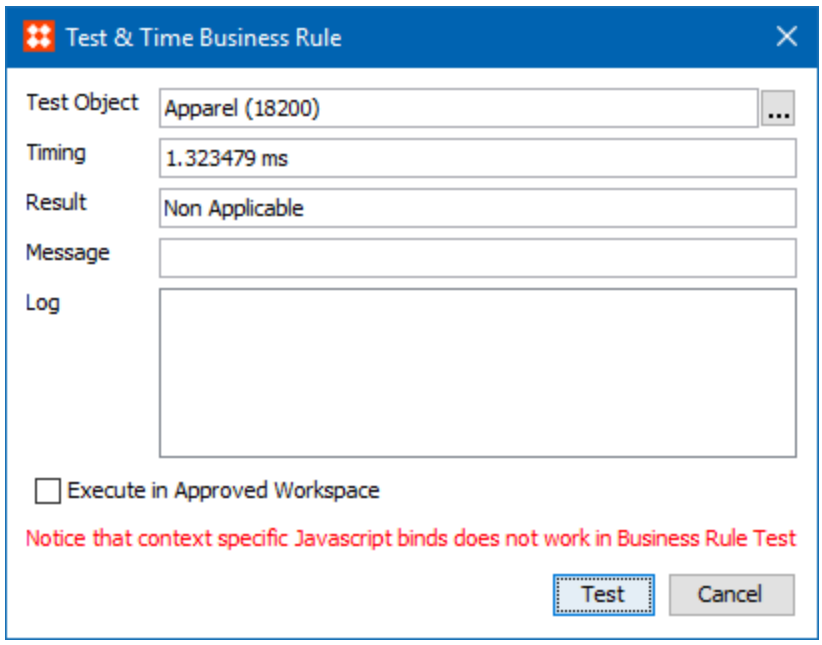


- **Failed** - This result is displayed when the business action has invalid methods or syntax or function as illustrated in the below image.

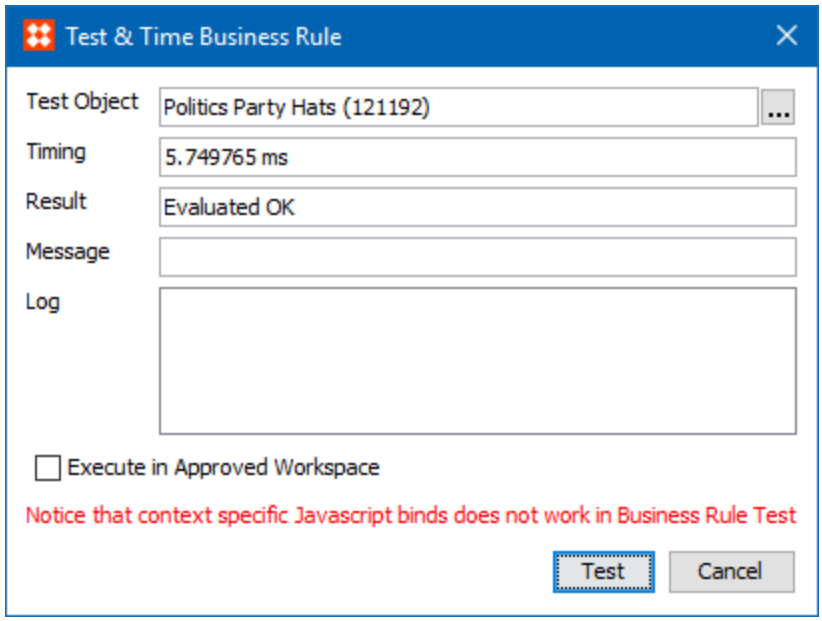


Condition Test Result Examples

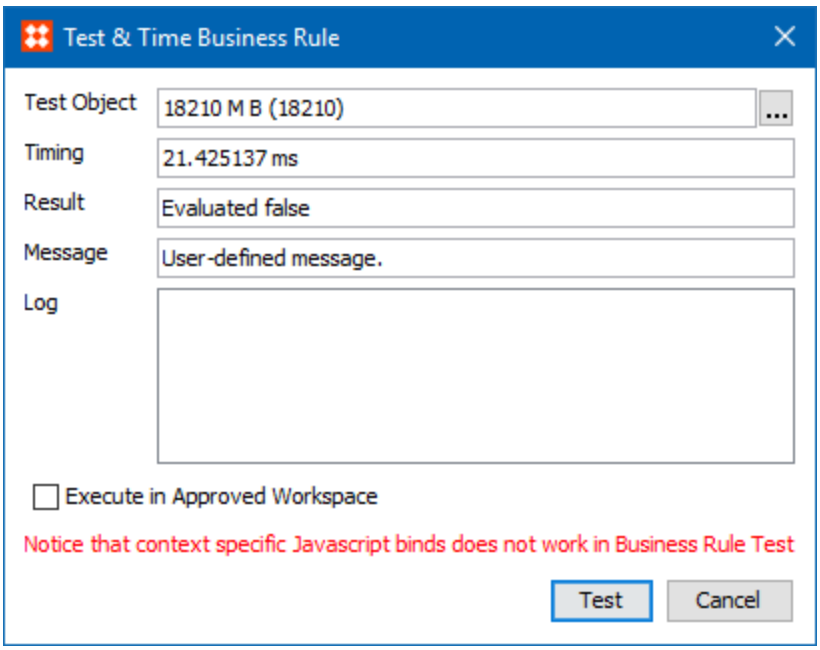
- **Non Applicable** - This result is displayed when the test object is not match with the business condition valid object types as illustrated in the below image.



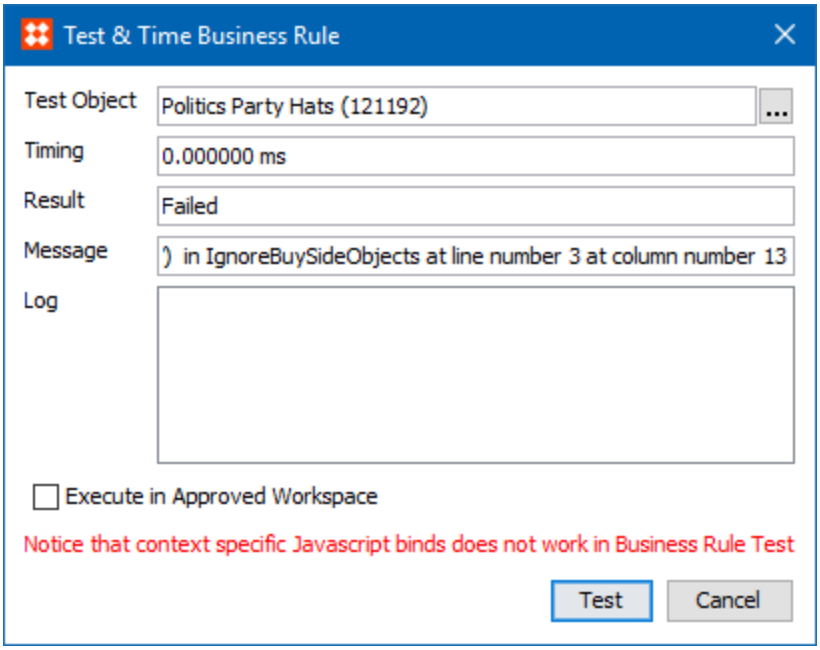
- **Evaluated OK** - This result is displayed when the business condition is true as illustrated in the below image.



- **Evaluated false** - This result is displayed when the business condition fails. If configured, a user-defined message can be displayed in the message field as illustrated in the below image. For more information on the message, see the **Localized Messages for JavaScript Business Rules** topic.



- **Failed** - This result is displayed when the business condition has invalid methods, syntax, or function as illustrated in the below image.



Maintaining a Global Business Rule

The settings for business rules are specified in the Business Rule editor. The following describes how to work with global business rules. For information about local business rules, see the **Business Rules in Workflows** topic.

In System Setup, expand the business rules node, then select the relevant business rule. This opens the business rule editor in a read-only mode.

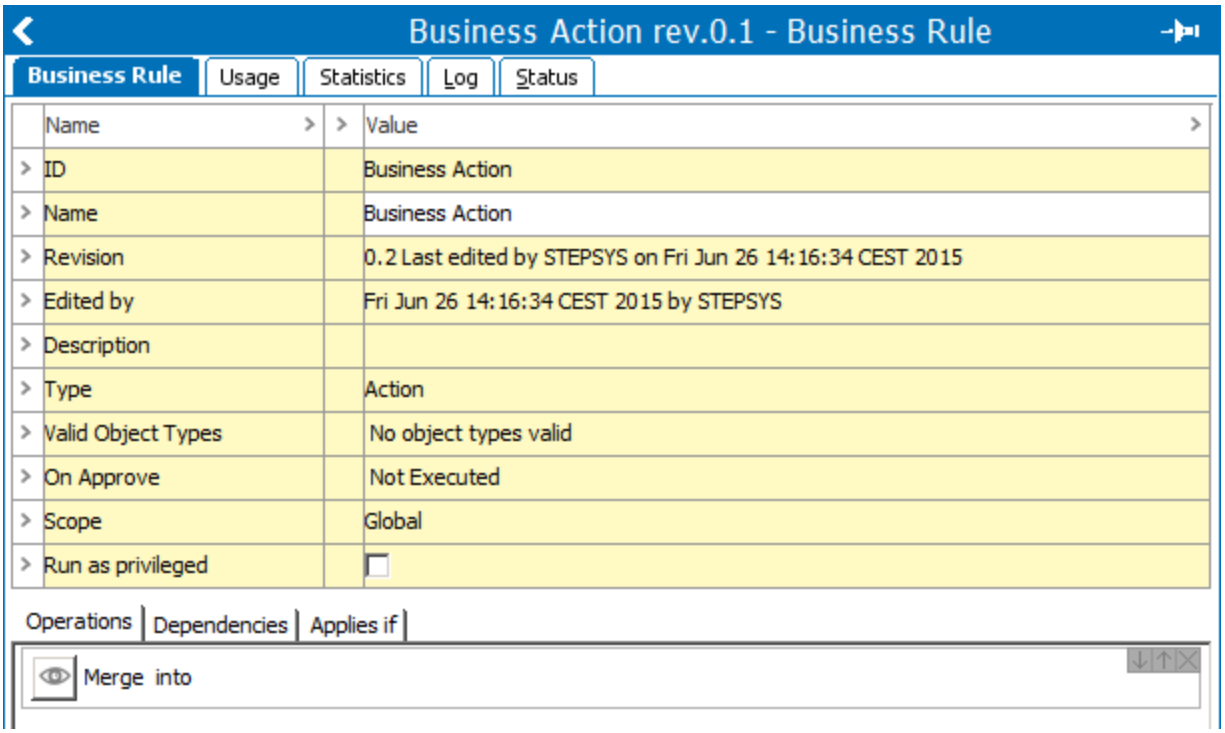
Note: Business rules viewed in the approved workspace are read-only. Business rules can only be edited from the main workspace.

For information on creating a business rule or library, see the **Creating a Business Rule or Library** topic.

For information on deleting a business rule or library, see the **Deleting a Business Rule or Library** topic.

For an overview of the decision points involved in running a business rule, see the chart in the **Running a Business Rule** topic.

Business Rule Tab



The Business Rule tab displays the following information:

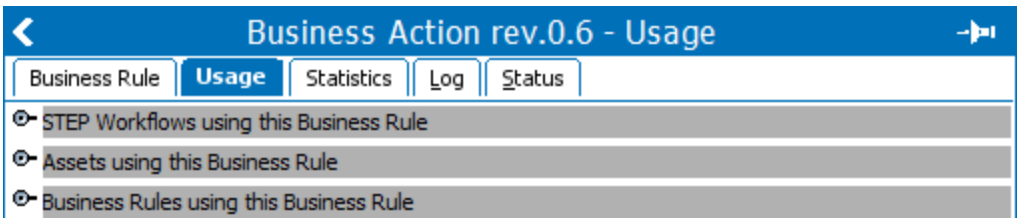
- **ID**: Displays the business rule ID.
- **Name**: Displays the business rule name.
- **Edited by**: Displays information about which STEP user created and last edited including the date and time.
- **Description**: Added by the user to display information about what the business rule does.
- **Type**: Displays whether the business rule is a condition, an action, a library, or a function.
- **Valid object types**: Displays the object types valid for the business rule. When a business rule runs, it first checks if the object type is valid for the business rule. If the object type is not valid, the result of the business rule is 'not applicable.' If it is valid, the 'Applies if' operations are checked, and if not valid, the business rule is 'not applicable.' If 'Applies if' operations are valid, the business rule operations are run. For more details and examples, see the **Testing a Business Rule** topic.
- **On Approve**: Determines if the business rule is tested or executed before or after approval. This setting is different based on the Type.
- **Scope**: Displays if the business rule is global or local. Global business rules are created via System Setup. Local business rules are created via a workflow and are local by default. For more information, see the **Global and Local Business Rules** topic.
- **Run as privileged**: Business rules are generally tested and executed with the privileges of the user who triggers the test or execution. This could be a user who approves an object or imports data into STEP. As a business rule designer however, often an action should happen or a test should be performed independent of the user who triggers the condition or action. This is accomplished via the 'Run as Privileged' parameter on the Business Rule editor. When this checkbox is selected, the Action or Condition will be executed or tested with global read / write privileges.

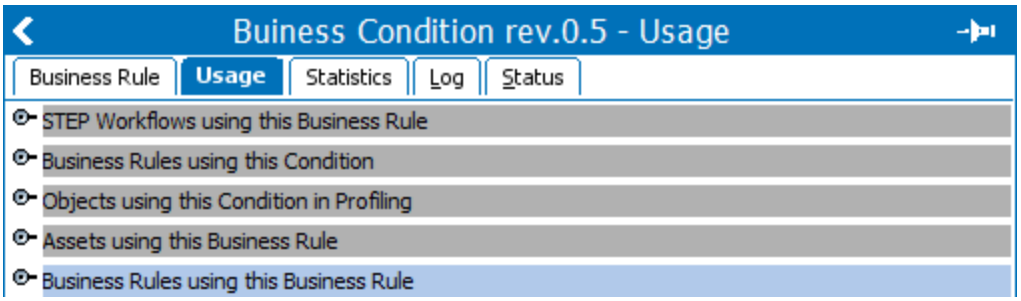
  The single Setup Action named 'Maintain business-rule' exists for maintaining business rules. This privilege can be given for specific setup groups. For more information, see the **Business Rules** section of the **Setup Actions** topic within the **System Setup / Super User Guide** documentation.

- **Operations Tab**: Any number of rules are displayed as read-only. Actions are executed from the top-down and rollback if one fails. Conditions are tested from the top-down, and return 'true' if all are true. Condition testing stops if one is false. To review, click the **View Operation** icon.
- **Dependencies Tab**: Dependencies to business libraries used for JavaScript conditions and action are displayed as read-only. To review, click the **View Operation** icon.
- **Applies Tab**: Preconditions for testing and/or executing the rule are displayed as read-only. If the conditions are not met, 'non-applicable' is displayed during testing / executing. To review, click the **View Operation** icon.
- **Edit Business Rule Link**: To edit the business rule, click the link. For more information, see **Editing a Business Rule**.

Usage Tab

The Usage tab displays information about where the business rule is used. The available flippers are determined by the type of business rule displayed, as shown in the following images.

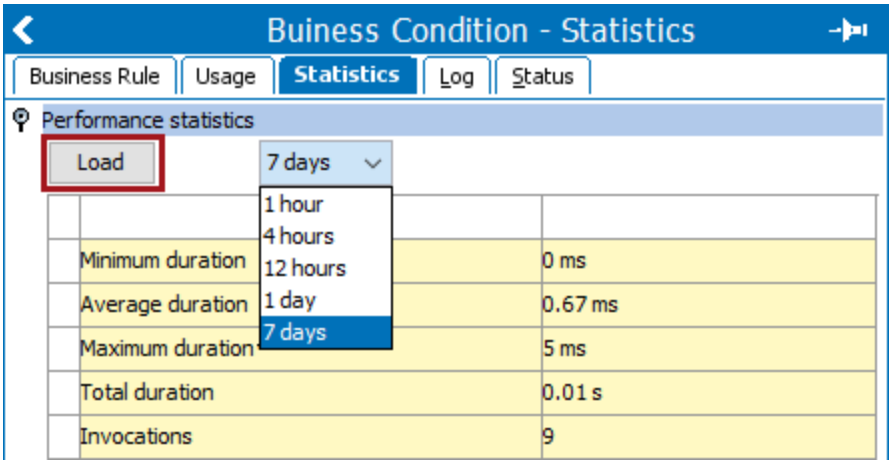




Statistics Tab

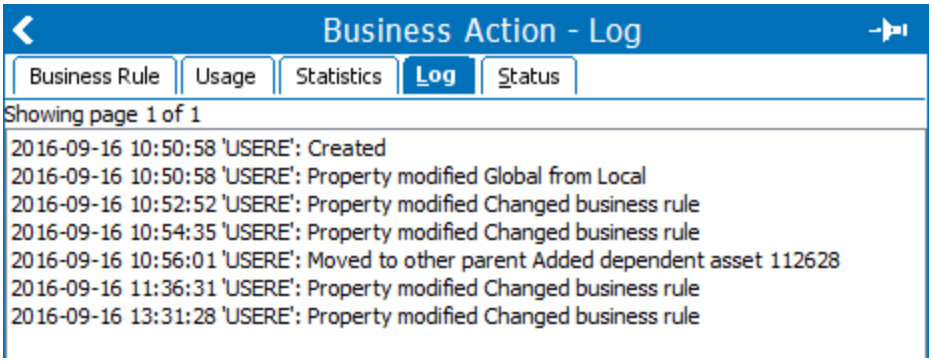
This tab shows statistics about how many times a business rule has been invoked and how it performed in the given period.

To view the statistics, select a time interval from the dropdown, and then click the **Load** button.



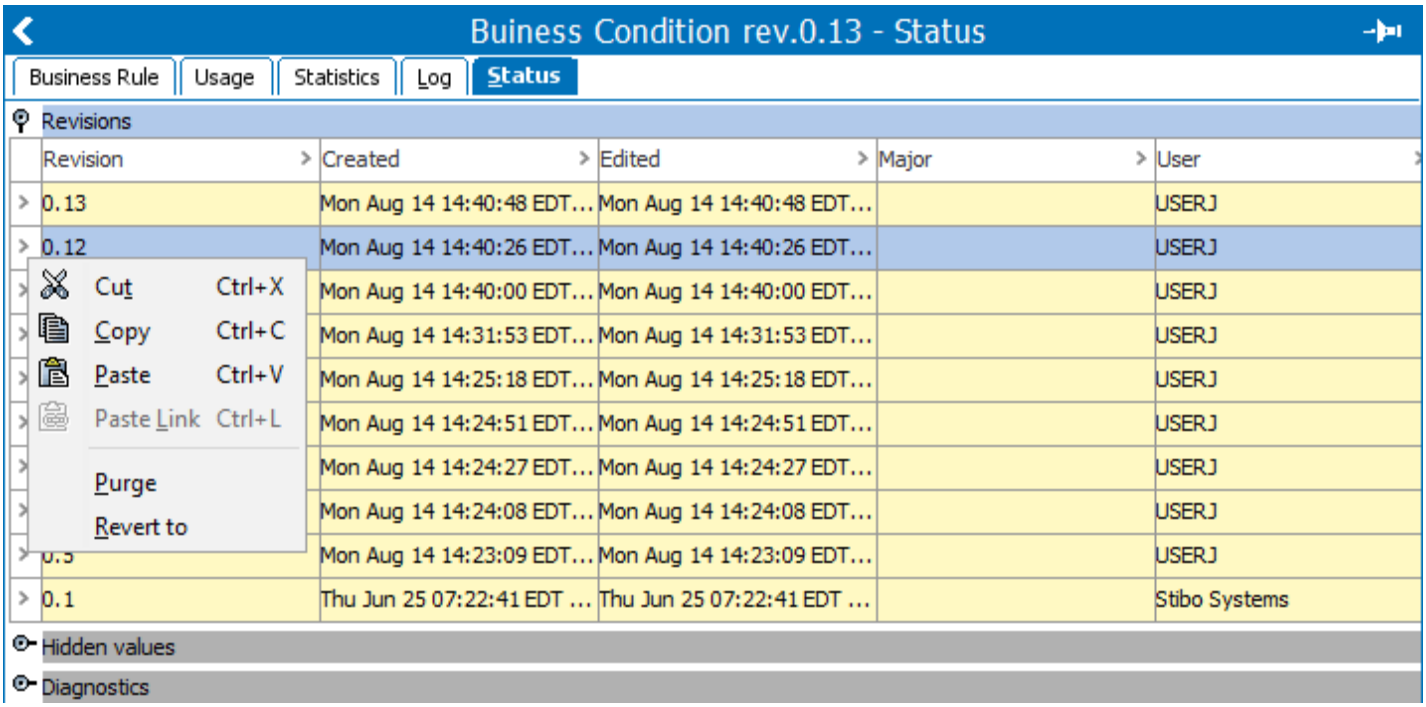
Log Tab

The Log tab shows the users and all changes that have been made.



Status Tab

The Status tab provides an overview of the revisions of the business rule, including the user who changed the object and when the change occurred. Previous versions of business rules can be viewed as read-only by clicking the eye icon for the rule.



- **Make Revision**: Select the business rule, on the Maintain menu, click the **Make Revision** option.
- **Revert to**: Select the desired revision row, right-click the > column, and click **Revert to** the option from the menu to restore a previous version.

Note: When reverting to a previous version of a workflow, any local business rules owned by the workflow in question will also be reverted to the revisions that were current at the time of the workflow revision. For more information, see the **Managing Workflows** section of the **Workflows** documentation.

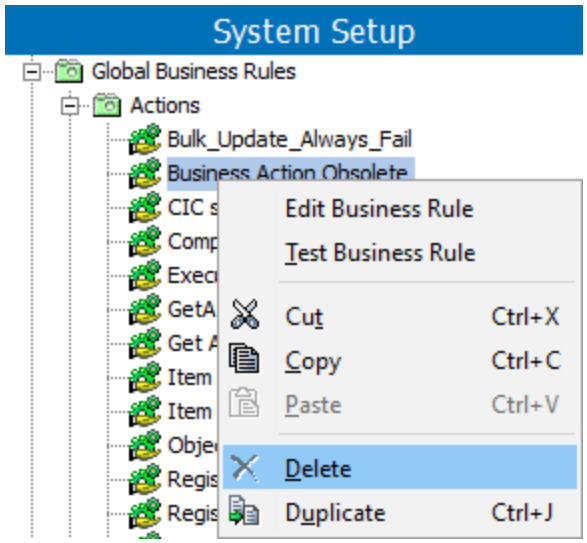
- To delete a revision, select the desired row, right-click the > column, and click the **Purge** option from the menu.

For detail about creating revisions, see the **Managing Revisions in STEP** topic within **System Setup / Super User Guide** documentation.

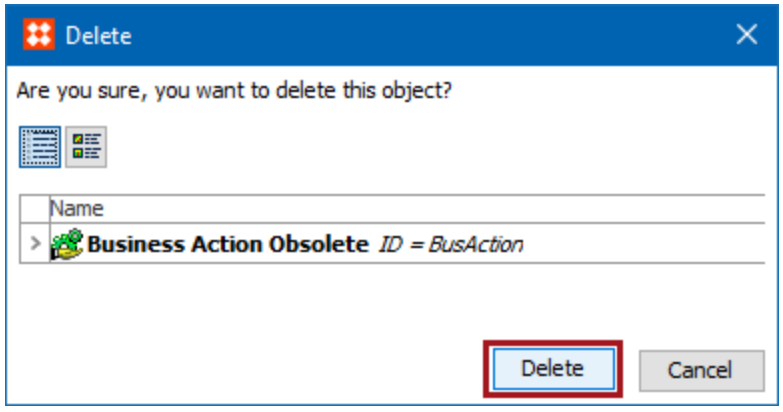
Deleting a Business Rule or Library

When no longer needed, business rules and libraries can be deleted.

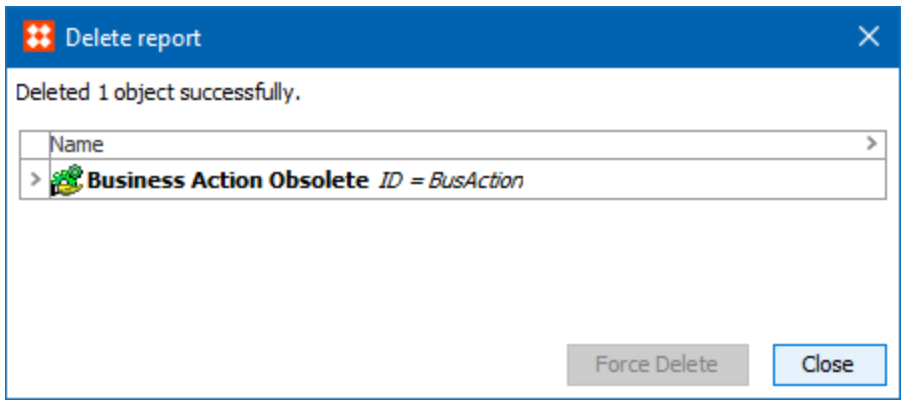
1. On System Setup, select the business rule or library to be deleted.
2. Right-click to display the menu and click the **Delete** option.



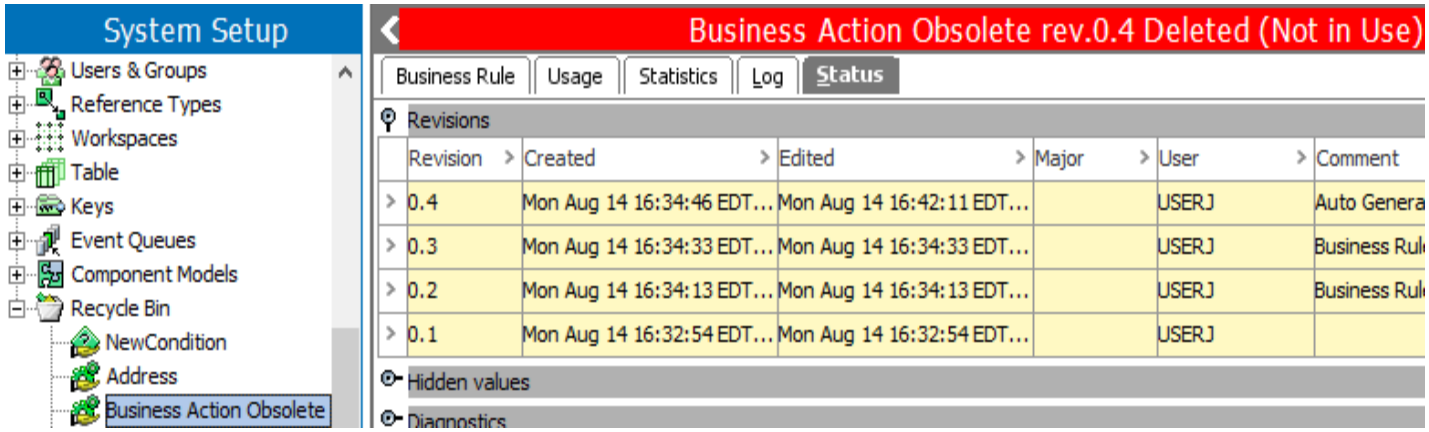
3. On the Delete dialog, click the **Delete** button to verify that the business rule should be moved to the System Setup Recycle Bin.



4. On the Delete Report dialog, click the Close button. Otherwise, follow the instructions for Force Delete.



5. The business rule or library is displayed in the System Setup Recycle Bin and can be revived or purged from that location. For more information, see the **Recycle Bin for System Setup** topic.



Exporting Business Rule Definitions as Comments

Business Rules definitions, including conditions, actions, functions, and libraries, can be exported as comments using Advanced STEPXML. These exports are intended to be used for submission to external source control systems for comparison purposes. Users can import them into source code repository systems where they can be compared from version to version. Editing and/or import of these files is not supported (e.g. users may not export, edit the comments, and re-import in STEP).

To export business rule definitions for external comparison, Advanced STEPXML must be used and the DefinitionsAsComments tag must be set to 'true'.

On the Select Format step of the outbound tool, choose the Advanced STEPXML format, then copy and paste the following text into the Template field:

```
<?xml version="1.0" encoding="utf-8"?>
<STEP-ProductInformation DefinitionsAsComments="true">
<BusinessLibraries ExportSize="All"/>
<BusinessRules ExportSize="All"/>
</STEP-ProductInformation>
```

An example of the output for a business rule definition as comments is shown below:

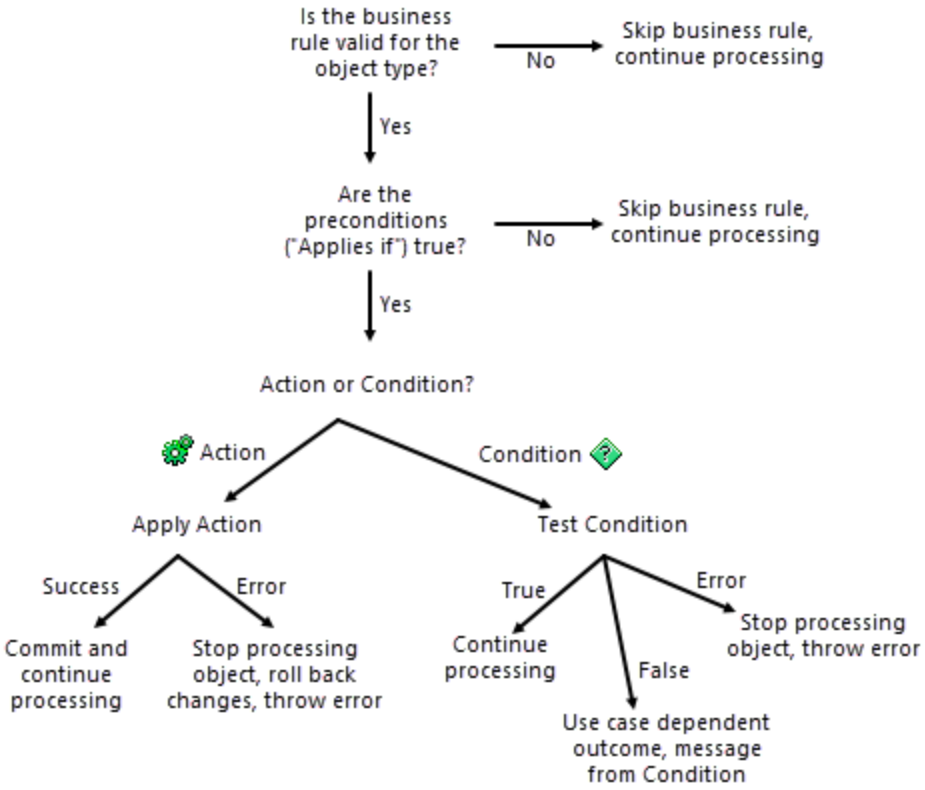
```
<BusinessRule...>
<!--Definition:[This will be the business rule definition. Removed for
brevity.-->
[Remaining configuration is included]
</BusinessRule>
```

Note: The content of the comment field is not part of the STEPXML XSD and therefore Stibo Systems reserves the right to change the format of the output content at any time.

For more information, see the **STEP-ProductInformation Tag in STEPXML** section of the **Data Exchange** documentation.

Running a Business Rule

The following chart shows the decisions that determine when and if a business rule runs.



For more details and examples of test results, see the **Testing a Business Rule** topic.

Business Actions

A Business Action is a piece of business logic that can be executed to manipulate data or perform an action. Business actions define the operations that can happen during a variety of system processes and events. For example, at approval of an object, within a workflow (on entry to a state, exit from a state, on the transition between tasks, or when a deadline is met), as part of an import process, or from a bulk update process.

While business actions can be created using JavaScript, simple operations are available as described in the table below.

Important: Business actions can be used to modify data and/or take action, while business conditions should be used to validate data only (not to modify data). For more information, see **Business Conditions**. Alternatively, business functions will produce an output based on non-changing input parameters. For more information on business functions, see the **Business Function** topic.

Error Messages

It is possible for business actions to return error messages, and when this occurs, processing is stopped and any updates are rolled back. There are use cases where this will streamline the writing and execution of business rules by mixing validations and data updates, especially if the conditions include data, as it appears after the updates have been made. However, it is strongly recommended that, whenever possible, conditions (which are read-only) and actions (which contain updates) be separate, and that any validation-related error messaging is performed only in the business condition. This will ensure the best performance. Additionally, there are situations where conditions can be evaluated dynamically to provide on-the-fly feedback to users prior to performing an expected action (e.g., enabling / disabling action buttons based on condition results). Keep in mind that when an action encounters an error message, any included updates will not be run until the action is actually performed.

Business Action Operations

To add a business action to a business rule, see the **Specifying a Business Action** topic.

Menu	Action	Description
Attribute values	**Merge Attribute Values**	Merges the value of one attribute into another attribute. For more information, see **Business Action: Merge Attribute Values**.
Data quality	**Generate Match Codes**	Generates match code values as a part of a business action. For more information, see **Business Action: Generate Match Codes**.
	Standardize address	Overwrites the existing standardized address. Additionally, can be used to generate Loqate address hash values, renew address validations

Menu	Action	Description
		older than a specified number of days, and enable CASS validation for US addresses. For more information, see **Business Action: Standardize Address**.
GDSN	**Create CIC object and start workflow**	For more information about GDSN Provider, see **Business Rules with GDSN**.
	Execute command	
	Execute pending command	
	Set CIC status for publication	
	Set pending command	
	Update package hierarchy status	
GDSN Receiver	**CICR exception handler**	For more information, see **GDSN Receiver**.
	Invoke CIC message	
	RFCIN message completed	
	RFCIN message failed	

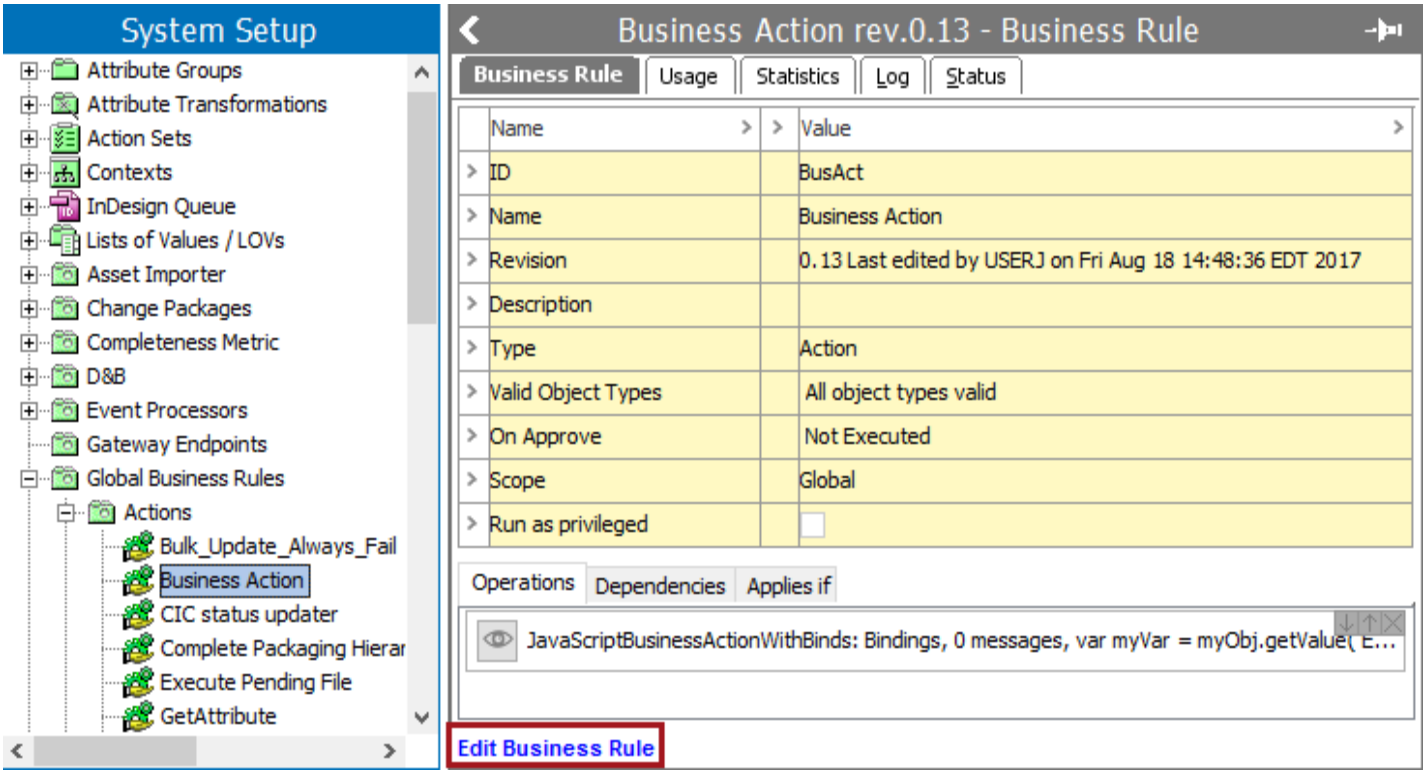
Menu	Action	Description
	RFCIN message send	
	Set status or remove subscription	
	Set subscription status	
References and Links	**Add Attribute Link**	Adds a link to an attribute without additional configuration. For more information, see **Business Action: Add Attribute Link**.
	Add Reference	Adds a reference of a specified type to a target. For more information, see **Business Action: Add Reference**.
	Add Referenced By	Adds a reference of a specified type from the specified source to the object that is executed as target. For more information, see **Business Action: Add Referenced By**.
	Automatic Classification	Classifies the selected objects automatically according to a set of rules. For more information, see **Business Action: Automatic Classification**.
	Remove Attribute Link	Makes it possible to remove a link to an attribute without additional configuration. For more information, see **Business Action: Remove Attribute Link**.
	Remove Reference	Removes references to a specific target or all references of a specific type. For more information, see **Business Action: Remove Reference**.
	Set Product to Classification Link Type	Assigns a new object type and classification link type to the selected product. For more information, see **Business Action: Set Product to Classification Link Type**.
Workflow	**Claim**	Claims the task of the current object in the specified workflow and state. For more information, see **Business Action: Claim**.

Menu	Action	Description
	Initiate Items in STEP Workflow	Initiates the object the action runs on in a specified workflow. For more information, see **Business Action: Initiate Items In STEP Workflow**.
	Remove object from STEP Workflow	Removes the object the action runs on from the specified workflow. For more information, see **Business Action: Remove Object from STEP Workflow**.
	Trigger STEP Workflow Event	Triggers a specified workflow event. For more information, see **Business Action: Trigger STEP Workflow Event**.
Execute JavaScript		Enables you to use the methods available in the STEP Public Java API. For more information, see **Execute JavaScript**.
Overlap Analysis		Compares records in a pre-defined format with all record of the same object type by specifying a matching algorithm. For more information, see **Business Action: Overlap Analysis**.
Reference Other Business Action		Allows the node to reference other business actions without additional configuration. For more information, see **Business Action: Reference Other Business Action**.
Send Email		Sends email to specified recipients. For more information, see **Business Action: Send Email**.
Send Republish Event		Sends a republish event to an event queue processor. For more information, see **Business Action: Send Republish Event**.
Set Attribute Value		Sets a specified attribute value for the object where the action is executed. For more information, see **Business Action: Set Attribute Value**.
Set Name		Sets the name of the object that the action runs on. For more information, see **Business Action: Set Name**.
Set Object Type		Changes the object type the action runs on. For more information, see **Business Action: Set Object Type**.
Set Workflow Variable		Enables assignment of workflow variables. For more information, see **Business Action: Set Workflow Variable**.

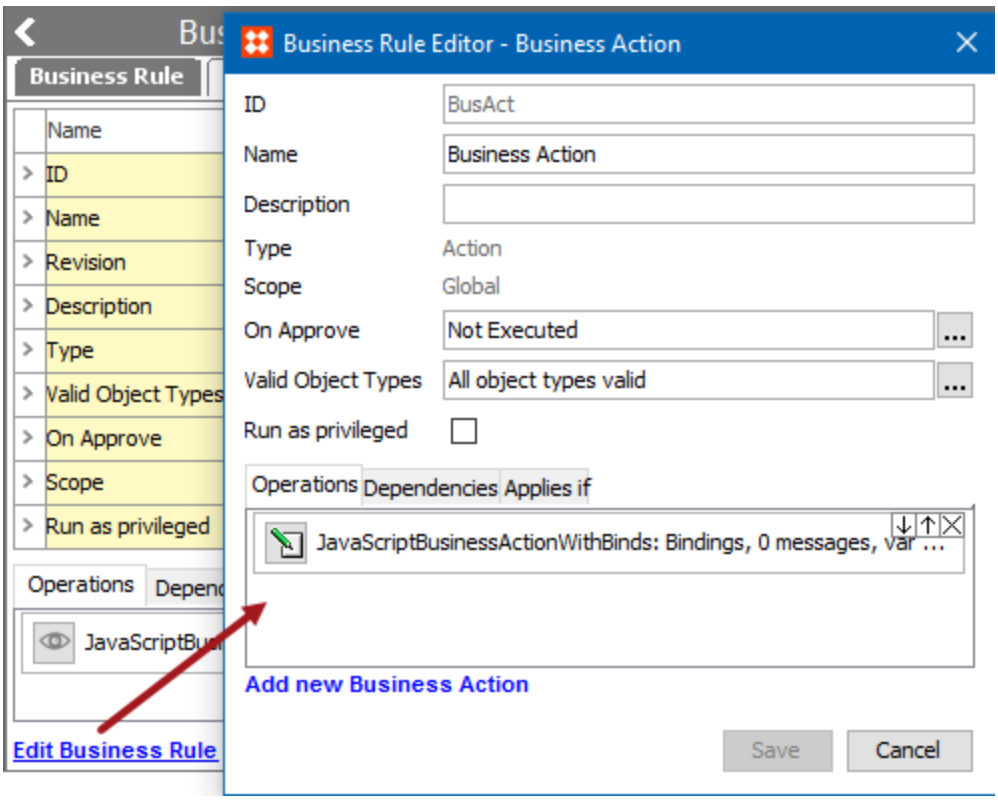
Specifying a Business Action

Once a business rule exists, use the following steps to add a business action. For information on creating a new business rule, see the **Creating a Business Rule or Library** topic.

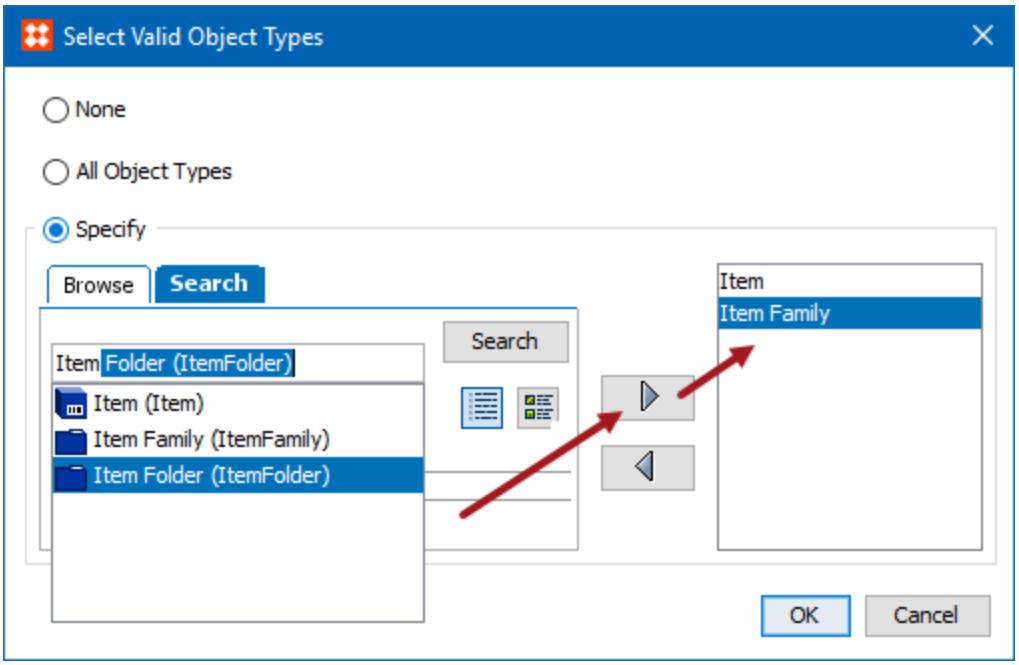
1. In System Setup, expand the **Business Rules** setup group and select an existing business action to display the business rule editor.



2. On the Business Rule tab, click the **Edit Business Rule** link to open the business rule editor for changes.



3. In the **Valid Object Type** field, click the ellipsis button **(…)** to display the Select Valid Object Types dialog.

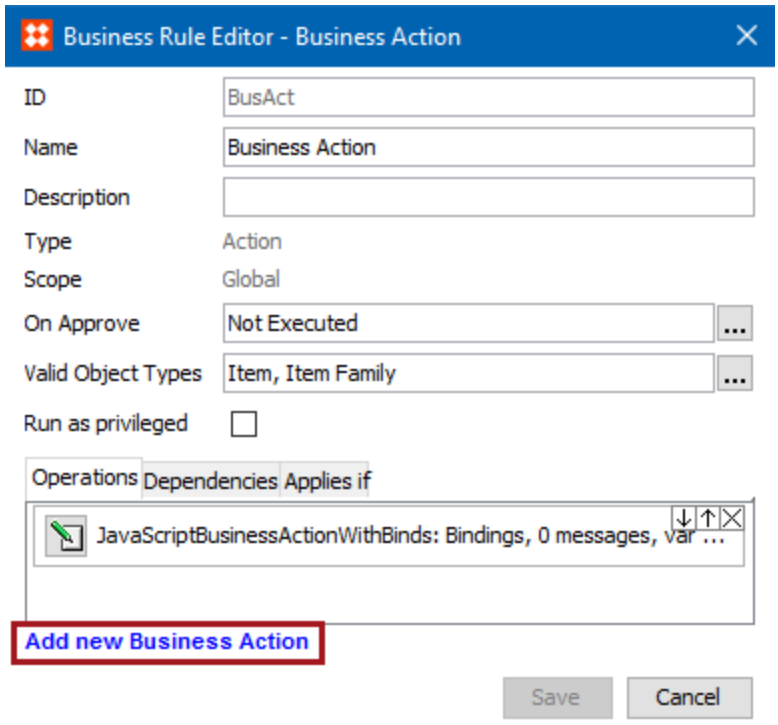


4. Choose a radio button:

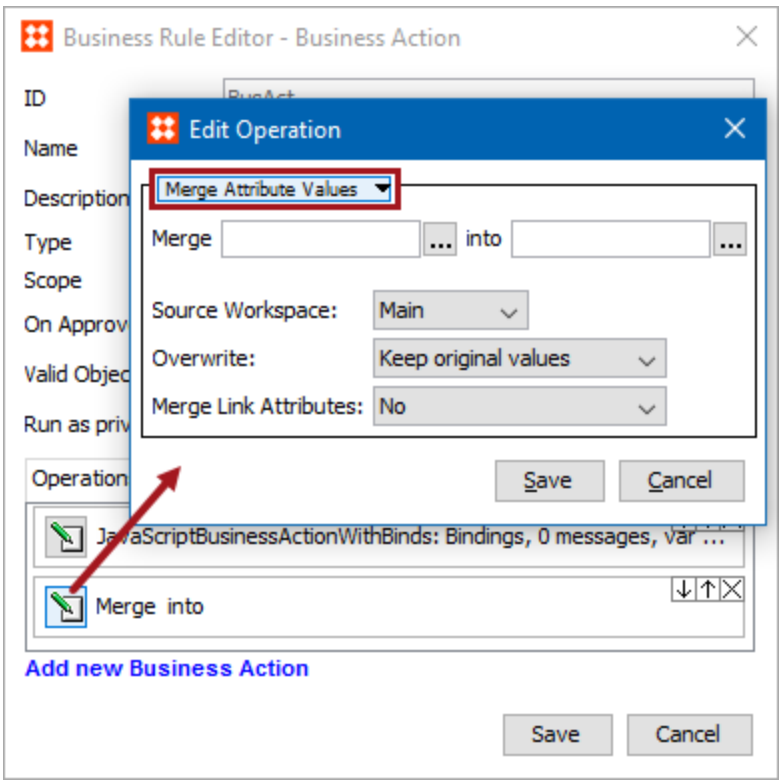
- **None** prevents the business action from running. This can be used to temporarily disable an action.
- **All Object Types** means the business action will run regardless of the object type. It is not recommended to use this option.
- **Specify** allows you to limit the object types that will cause the business action to run. Selecting Specify displays the Browse and Search tabs. Identify the desired object types and use the right arrow button to move the item to the right panel. Use the left arrow button to remove an object type from selection.

After selecting object types, click the **OK** button to save the changes.

5. In the lower left corner, click the **Add New Business Action** link to add an additional operation.



6. Click the **Edit Operation** icon for the newly added business action.



7. In the Edit Operation dialog, use the dropdown list to select the preferred action. The actions are described in the **Business Actions** topic.
8. Click **Save** to save the changes.
9. Add additional business actions as needed.

Important: When more than one action is specified, all actions are carried out when the overall business action is executed.

Business Action: Add Attribute Link

This action can link a specification attribute to a product. This allows you to link an attribute with a specific hierarchy, and then inherit it to all children of that hierarchy, based on the validity of the product attribute. The attribute is displayed on the product's References tab under the 'Linked Attributes from Product Hierarchy' flipper.

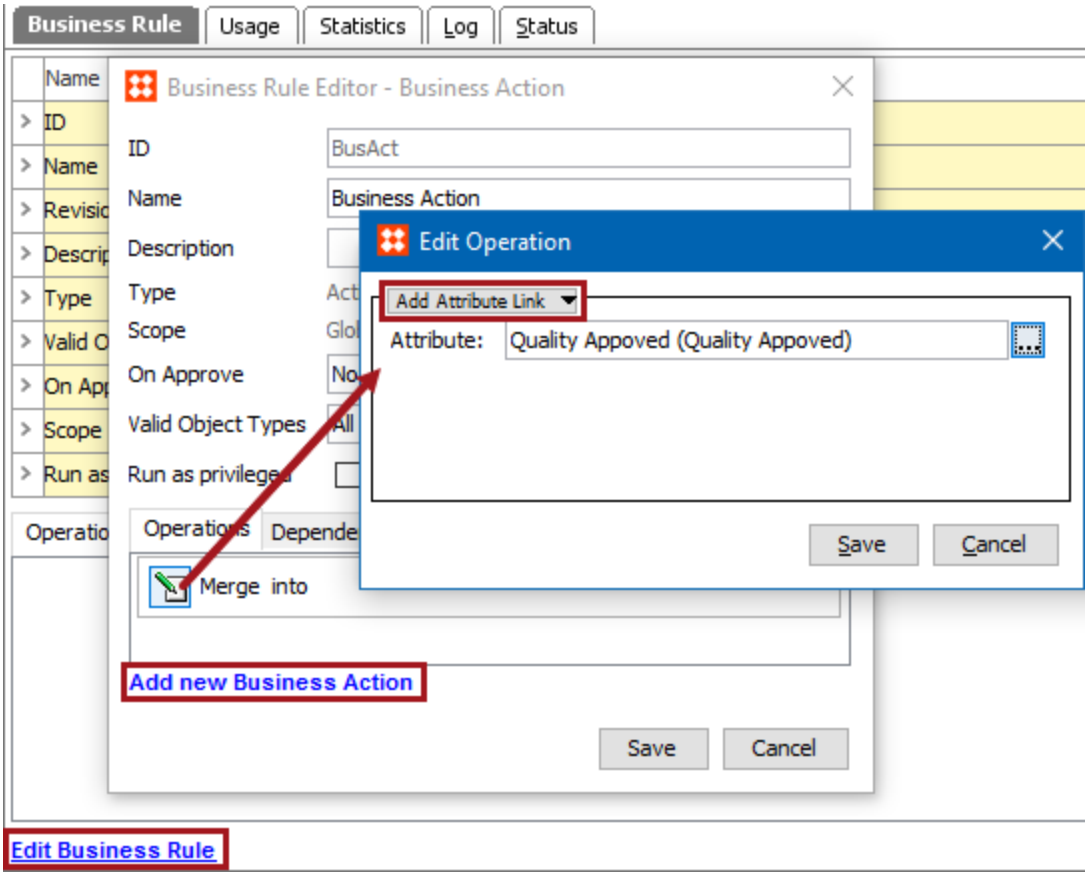
For example, the product 'Dining Tables' is under a 'Furniture' section of the primary product hierarchy (PPH). A new line of LED dining tables is introduced, for which there is a new 'Battery' attribute is required. A business action can be used to create the product attribute link between the tables in the LED hierarchy and the battery attribute.

Prerequisites

Before using this action:

1. Ensure the product attribute link type is set up. For more information, see the **Product Attribute Link Type** topic in the **System Setup / Super User Guide** documentation.
2. Ensure the selected attribute is valid on the product object type being linked.
3. Create a business rule as defined in the **Creating a Business Rule or Library** topic.

Configuration



1. On the Edit Operation dialog, select the References and Links > **Add Attribute Link** option from the dropdown.
2. In the **Attribute** parameter, click the ellipsis button **(…)** to display the Select Attribute dialog, select a reference type, and click the **Select** button.
3. Click the **Save** button to add the operation to the business rule editor.

Business Action: Add Reference

This action can create a reference / link to an object with a specific target and a type. This allows you to add an entry on the References tab of the editor for the selected object.

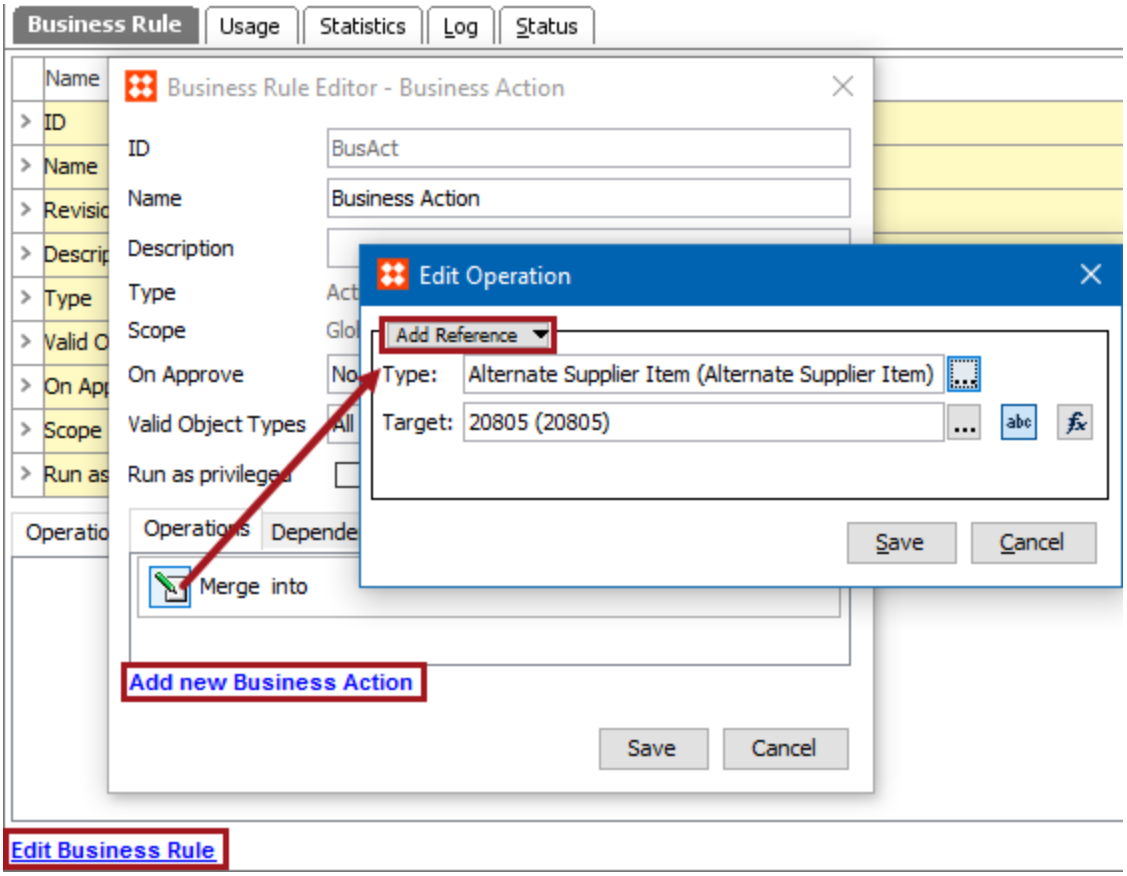
For example, a business action can be used to reference the object to another object automatically, based on specified type and target. Additionally, it can automatically link the products to particular classification on import by adding an Add Reference business action to Import Manager.

Prerequisites

Before using this action:

1. Ensure a valid reference type with the necessary target object exists, and that the target object is valid for the source object. For more information, see the **Reference and Link Types** topic in the **System Setup / Super User Guide** documentation.
2. Create a business rule as defined in the **Creating a Business Rule or Library** topic.

Configuration



1. On the Edit Operation dialog, select the References and Links > **Add Reference** option from the dropdown.

2. In the **Type** parameter, click the ellipsis button **(…)** to display the Select Reference Type dialog, select a reference type, and click the **Select** button.
3. In the **Target** parameter, use one of these methods to select the relevant target. The target must be in accordance with the selected type, otherwise an error is displayed.
   - Click the 'abc' button (), click the ellipsis button **(…)** to display the Select Reference Target dialog, select a valid target, and click the **Select** button.
   - Click the 'fx' button (), click the ellipsis button **(…)** to display the Function Editor dialog, write a calculation to select a valid target, and click the **Select** button. For more information, see the **Using Function Editor** topic in the **System Setup / Super User Guide** documentation.
4. Click the **Save** button to add the operation to the business rule editor.

Configuration Notes

- Attempting to add a target which already exists on the object causes the action to fail with the message 'Reference already exists :< reference type>'
- When the calculated function cannot return the specific Target Type ID, the 'Could not get the reference target.' is returned.

Business Action: Add Referenced By

This action can create a reference to an object with a specific source and a type. This allows you to add an entry on the Referenced By tab of the editor for the selected object.

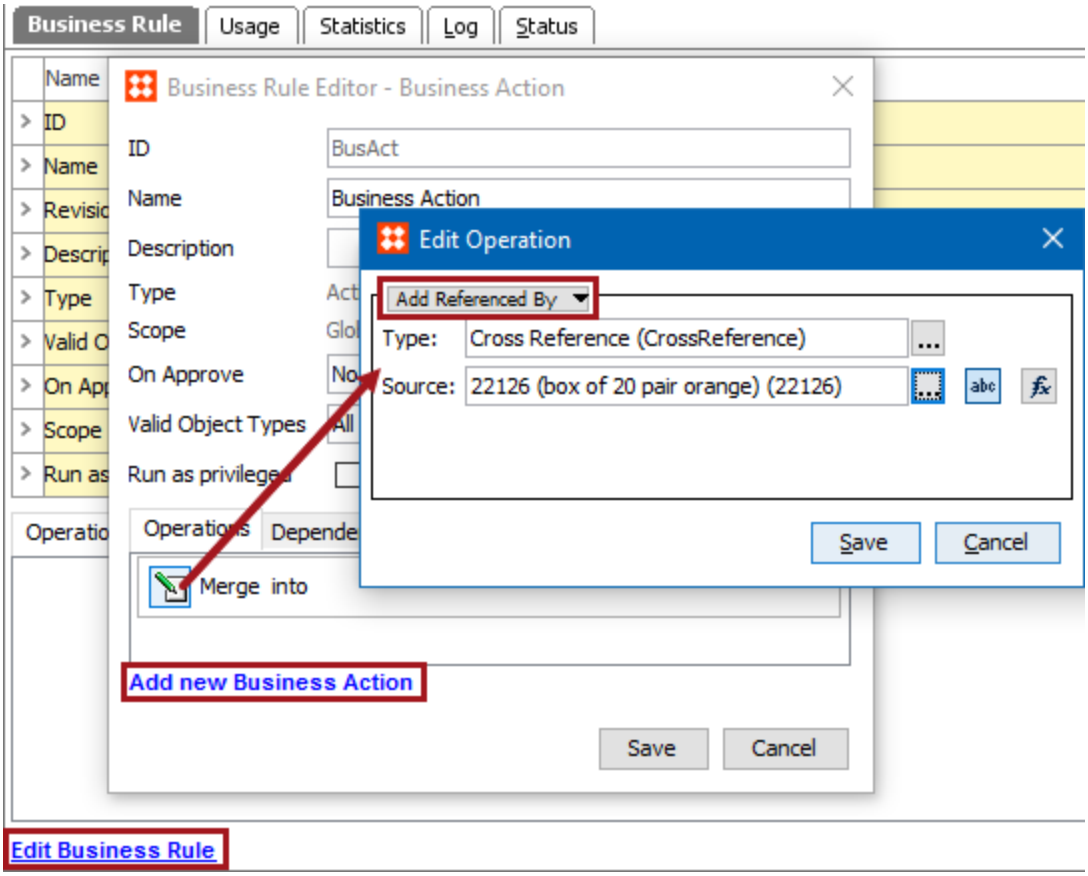
For example, a business action can be used to add an object as the source to an image using a specified reference type.

Prerequisites

Before using this action:

1. Ensure a valid reference type with the necessary source object exists, and that the source object is valid for the target object. For more information, see the **Reference and Link Types** topic in the **System Setup / Super User Guide** documentation.
2. Create a business rule as defined in the **Creating a Business Rule or Library** topic.

Configuration



1. On the Edit Operation dialog, select the References and Links > **Add Referenced By** option from the dropdown.

2. In the **Type** parameter, click the ellipsis button **(…)** to display the Select Reference Type dialog, select a reference type, and click the **Select** button.
3. In the **Source** parameter, use one of these methods to select the relevant source. The source must be in accordance with the selected type, otherwise an error is displayed.
   - Click the 'abc' button (), click the ellipsis button **(…)** to display the Select Reference Source dialog, select a valid source, and click the **Select** button.
   - Click the 'fx' button (), click the ellipsis button **(…)** to display the Function Editor dialog, write a calculation to select a valid source, and click the **Select** button. For more information, see the **Using Function Editor** topic in the **System Setup / Super User Guide** documentation.
4. Click the **Save** button to add the operation to the business rule editor.

Business Action: Automatic Classification

This action can, upon approval, invoke a rule set on the object to create or maintain the reference, or move the object to a different hierarchy. the selected objects automatically according to a set of rules.

For example, upon import, executing this business action can automatically classify object in a website hierarchy or a classification hierarchy, based on a standard taxonomy like ETIM or UNSPSC. Additionally, when executing this business action, objects can be classified or reparented based on an attribute value.

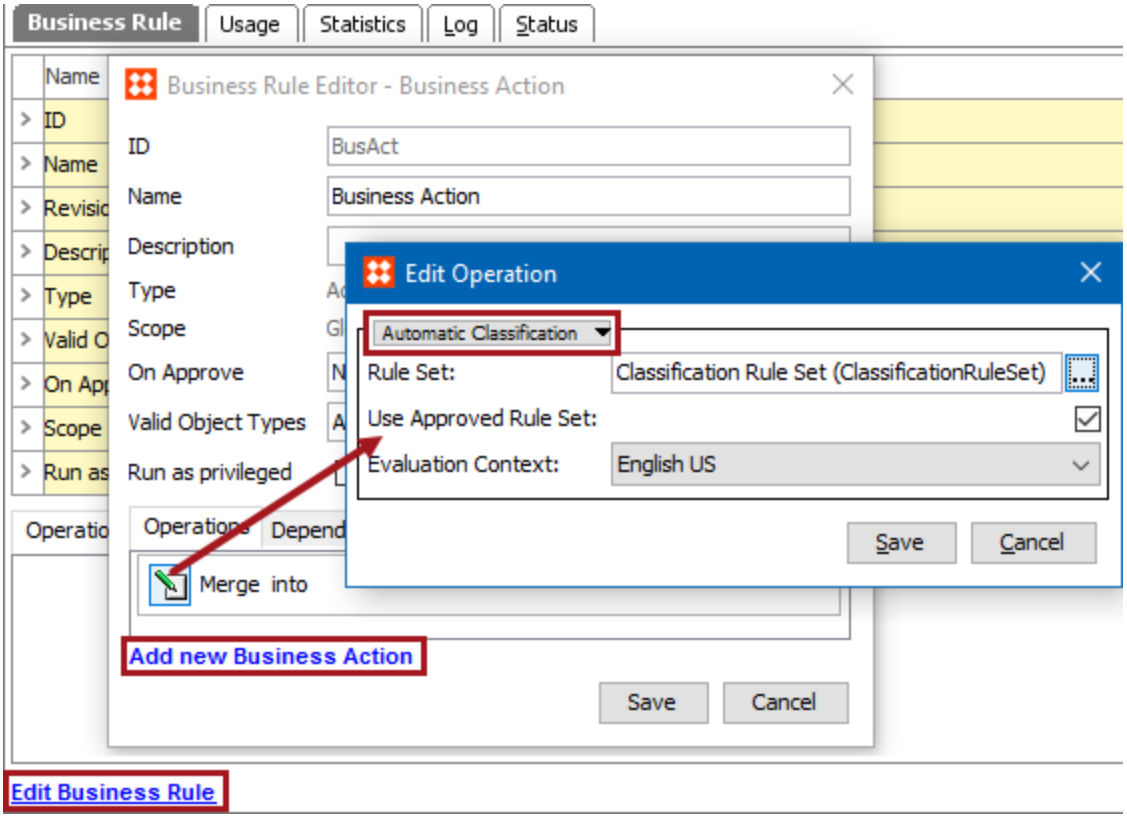
For more information about automatic classification, see the **Using Automatic Classification with Business Actions** topic in the **Automatic Classification** documentation.

Prerequisites

Before using this action:

1. Ensure the Automatic Classification license is installed.
2. Ensure a rule set configuration for automatic classification exists, as defined in the **Initial Setup for Automatic Classification** topic.
3. Create a business rule as defined in the **Creating a Business Rule or Library** topic.

Configuration



1. On the Edit Operation dialog, select References and Link > **Automatic Classification** option from the dropdown.
2. For **Rule Set**, click the ellipsis button **(…)**, and select the rule set to be executed.
3. For **Use Approved Rule Set**, check the checkbox to use the approved version of the rule set.
4. For **Evaluation Context**, select the context where the action should occur.
5. Click the **Save** button to add the operation to the business rule editor.

Business Action: Claim

This action allows a user to claim the workflow task of the current object from the group. Additionally, if a workflow task is already assigned to a specific user, this action claims it for the current user.

For example, a workflow with ID 'QA' includes quality assurance tasks that are assigned to a 'Super Users' group. The user 'Admin' wants to be assigned tasks when they enter the 'Review' state. The Claim business action is configured on the 'On Entry' tab of the workflow State Editor for the 'Review' state. This causes tasks to automatically be assigned to 'Admin' when the task enters the 'Review' state.

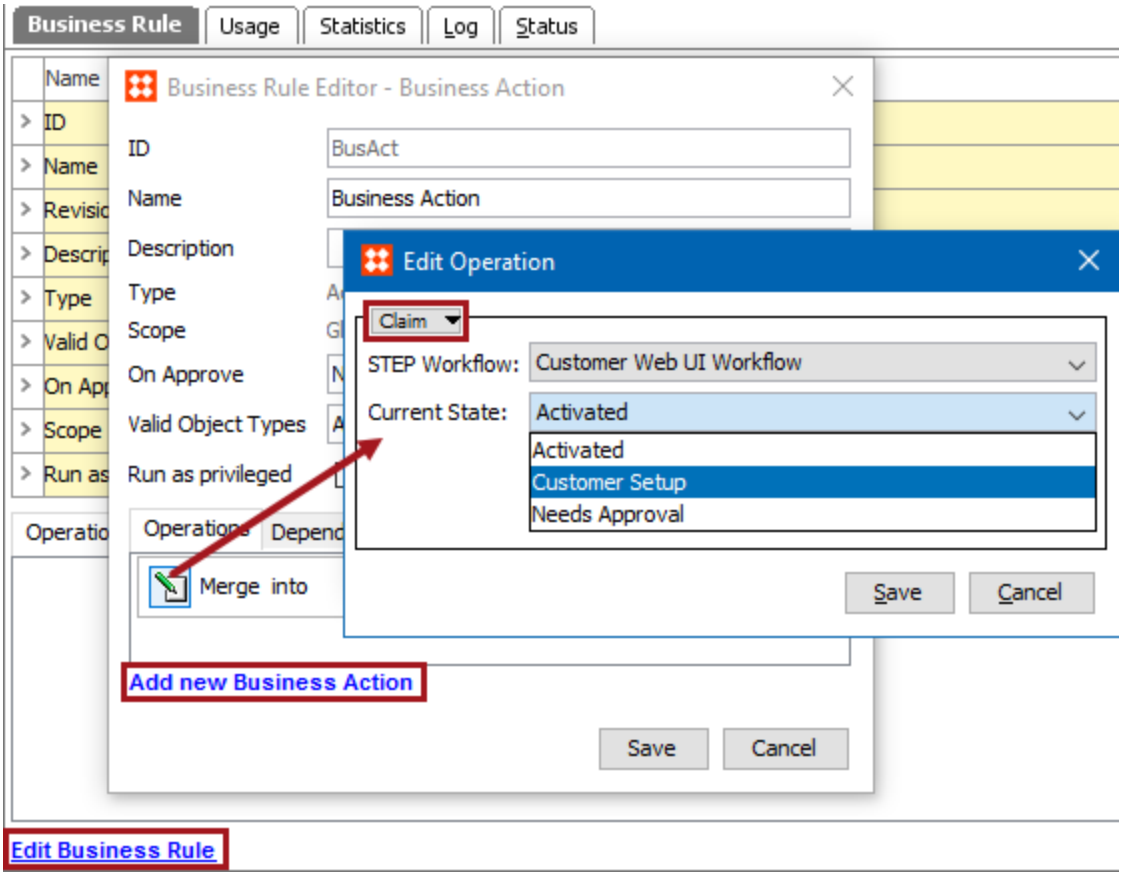
For more information, see the **Business Rules and Workflows** topic in the **Workflows** documentation.

Prerequisites

Before using this action:

1. Ensure the workflow exists, and has multiple states.
2. Create a business rule as defined in the **Creating a Business Rule or Library** topic.

Configuration



1. On the Edit Operation dialog, select the Workflow > **Claim** option from the dropdown.
2. For **STEP Workflow**, select the relevant workflow from the dropdown.
3. For **Current State**, from the dropdown, select from the available states for the workflow.
4. Click the **Save** button to add the operation to the business rule editor.

Business Action: Execute JavaScript

In addition to the standard business rule operations, more complex functions can be carried out using JavaScript and the Scripting API. This action enables you to use the same STEP Public Java API methods that are available in the Evaluate JavaScript business condition operation.

Using JavaScript in business rules can include:

- **Execute JavaScript** operation is used for actions. For more information about business rule actions, see **Business Actions**.
- **JavaScript Binds** for action and condition operations. Binds provide access to the STEP data being worked on. For more information, see **JavaScript Binds**.

  Important: Although the same JavaScript binds are available for both actions and conditions, changing STEP data via an Evaluate JavaScript business condition is not supported.

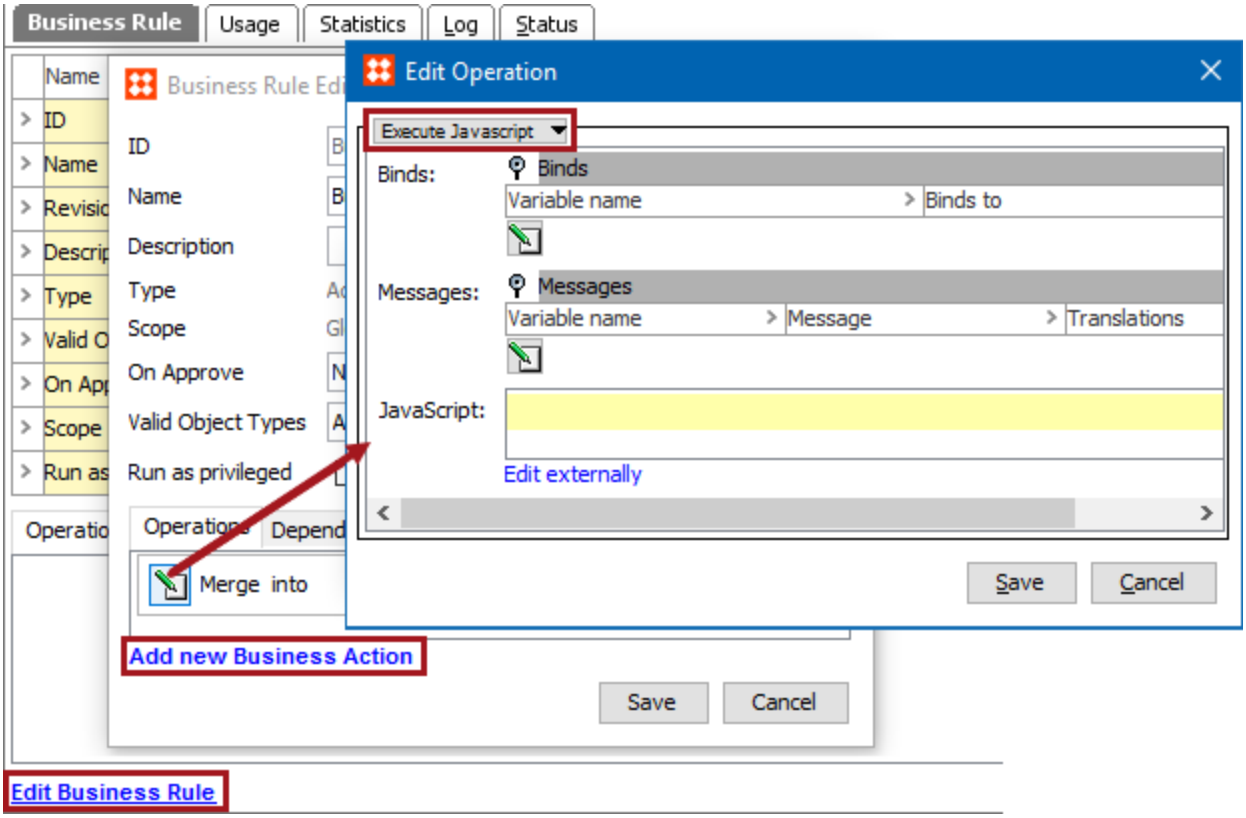
- **Business Libraries** is a set of JavaScript functions that can be reused in multiple business rules. For more information, see **Business Libraries**.
- **Scripting API** documentation is available by clicking the **STEP API Documentation** button on the STEP WebStart page.

Prerequisites

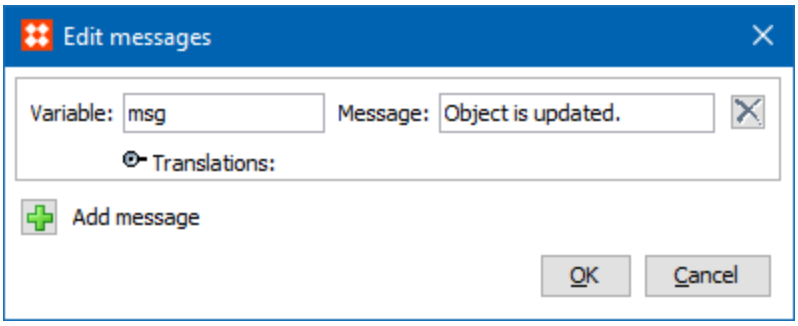
Before using this action:

- Understand the implications of using JavaScript as defined in the **Java vs. JavaScript** topic.
- Understand the considerations that should be taken as defined in the **JavaScript Considerations** topic.
- Create a business rule as defined in the **Creating a Business Rule or Library** topic.

Configuration



1. On the Edit Operation dialog, select the **Execute JavaScript** option from the dropdown.
2. For the **Binds** parameter, create binds needed for the JavaScript code as described in the **Adding a Bind** topic.
3. For the **Messages** parameter, create messages needed for the JavaScript code as follows:
   - Click the **Edit** button ( ) to display the Edit Messages dialog.



   - Click the **Add message** button ( ) to create a new message entry.
   - Click the **Remove** button ( ) to delete a message entry.

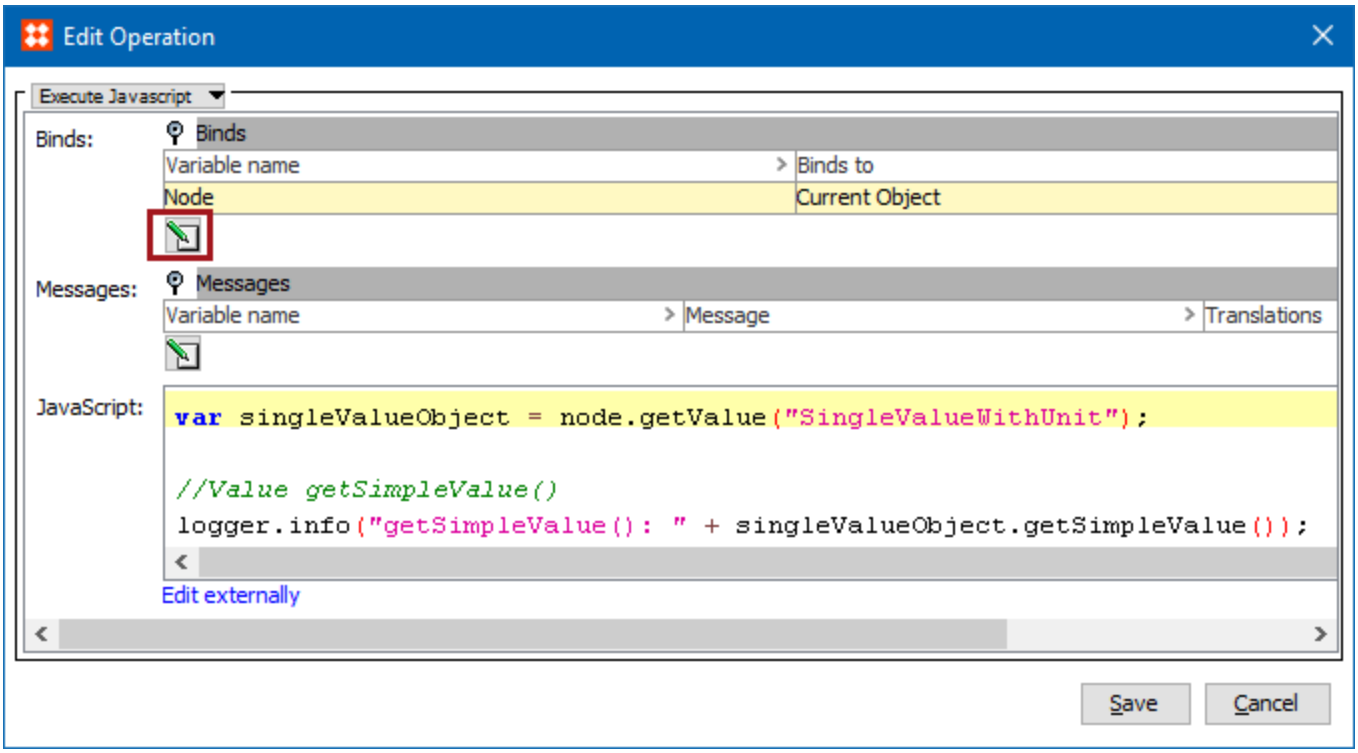
- For the **Variables** text box, type a label for the variable.
- For the **Message** text box, type the message.
- When necessary, open the Translations flipper and provide the data required. For more information on translatable messages, see the **Adding a Localized Business Rule Message** topic.
- Click **OK** to save the messages to the operation.

4. For the **JavaScript** parameter, add the JavaScript code.
   - Click the **Test JavaScript** button to test modifications made to the JavaScript.
5. Close the dialog usign the appropriate method.
   - Click the **Save** button to keep the modifications (including changes to JavaScript) and add the operation to the business rule editor.
   - Click the **Cancel** button to roll back the modifications (including changes to JavaScript).

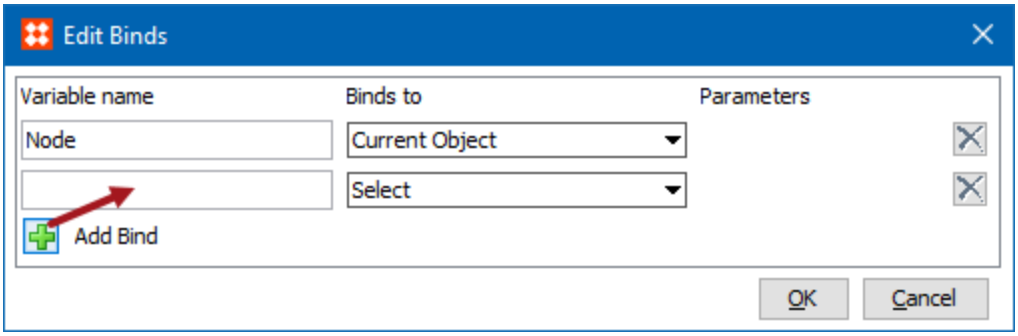
Adding a Bind

Binds allow you to reference an object and/or value within the JavaScript code.

1. Open the Edit Operation dialog for an existing JavaScript business rule.
2. On the Binds parameter, click the Edit button ( ) to display the Edit Binds dialog.

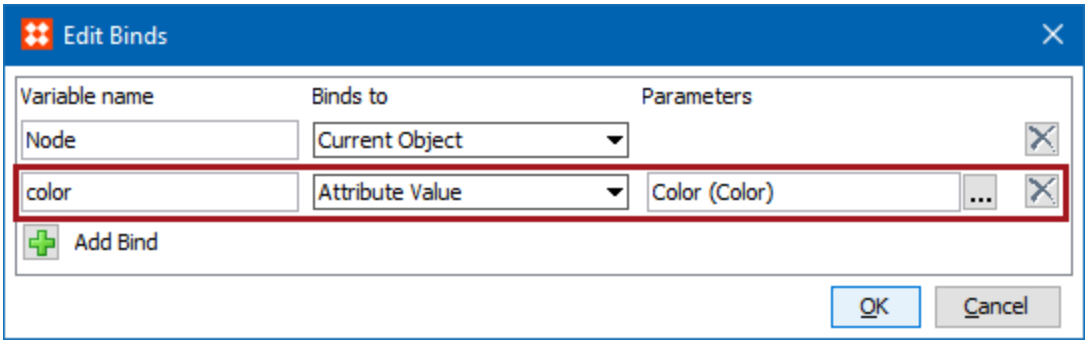


3. Click the Add Bind button ( ) and a new bind row is added to the list.

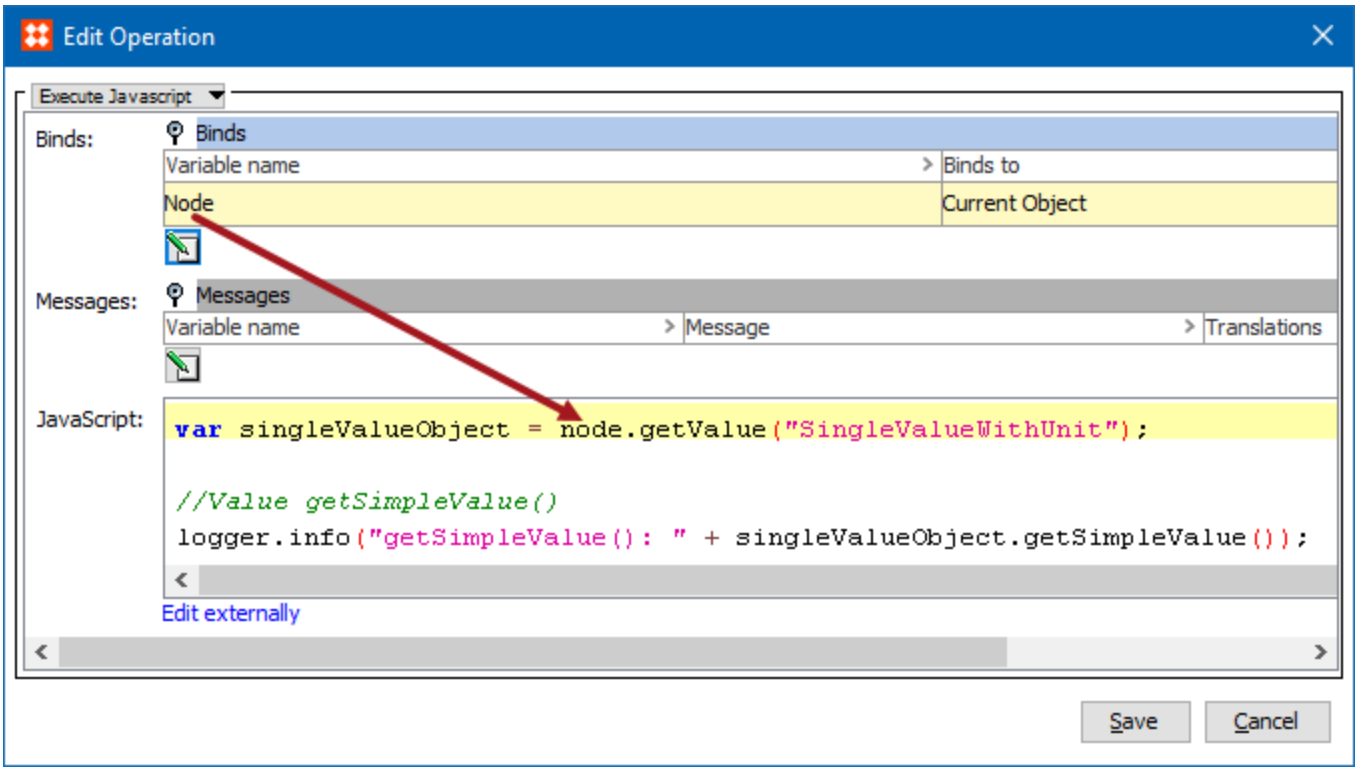


4. Add the following information:

- **Variable name** text box - Type text that will be used in the JavaScript code to identify the bound element.
- **Binds to** dropdown - allows selection of the element to be bound. For information about available binds, see the **JavaScript Binds** topic.
- **Parameters** picker - when required by the 'Binds to' selection, add the necessary selection by typing into the field to use the typeahead search functionality, or click the ellipsis button **(...)** to display the appropriate dialog. Use search or browse to select the item, then click the **Select** button. For example, when Attribute Value is selected for Binds to, the Select Attribute dialog is available.
- **X** button - removes the bind on the row.



5. Click the **OK** button to save the binds.
6. Add JavaScript code that includes the variable name for the bind. This is demonstrated below where the 'Node' variable name (which returns the Current Object) is used to get an attribute value on the current object.



Java vs. JavaScript

When writing JavaScript Business Rules it is important to understand that the public STEP API is a Java API. Thus all returned objects will be Java objects, not JavaScript objects.

For example, consider getting the length of the value for the attribute 'Description'. The following script will not work because the Java String returned by getSimpleValue() does not have a length property.

```
var length = node.getValue("Description").getSimpleValue().length;
```

To achieve this, you must use one of the following options:

- Use the Java length() method

```
var length = node.getValue("Description").getSimpleValue().length();
```

- Convert the value to a JavaScript string first

```
var value = node.getValue("Description").getSimpleValue() + "";
var length = value.length;
```

In general, be careful when making comparisons. As illustrated below, especially when using the 'strict' comparison operators, this can very easily lead to errors. The value for attribute with ID 'attID' on current object is '3' (validation base type will not matter as the value is output as a String), and the variable 'node' holds current object.

Notice the result of each test as indicated by the commented text (following //):

```
var value = node.getValue("attID").getSimpleValue();
logger.warning(value == 3); // WARNING: true
logger.warning(value.equals(3)); // WARNING: false
logger.warning(value.equals("3")); // WARNING: true
logger.warning(value === 3); // WARNING: false
logger.warning(value === "3"); // WARNING: false
logger.warning(value > 2); // WARNING: true
logger.warning(value <= 3); // WARNING: true
```

JavaScript Binds

The same JavaScript binds are available for all business rules, using the functionality exposed in the public Java API. JavaScript in a business rule will have access to the standard Java packages and connection into the STEP Java API can be created via binds where Java objects are bound to JavaScript variables. For more information, click the STEP API Documentation button on the WebStart page, and see the Javadoc link under the Extension API section.

The following documentation addresses the binds, regardless of their use. For steps required to add a JavaScript bind, see **Adding a Bind**.

Important: STEP data should only be changed via an Execute JavaScript business action. Changing STEP data via an Evaluate JavaScript business condition is not supported. See the **JavaScript Considerations** topic for more info and suggested uses.

The STEP Scripting Java API binds are available in the groupings listed below:

Bind	Resolves to
Approve Context Bind	
Approve Context	For business actions, instance of com.stibo.core.domain.spi.approve.ApproveTrigger.ApproveTriggerContext and the type specific sub interface (e.g. com.stibo.core.domain.spi.approve.ApproveProductTrigger.ApproveProductTriggerContext). For business conditions, instance of com.stibo.core.domain.spi.approve.ApprovePlugin.ApproveContext.
Asset Related Bind	
Asset Importer Configuration	com.stibo.assetimporter.domain.configuration.AssetImportConfiguration.
Attribute Related Binds	
Attribute	Instance of com.stibo.core.domain.Attribute for the selected attribute.
Attribute Group	Instance of com.stibo.core.domain.AttributeGroup for the selected attribute group.
Attribute Validated Parameters	User input as java.lang.String.
Attribute Value	java.lang.String value for the selected attribute.

Bind	Resolves to
List of Values	Instance of com.stibo.core.domain.ListOfValues for the selected list of values.
Unit	Instance of com.stibo.core.domain.Unit for the selected unit.
Unit Group	Instance of com.stibo.core.domain.UnitGroup for selected unit group.
Business Rules	
Business Function	Instance of com.stibo.core.domain.businessrule.businessfunction.BusinessFunction for the selected business function.
Current Object Bind	
Current Object	Instance of com.stibo.core.domain.Node and the sub interface for the type that the business rule is evaluated/executed against. E.g. com.stibo.core.domain.Product or com.stibo.core.domain.entity.Entity.
Data Container	
Current Data Container Object	Instance of com.stibo.core.domain.datacontainerobject.DataContainerObject.
Pair Of Attribute Values	Instance of com.stibo.core.domain.value.simplevaluepair.SimpleValuePair.
e-Signature Related	
Basic e-Signature	Instance of com.stibo.basicesignature.domain.BasicEsignature.
Event Binds	
Current Event Batch	Instance of com.stibo.outbound.businessactionpreprocessor.EventBatch.
Current Event Queue	Instance of com.stibo.core.domain.eventqueue.EventQueue.
Current Event Type	Instance of com.stibo.core.domain.eventqueue.SimpleEventType. Can be a derived event type (com.stibo.core.domain.eventqueue.DerivedEventType) or a basic (core) event type (com.stibo.core.domain.eventqueue.BasicEventType).
Event Queue	Instance of com.stibo.core.domain.eventqueue.EventQueue for the selected event queue, outbound integration endpoint or event processor.

Bind	Resolves to
Event Type	Instance of com.stibo.core.domain.eventqueue.DerivedEventType for the selected derived event type.
Gateway Integration Endpoint Bind	
Gateway Integration Endpoint	Instance of com.stibo.gateway.rest.REST given that the selected gateway integration endpoint is REST based (currently the only option).
GDSN Binds	
GDSN Product	Instance of com.stibo.gdsn2.domain.model.GDSNProduct.
GDSN Provider Datapool	Instance of com.stibo.gdsn2.domain.model.configuration.GDSNProviderDatapool.
GDSN Provider Message Exception Description	java.lang.String.
GDSN Provider Message Exception File	java.io.File
GDSN Provider Message Original File	java.io.File
GDSN Provider Original Node	Instance of com.stibo.core.domain.Node
GDSN Publisher Data Map	Instance of com.stibo.gdsn2.domain.datamap.GDSNDataMap.
GDSN Publisher Product Validation	Instance of com.stibo.gdsn2.domain.impl.businessrules.GDSNValidationContext @Deprecated.
GDSN Receiver Data Map	Instance of com.stibo.gdsn2.receiver.domain.impl.businessactioncontext.GDSNReceiverDataMapContext.
GDSN Receiver Message Exception	java.lang.String.

Bind	Resolves to
Description	
GDSN Receiver Message Exception File	java.io.File.
GDSN Receiver Message Original File	java.io.File.
GDSN Receiver Original Node	Instance of com.stibo.core.domain.Node.
GDSN Receiver Packaging Product	Instance of com.stibo.core.domain.Node.
GDSN Recipient	Instance of com.stibo.gdsn2.domain.model.configuration.GDSNRecipient.
GDSN Target Market	Instance of com.stibo.gdsn2.domain.model.configuration.GDSNTargetMarket.
Mongo DB Binds	
JSON Document	A JSON object for the document that has been stored in the MongoDB database.
MongoDB Context	Instance of com.stibo.components.mongodbadapter.server.MongoDBActionContext.
Object Aspects Binds	
ID	ID of the com.stibo.core.domain.Node that the business rule is evaluated/executed against as a java.lang.String.
Name	Name of the com.stibo.core.domain.Node that the business rule is evaluated/executed against as a java.lang.String.
Other Binds	
Conditionally Invalid Values	java.util.Set<com.stibo.core.domain.Value>
Import Change Info	Instance of com.stibo.core.domain.importer.changeinfo.ImportChangeInfo.

Bind	Resolves to
Logger	Instance of java.util.logging.Logger.
Lookup Table Home	com.stibo.lookuptable.domain.LookupTableHome.
Mail Home	com.stibo.mail.home.MailHome.
Reference Type Bind	
Reference Type	Instance of com.stibo.core.domain.ReferenceType for the selected reference type.
Secondary Object Bind	
Secondary Object	Instance of either com.stibo.core.domain.Product or com.stibo.core.domain.entity.Entity.
STEP Manager Bind	
STEP Manager	Instance of com.stibo.core.domain.Manager.
Survivorship Rule Source Objects Bind	
Survivorship Rule Source Objects	java.util.Set<com.stibo.core.domain.entity.Entity>
Tree Object Binds	
Asset	Instance of com.stibo.core.domain.Asset for the selected asset.
Classification	Instance of com.stibo.core.domain.Classification for the selected classification.
Entity	Instance of com.stibo.core.domain.entity.Entity for the selected entity.
Product	Instance of com.stibo.core.domain.Product for the selected product.
User Binds	
User	Instance of com.stibo.core.domain.User for the selected user.

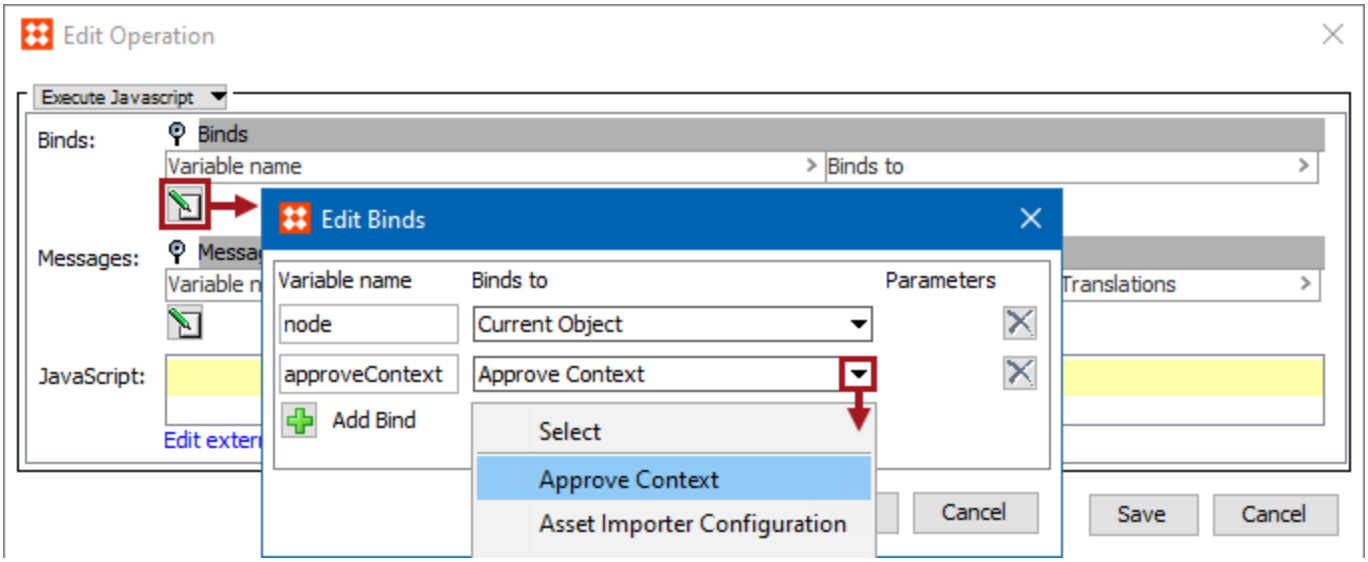
Bind	Resolves to
User Group	Instance of com.stibo.core.domain.Group for the selected user group.
Workflow Binds	
Current Transition	Instance of com.stibo.core.domain.state.currenttransition.CurrentTransition.
Current Workflow	Instance of com.stibo.core.domain.state.Workflow.
Workflow Parameters	Map<java.lang.String, java.lang.String>
Workflow State	Instance of com.stibo.core.domain.state.State.

The Global Binds available in Matching Algorithms are different from business rule JavaScript binds. For more information about matching algorithm binds, see the **Configuring Match Codes** section of the **Matching, Linking, and Merging** documentation.

Approve Context Bind

Business rules tested via conditions or executed via actions during approval have access to the Approve Context bind. The Approve Context bind is applicable before approval or on approval. Its use varies between actions and conditions. See the **Business Conditions** and **Business Actions** sections below for details and the available methods.

The bind can be found within the 'Binds to' dropdown, as shown below.



Configuration

To use any bind:

1. Create a business rule as defined in the **Creating a Business Rule or Library** topic.
2. Edit the business rule as defined in the **Editing a Business Rule** topic.
3. In the Edit Operation dialog, add the bind to a business rule, as defined in the **Adding a Bind** topic.
4. In the Edit Operation dialog, optionally add Messages, as defined in the **Adding a Localized Business Rule Message** topic.
5. In the Edit Operation dialog, add JavaScript to call the bind.

JavaScript Methods

The interface has the following methods that can compare data in Main or Approved workspaces, and can also access the set of part objects to be synchronized:

```
getApprovedManager()

getApprovedNodeBeforeApproval()

getApprovedNode()

getMainManager()
```

```
getMainNode()
getPartObjects()
```

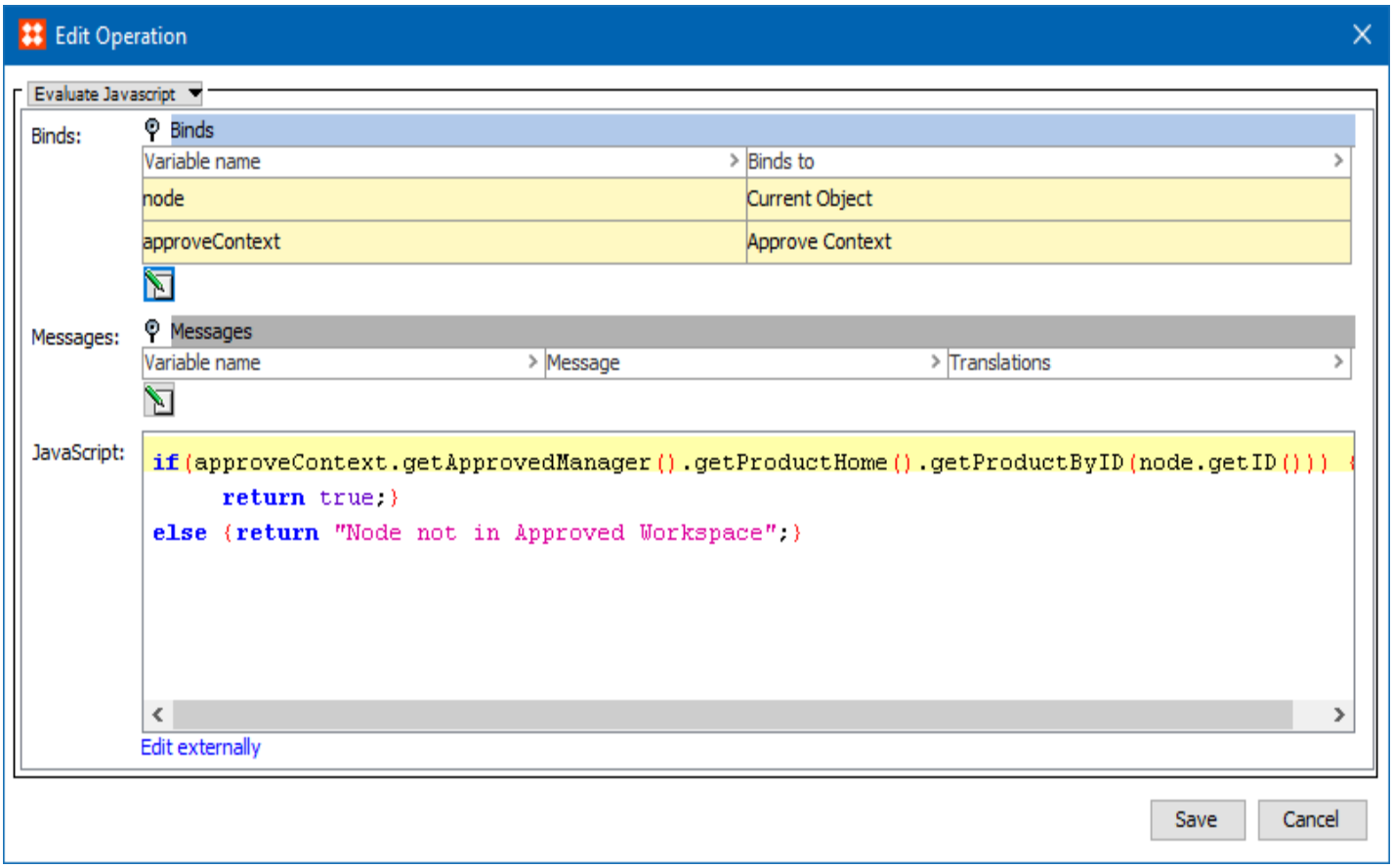
Note: Business rules used in OIEPs or Event Processors as event filters or event generators run after approval (when required). Use the 'getApprovedNodeBeforeApproval()' method to inspect the node as it looked prior to the approval, or use the 'getApprovedNode()' method to return the node after approval.

Business Condition

When used in a Business Condition On Approval, the bound class is 'ApprovePlugin.ApproveContext'.

When called Before Approval, the following code checks if the current node is already in the approved workspace. It evaluates to false if the node is not in the approved workspace.

For a full JavaScript code example, see the online version of this topic.



Be aware that the condition After Approval check is carried out in the Approved workspace after data has been synchronized. In this case, the getApprovedNode() method would return the object post synchronization.

For standard approval condition checks, you will likely not have to use the Approve Context at all. Thus, a simple value check, like the Current Object bound as 'node' below, will automatically resolve to the Main node Before Approval and the Approved node (post synchronization) After Approval.

```
var ean = node.getValue("EAN").getSimpleValue();
```

```
return ean != null && ean.length() == 13 ? true : "EAN must have 13 digits";
```

Business Action

When used in a Business Action, the bound class is 'ApproveTrigger.ApproveTriggerContext' and can be used when running On Approve.

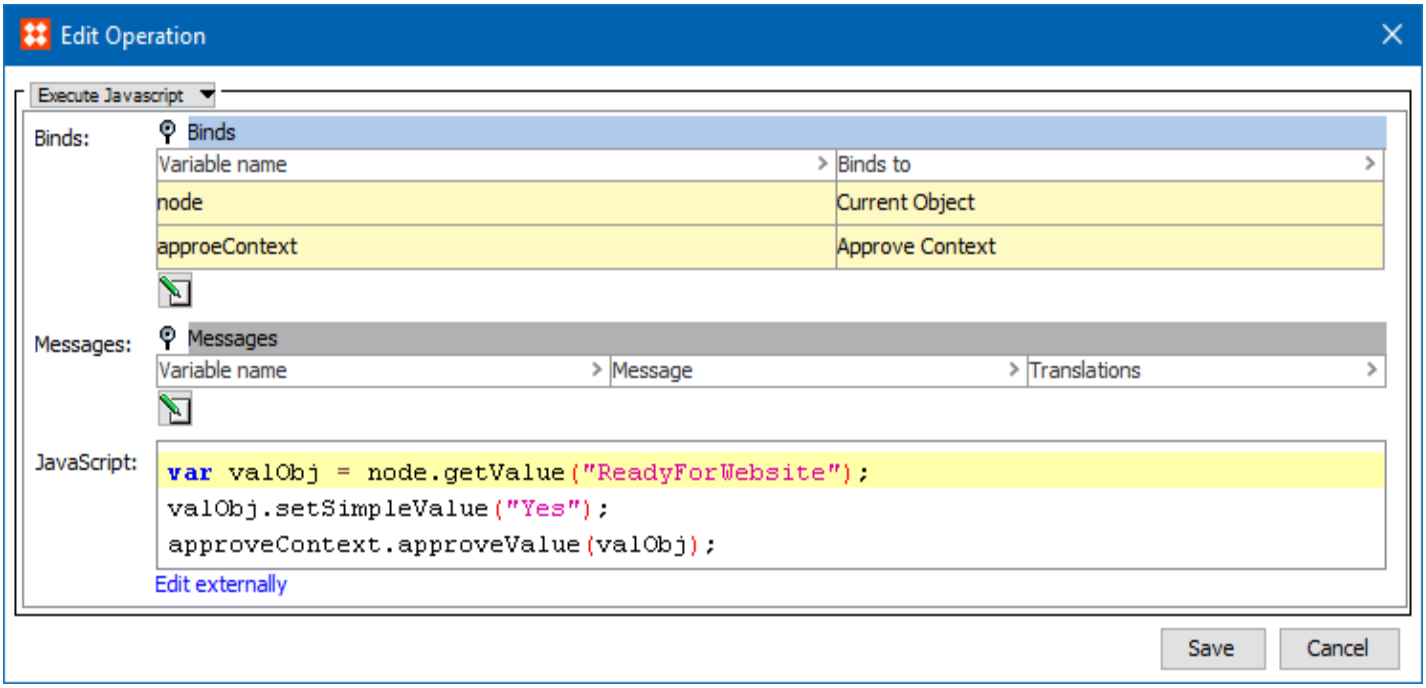
Example

The following is example JavaScript that uses this bind.

Important: The example scripts should not be used as-is without thorough testing, including updating the script to match object and link types that exist on your system.

In this example, in actions executed during approval, the Approve Context binding gives access to a super type-specific sub-interface for products: ApproveProductTrigger.ApproveProductTriggerContext. Using these sub-interfaces you can add part objects changed by an action during approval to the set of part objects to be approved as shown below:

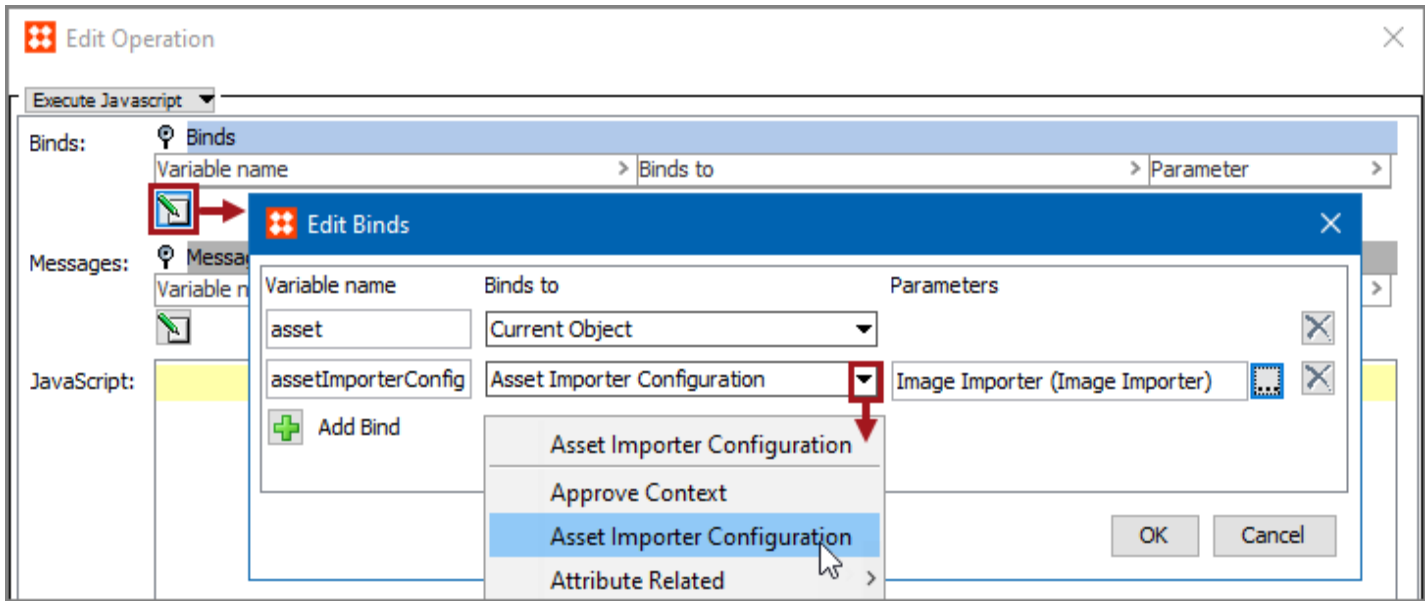
```
var valObj = node.getValue("ReadyForWebsite");
valObj.setSimpleValue("Yes");
approveContext.approveValue(valObj);
```



Note: Actions executed On Approval can have changes synchronized automatically in the Business Rule Editor 'On Approve' action parameter by checking the 'Approve changes to current object' checkbox. For more information, see the **On Approve Parameter** section of the **Editing a Business Rule** documentation.

Asset Importer Configuration Bind

The Asset Importer Configuration bind is used to create / update asset content with new file content using an Asset Importer Configuration. The bind can be found within the 'Binds to' dropdown, as shown below.



Configuration

To use any bind:

1. Create a business rule as defined in the **Creating a Business Rule or Library** topic.
2. Edit the business rule as defined in the **Editing a Business Rule** topic.
3. In the Edit Operation dialog, add the bind to a business rule, as defined in the **Adding a Bind** topic.
4. In the Edit Operation dialog, optionally add Messages, as defined in the **Adding a Localized Business Rule Message** topic.
5. In the Edit Operation dialog, add JavaScript to call the bind.

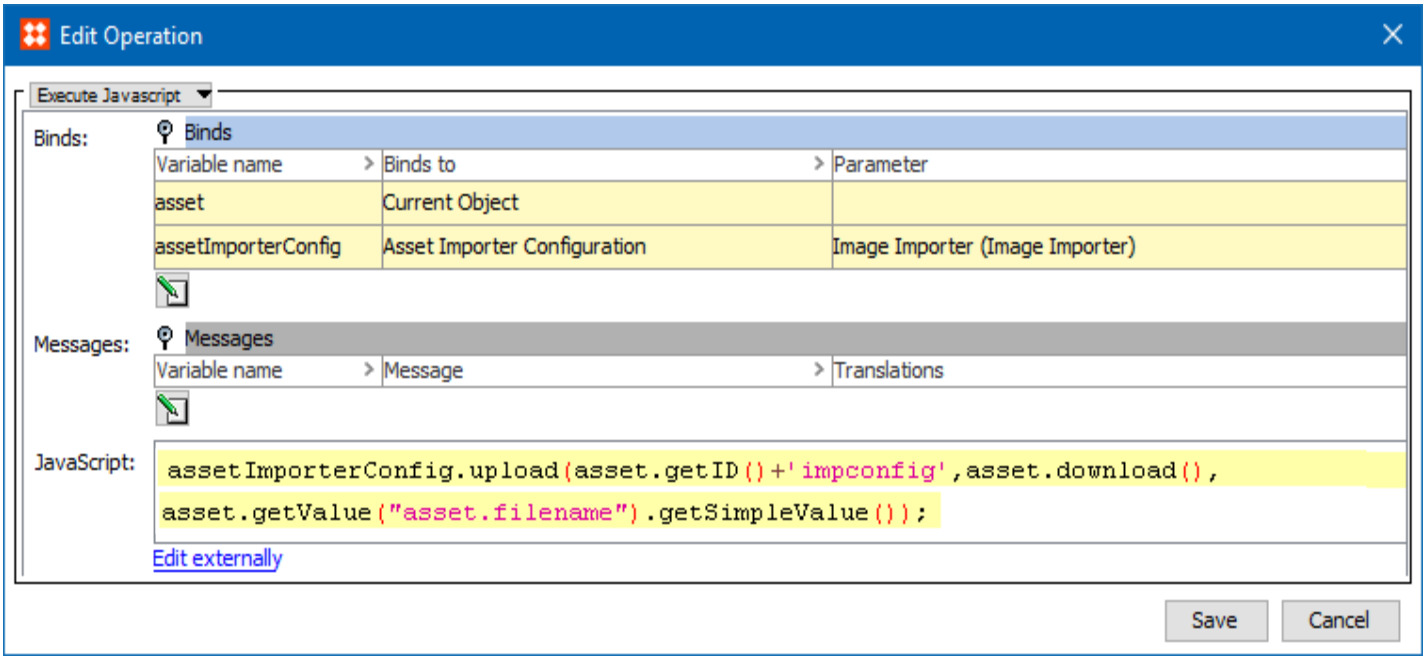
Example

The following is example JavaScript that uses this bind.

Important: The example scripts should not be used as-is without thorough testing, including updating the script to match object and link types that exist on your system.

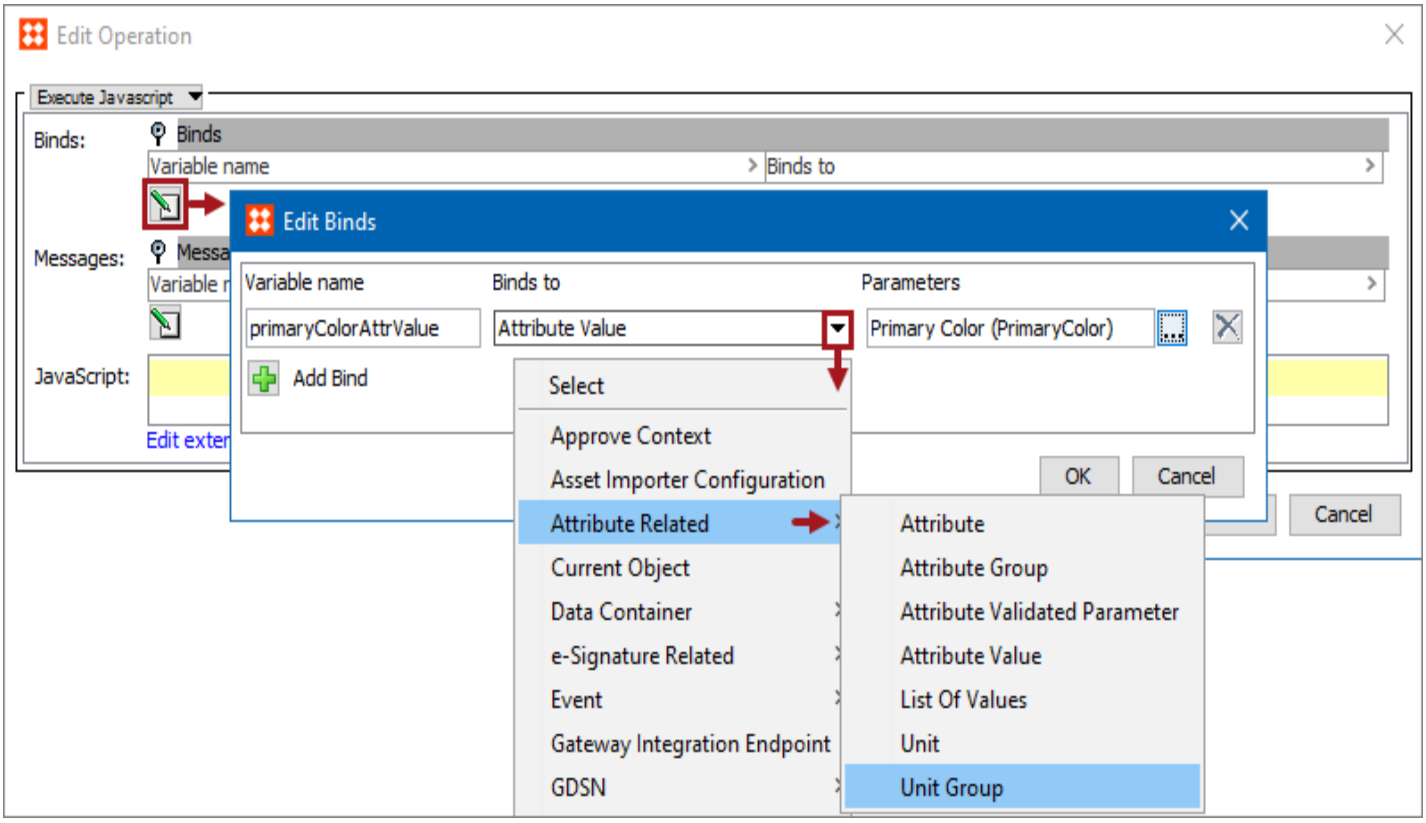
In the example below, the existing asset is updated via the Asset Importer Configuration so that the asset will be passed through all the steps of asset import configuration.

```
assetImporterConfig.upload(asset.getID()+'impconfig',asset.download(),
asset.getValue("asset.filename").getSimpleValue());
```


Edit Operation
Execute Javascript
Binds:
Binds
Variable name
Binds to
Parameter
asset
Current Object
assetImporterConfig
Asset Importer Configuration
Image Importer (Image Importer)
Messages:
Messages
Variable name
Message
Translations
JavaScript:
assetImporterConfig.upload(asset.getID()+'impconfig',asset.download(),
asset.getValue("asset.filename").getSimpleValue());
Edit externally
Save
Cancel

Attribute Related Binds

The Attribute Related binds allow access to the selected object or value. Many binds can be attribute related, however the binds listed below are available after selecting the 'Attribute Related' option within the 'Binds to' dropdown, as shown below.



Each bind is defined in the sections below.

Configuration

To use any bind:

1. Create a business rule as defined in the **Creating a Business Rule or Library** topic.
2. Edit the business rule as defined in the **Editing a Business Rule** topic.
3. In the Edit Operation dialog, add the bind to a business rule, as defined in the **Adding a Bind** topic.
4. In the Edit Operation dialog, optionally add Messages, as defined in the **Adding a Localized Business Rule Message** topic.
5. In the Edit Operation dialog, add JavaScript to call the bind.

Note: Example scripts should not be used as-is without thorough testing, including updating the script to match object and link types that exist on your system

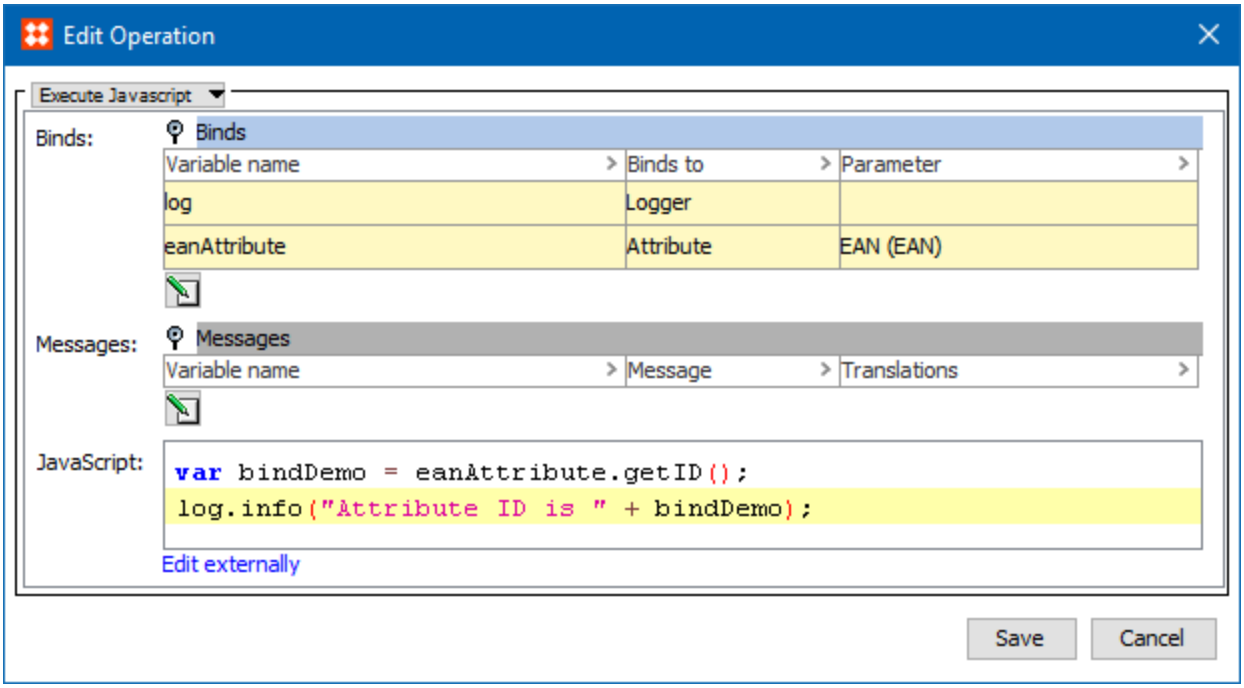
---

Attribute

The selected attribute is bound to the variable.

In this example, the Variable Name is 'eanAttribute', the Binds to field is Attribute, and the Parameters field displays the EAN attribute. The following JavaScript returns the ID of the EAN attribute as an Info log message:

```
var bindDemo = eanAttribute.getID();
log.info("Attribute ID is " + bindDemo);
```

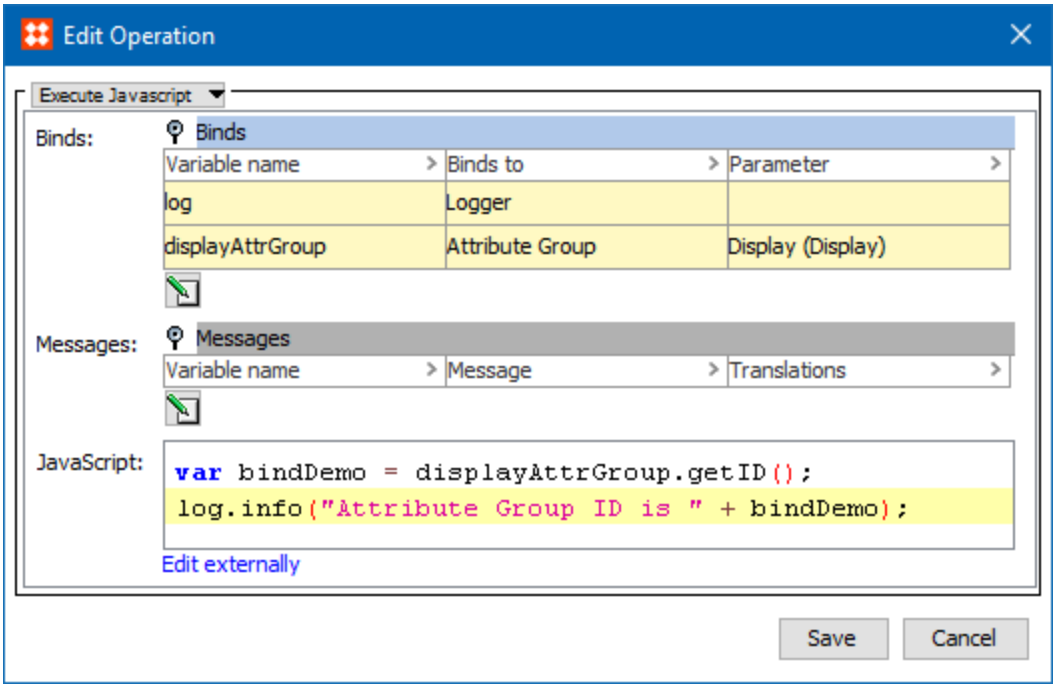


Attribute Group

The selected attribute group is bound to the variable.

In this example, the Variable Name is 'displayAttrGroup', the Binds to field is Attribute Group, and the Parameters field shows the Display attribute. The following JavaScript returns the ID of the Display attribute group as an Info log message:

```
var bindDemo = displayAttrGroup.getID();
log.info("Attribute Group ID is " + bindDemo);
```



Attribute Validated Parameter

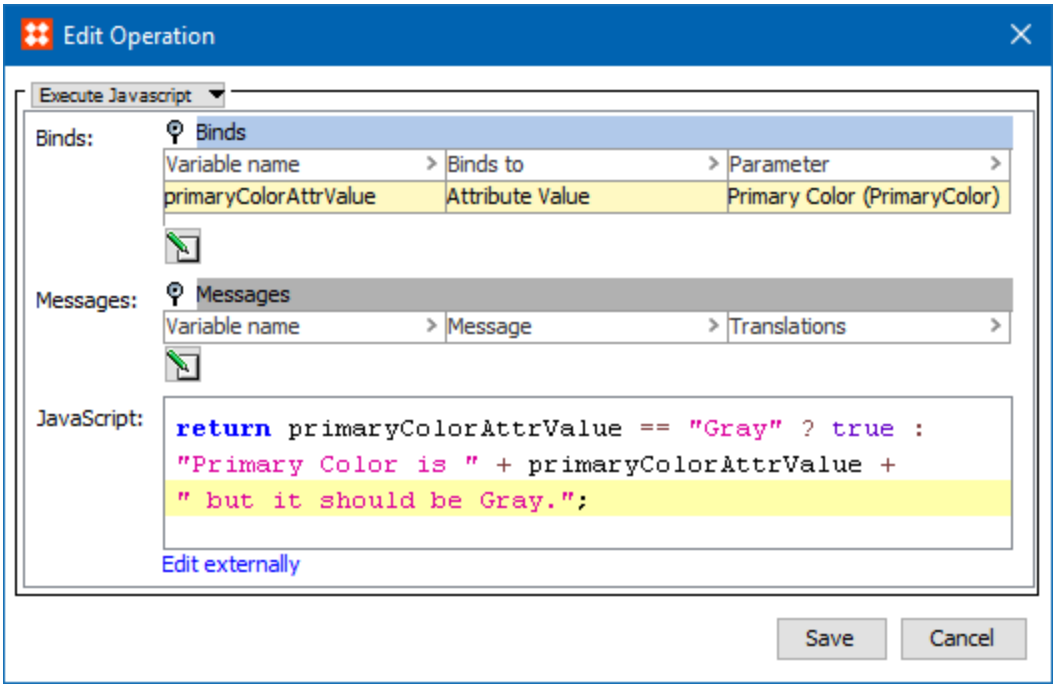
For more information, see the **Parameterized Business Actions in Web UI** section of the **Web UI Getting Started** documentation.

Attribute Value

The value of the selected attribute is bound directly as a String variable. Primarily used for Web UI live validation of conditions and when defining Match Codes.

Note: This variable also works in the Web UI.

In this example, the Variable Name is 'primaryColorAttrValue', the Binds to field is Attribute Value, and the Parameters field displays the Primary Color attribute. For the JavaScript example of the Attribute Value, see the online version of this topic.

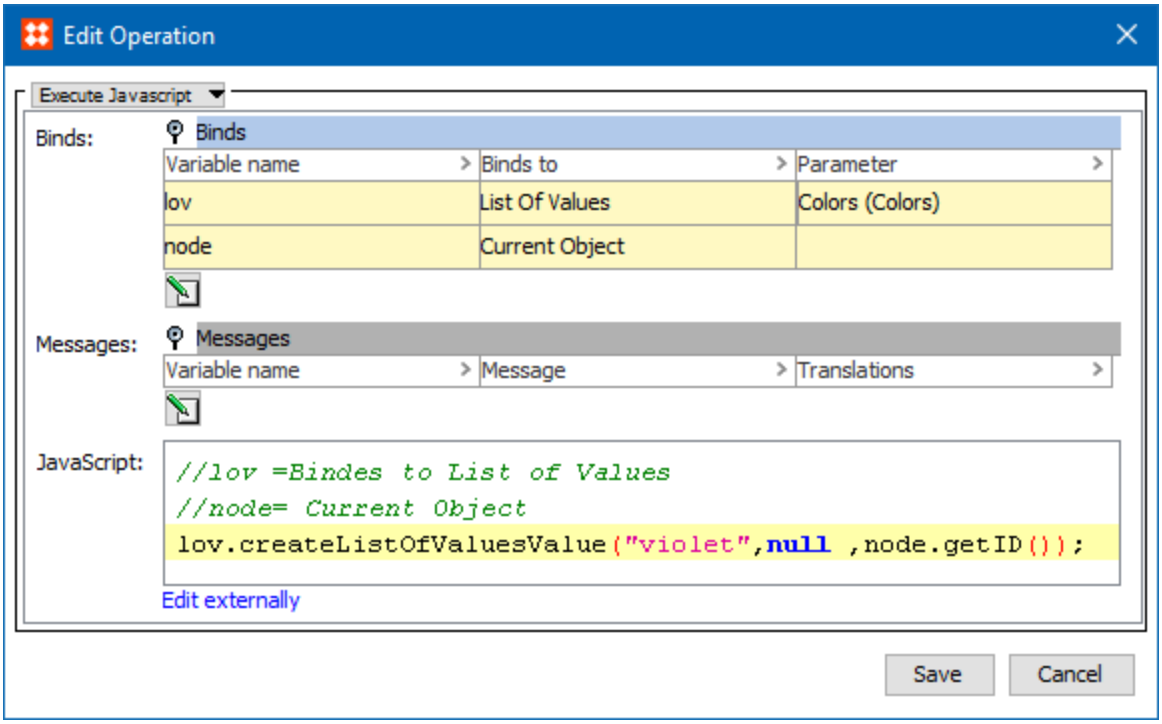


List of Values

The selected LOV (called 'Domain' in the workbench) is bound to the action or condition variable.

For example, to add one more color 'violet' to the 'Colors' LOV.

```
//lov =Bindes to List of Values
//node= Current Object
lov.createListOfValuesValue("violet",null ,node.getID());
```



Unit

The selected unit is bound to the action or condition variable.

For example, to add one more size 5 cm to the 'Size' LOV.

```
//lov= Binds to List of Values
//node=Current object
//unit= Select the unit which you want to bind (cm, m , g/cm^2 etc)
lov.createListOfValuesValue(5, unit ,node.getID());
```

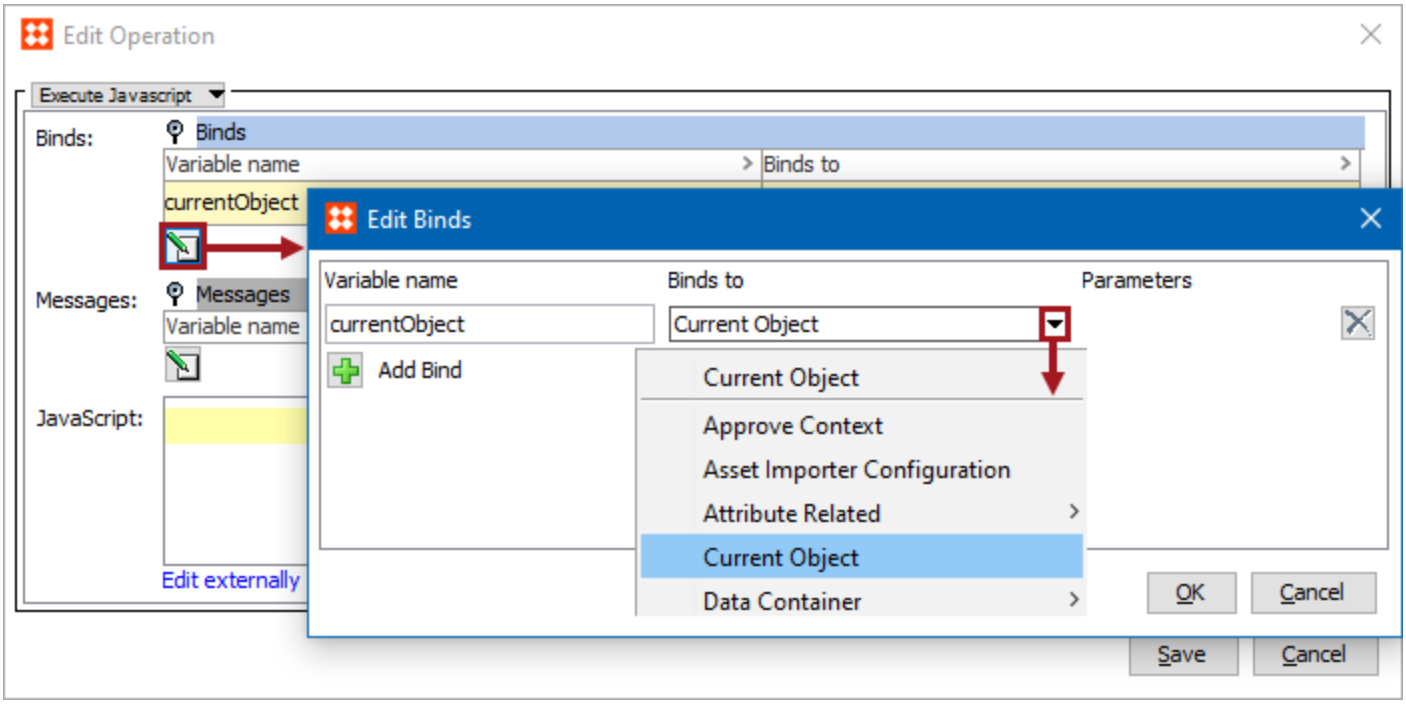
Unit Group

The selected unit group is bound to the action or condition variable.

Current Object Bind

The Current Object bind gives access to the STEP object that the business rule is being evaluated (via condition) or executed (via action) against. The bind can be found within the 'Binds to' dropdown, as shown below.

Note: If the Current Object is bound into JavaScript, the business rule cannot be used in the Web UI.



Configuration

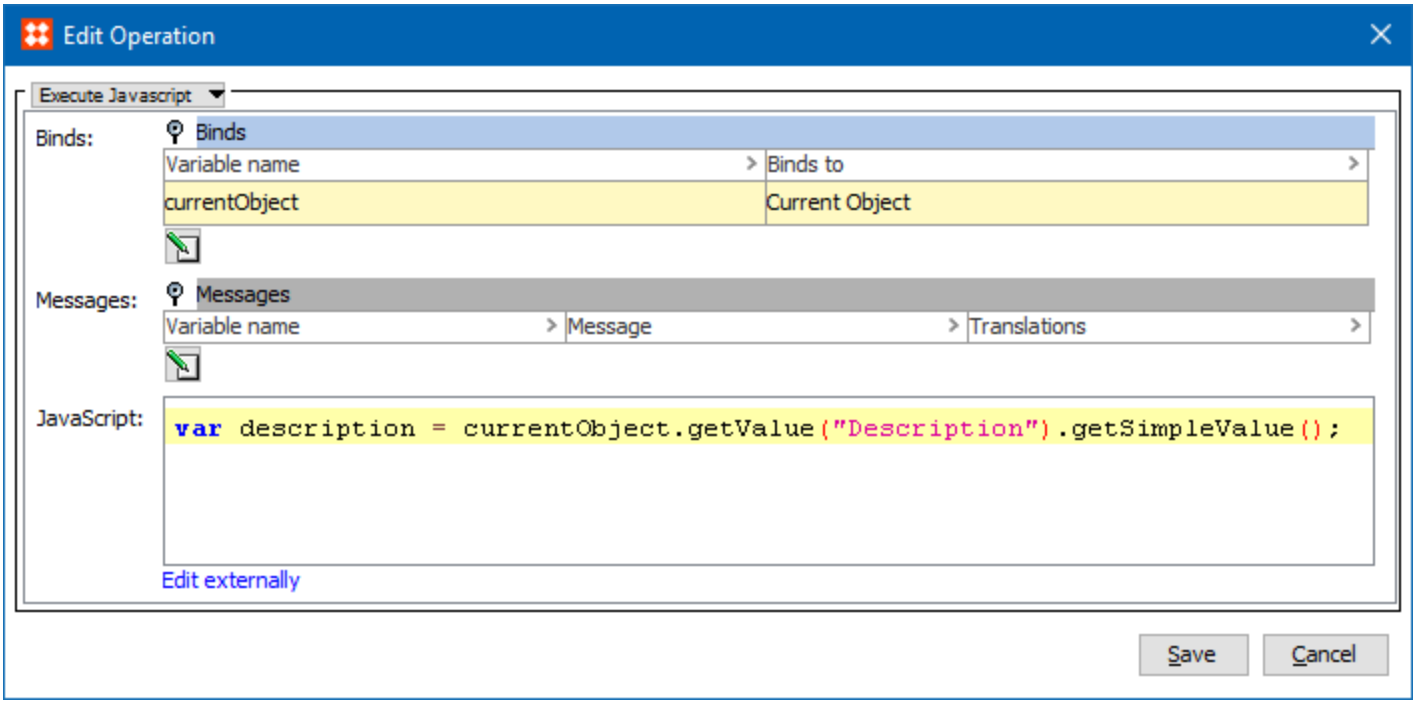
To use any bind:

1. Create a business rule as defined in the **Creating a Business Rule or Library** topic.
2. Edit the business rule as defined in the **Editing a Business Rule** topic.
3. In the Edit Operation dialog, add the bind to a business rule, as defined in the **Adding a Bind** topic.
4. In the Edit Operation dialog, optionally add Messages, as defined in the **Adding a Localized Business Rule Message** topic.
5. In the Edit Operation dialog, add JavaScript to call the bind.

Example

The following is example JavaScript that uses this bind.

Important: The example scripts should not be used as-is without thorough testing, including updating the script to match object and link types that exist on your system.



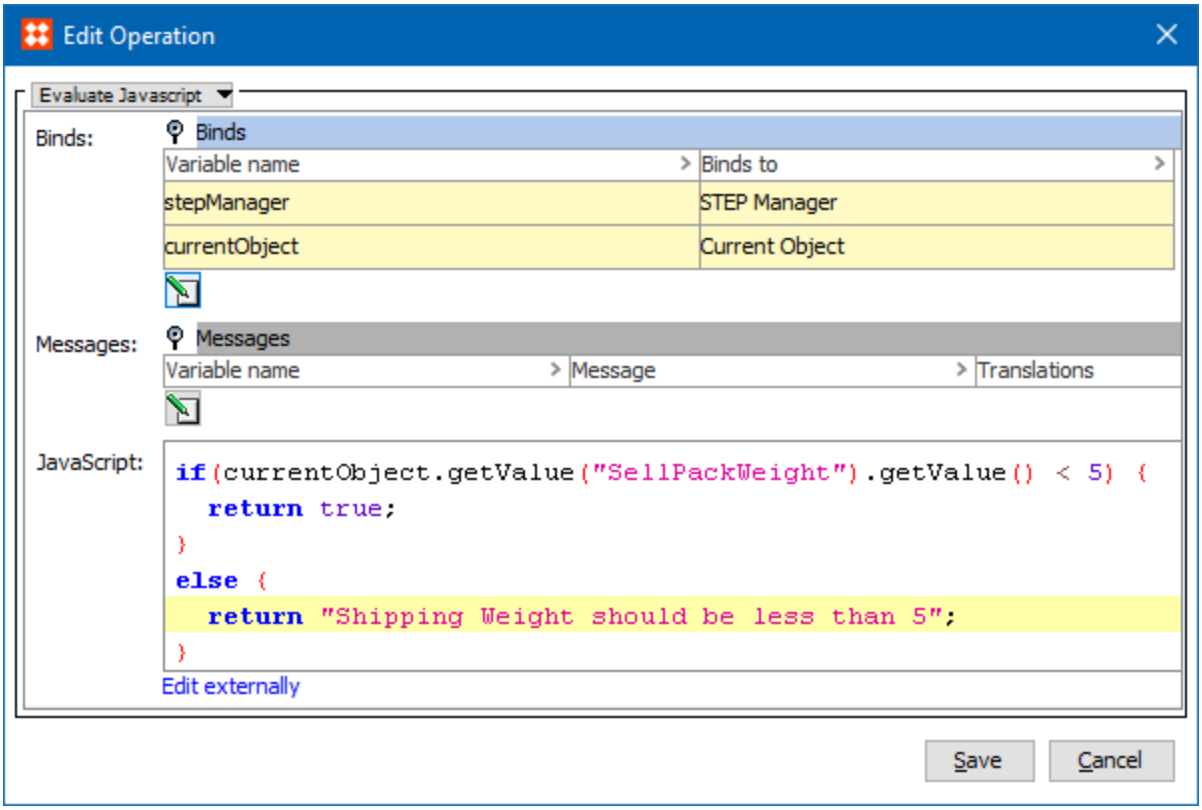
This sample code above expects a bind to currentObject, gets the value of the 'Description' attribute on the current object, and stores it in the variable 'Description'.

Condition Examples

The following condition samples expect a bind to Current Object with the variable 'currentObject' and a bind to STEP Manager with the variable 'stepManager'.

Simple Value Test

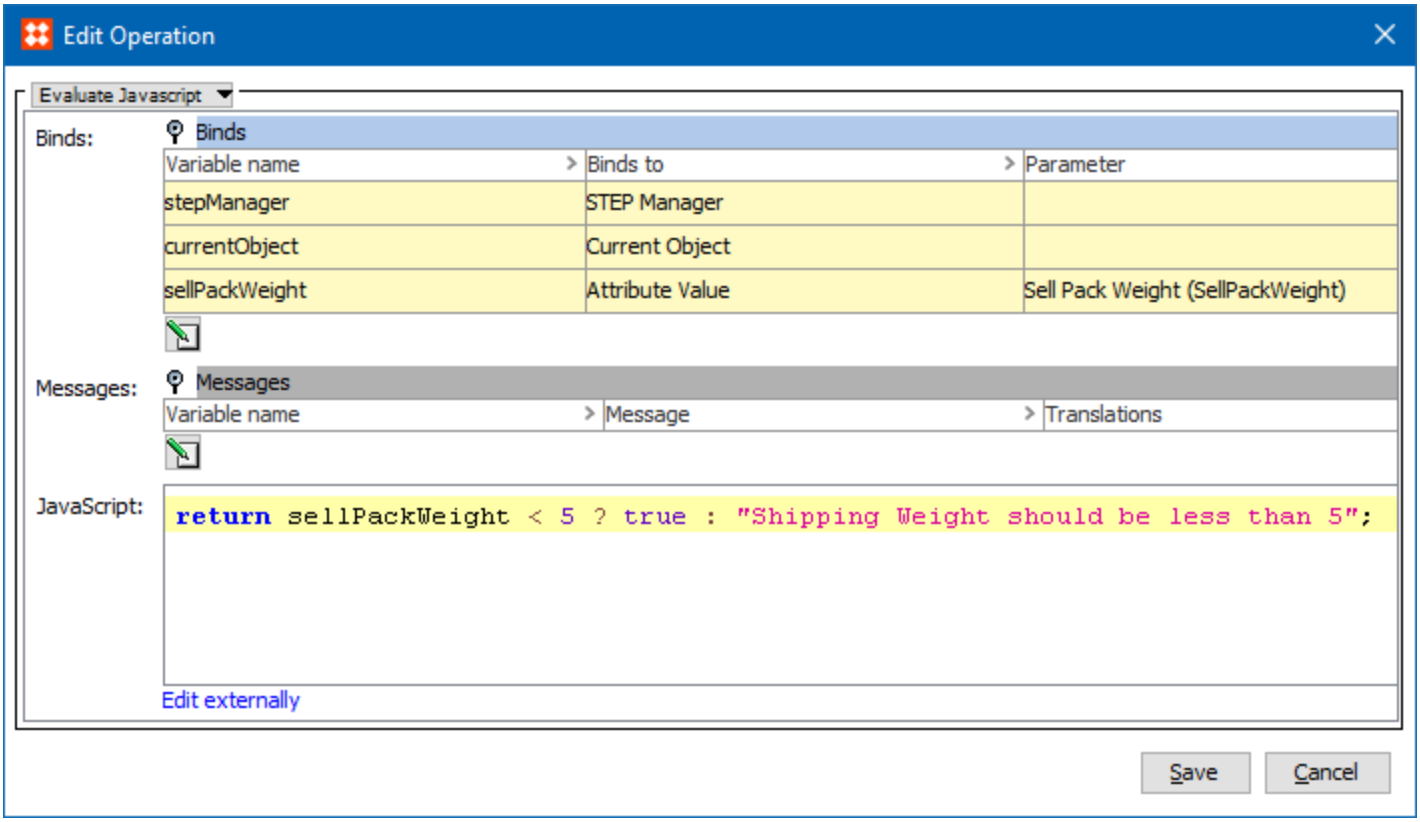
This condition checks the attribute 'SellPackWeight', and returns a message if the value is five or greater.



Using the same binds, the condition could alternatively be written as:

```
return currentObject.getValue("SellPackWeight").getValue() < 5 ? true :
"Shipping Weight should be less than 5";
```

With an additional bind of Attribute Value to the variable sellPackWeight (for attribute 'Sell Pack Weight'), the same output could be achieved as:



Condition for Object Type Test

This JavaScript returns true or false, dependent on whether the current object is of the Object Type = SalesItem:

```
return currentObject.getObjectType().getID() == "SalesItem";
```

Condition to Check Data on Referenced Objects

This JavaScript iterates over references of the type 'Part of Kit' and evaluates to true if none of the referenced objects has a shipping weight greater than or equal to five:

```
var refType =
stepManager.getReferenceTypeHome().getReferenceTypeByID("Accessory");
var refsList = currentObject.getReferences(refType);
var result = true;
for(var i = 0; i < refsList.size(); i++) {
 if(refsList.get(i).getTarget().getValue("SellPackWeight").getValue() > 5) {
   result = "Accessory with ID '" + refsList.get(i).getTarget().getID() + "' is
too heavy";
   break;
 }
}
return result;
```

Action Examples

The following action samples expect a bind to Current Object with the variable 'node' and a bind to STEP Manager with the variable 'step'.

Action to Set Value

The following sample sets the value for the 'Live' attribute as Yes.

```
currentObject.getValue("Live").setSimpleValue("Yes");
```

The following sample adds a reference to the object, checking first that the reference does not already exist (local or inherited).

```
var refType = stepManager.getReferenceTypeHome().getReferenceTypeByID
("Accessory");
var targetObj = stepManager.getProductHome().getProductByID("I-SalesItem-
1122");
var refsList = currentObject.getReferences(refType);
var refExists = false;
for(var i = 0; i < refsList.size(); i++) {
   if(refsList.get(i).getTarget().getID().equals(targetObj.getID())) {
      refExists = true;
      break;
   }
}
if (!refExists) {
currentObject.createReference(targetObj, refType);
}
```

Data Container Binds

Business rules can use any of the following Data Container binds.

Configuration

To use any bind:

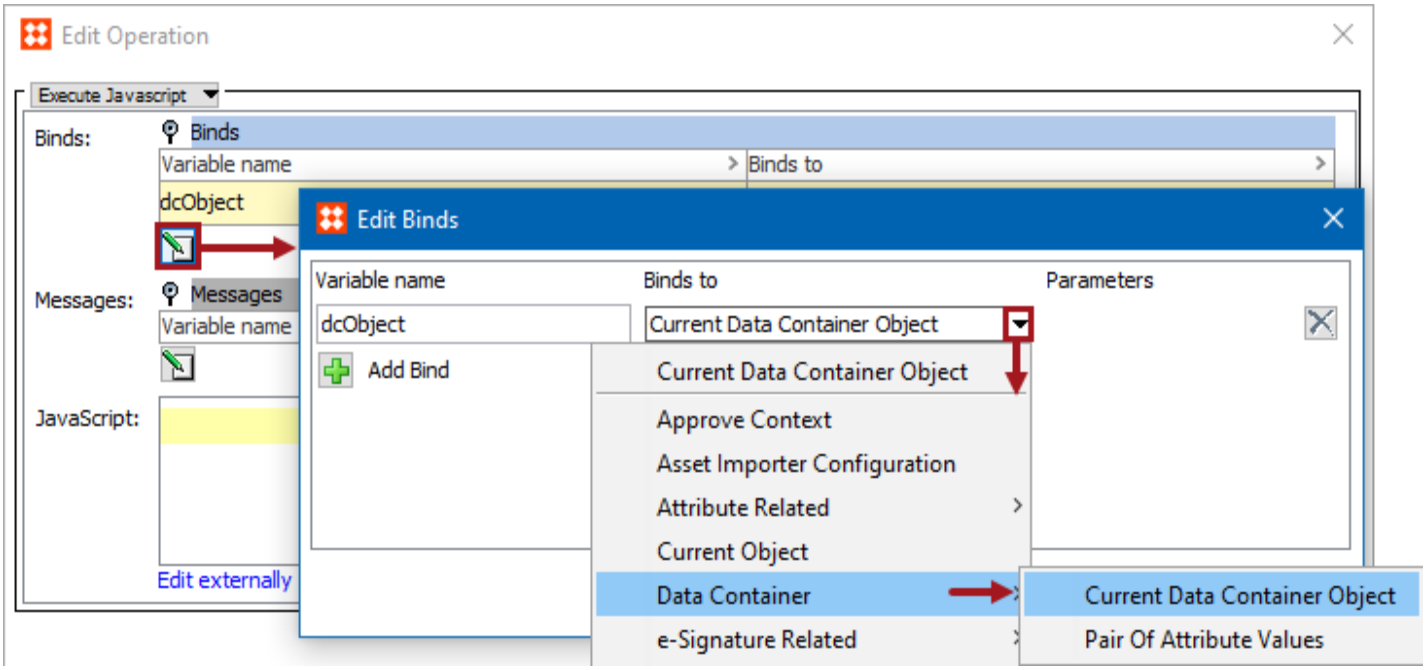
1. Create a business rule as defined in the **Creating a Business Rule or Library** topic.
2. Edit the business rule as defined in the **Editing a Business Rule** topic.
3. In the Edit Operation dialog, add the bind to a business rule, as defined in the **Adding a Bind** topic.
4. In the Edit Operation dialog, optionally add Messages, as defined in the **Adding a Localized Business Rule Message** topic.
5. In the Edit Operation dialog, add JavaScript to call the bind.

Important: The example scripts should not be used as-is without thorough testing, including updating the script to match object and link types that exist on your system.

Each bind is defined in the sections below.

Current Data Container Object

This bind is used in the 'Update Data Containers' Bulk Update operation, in the related JavaScript business conditions and actions. The selected data container is bound to the variable.



Condition

This JavaScript business condition is used to find entities with Data Containers that have an attribute 'City' with the value 'New York'.

```
var cityname = dc.getValue("City").getSimpleValue();

var fixedname = "New York";

    if (cityname == fixedname) {
        return true;

     }
```

Action

This JavaScript business action is then used to manipulate specified attributes on the Data Containers found.

```
dc.getValue("Street").setSimpleValue("(" + dc.getValue("Street").getSimpleValue
() + ")");
```

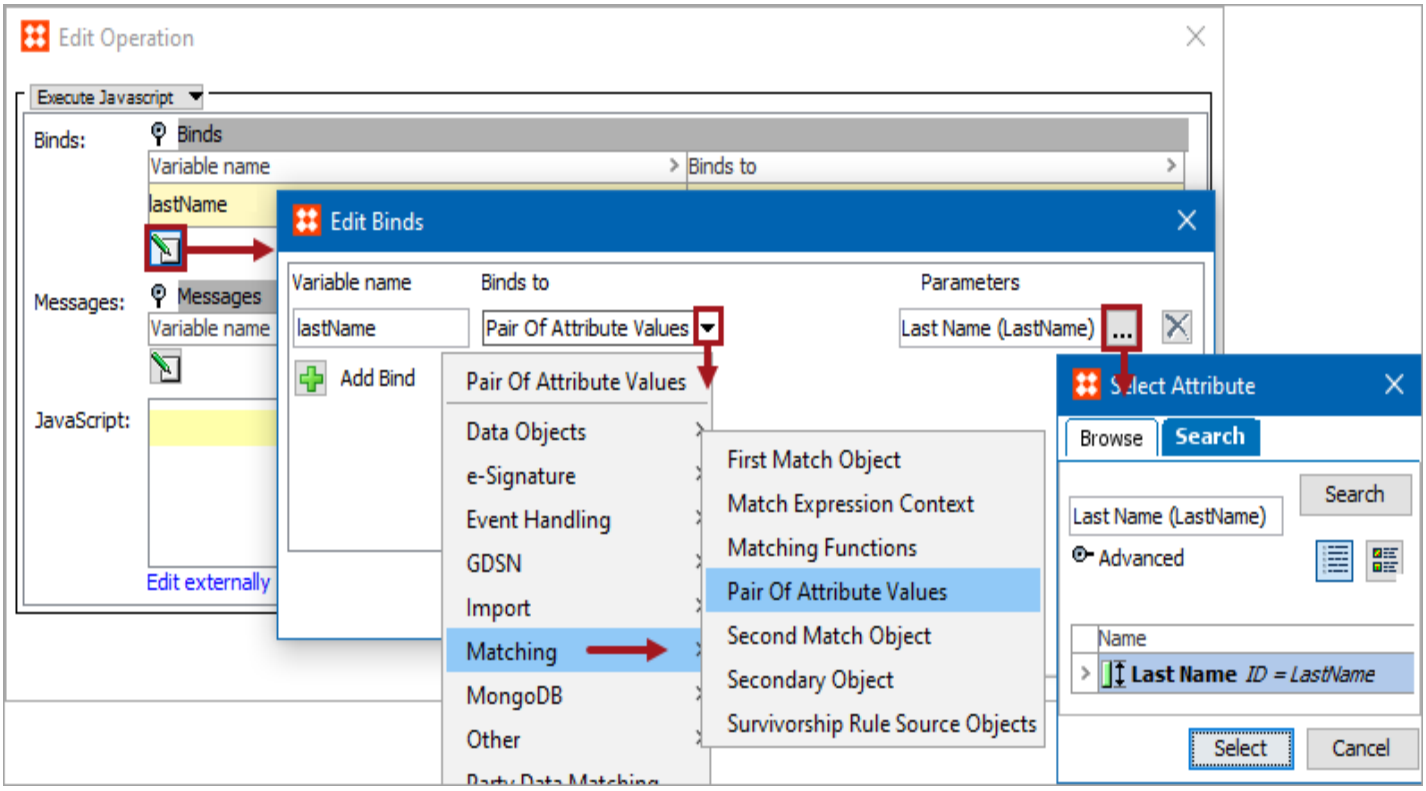
Pair of Attribute Values

By comparing attribute values, the Data Container survivorship rules can determine whether the data container instances on the source records are the same as on the data container instances on the golden records.

If a check of individual attributes between the source and golden records is required to determine when the source record data container should overwrite the golden record data container, the Pair of Attribute Variables bind can be created for each attribute to be compared.

This bind is in the Public API under com.stibo.core.domain.value.simplevaluepair.

The selected attribute is bound to the variable.



In this example, the Variable Name is 'email', the Binds to field is Pair of Attribute values, and the Parameters field displays the Email attribute used on a Data Container. The following JavaScript is used in a JavaScript Business Condition on the Data Containers Trusted Source or Most Recent Survivorship Rules, and compares the attribute values on the source object and golden record:

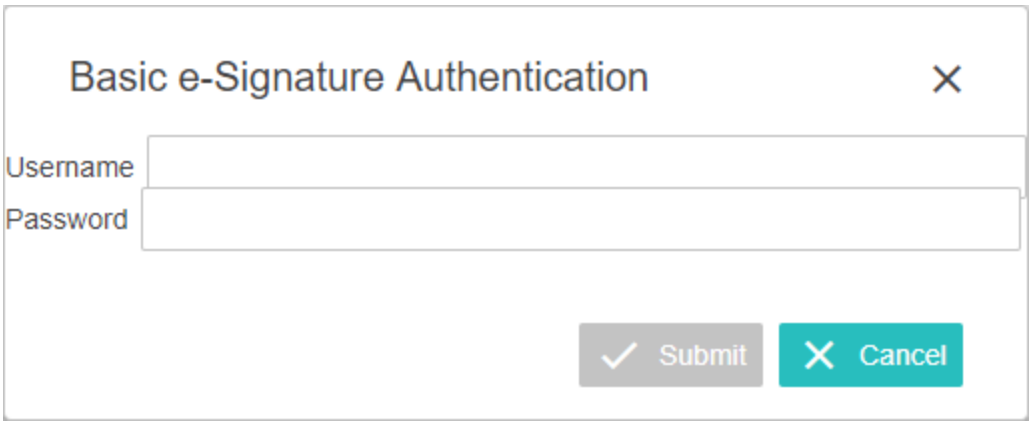
```
var s1 = email.getValue1();
var s2 = email.getValue2();
if(s1==s2){
   return true;
}
else{
   return "Error"
}
```

For more information on survivorship rules for data containers, see the **Golden Record Survivorship Rule Types** topic in the **Matching, Linking, and Merging** documentation.

e-Signature

Using e-Signature forces users to re-authenticate prior to taking action in Web UI, and ensures the security of the data being committed. It can be used in a global business action on a transition in a workflow.

With e-Signature, users are able to customize how they want the re-authentication of data to be captured. Some examples are storing the data at the instance it was captured for audit purposes, recording who authenticated and when, or configuring it to archive and submit the signed data as per customer and industry requirements. As standard global business action rules are used for e-Signature, it is fully at the discretion of the customer for how to implement the rule to ensure that the desired data capture and/or logging occurs. Using the standard e-Signature business rule action option simply prompts the re-authentication dialog to require that a user provide username and password login credentials before taking action.



Global business rules using e-Signature are applied in the STEP Workbench, however re-authentication using e-Signature is only available in Web UI. Users attempting to call a business rule using e-Signature in the workbench will receive an error.

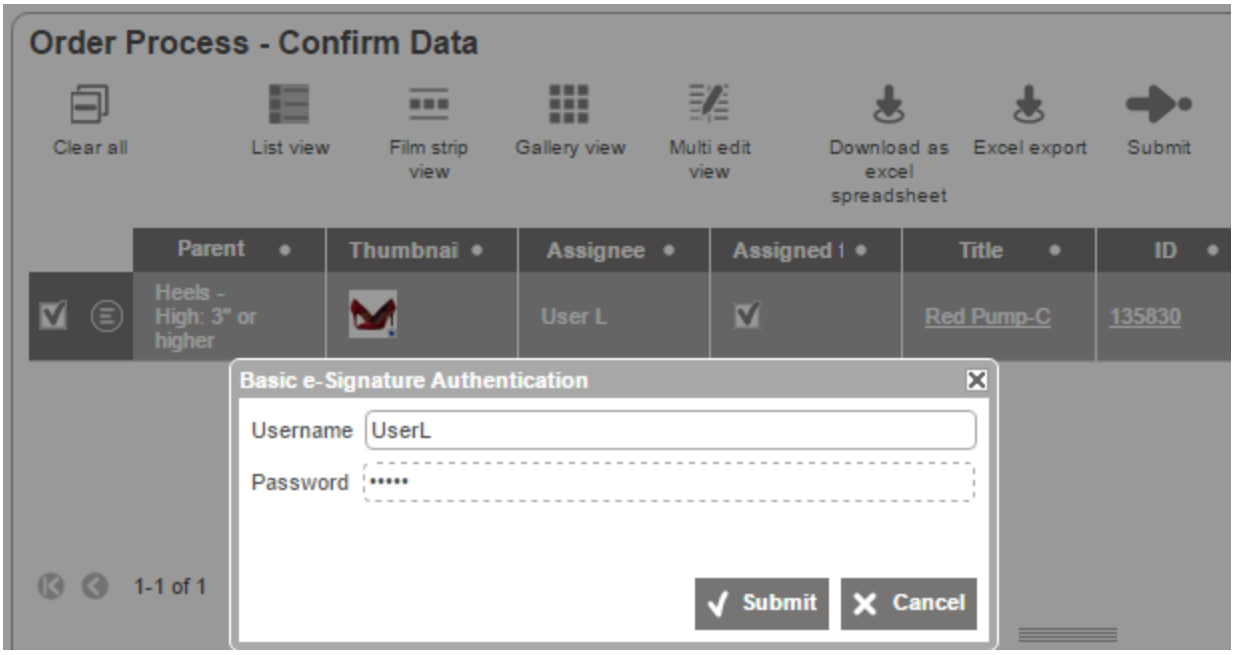
For more on how to configure e-Signature, see **Applying e-Signature to a Transition in a Workflow** in the **Business Rules** documentation.

For more on using e-Signature, see **Using e-Signature in Web UI** in the **Business Rules** documentation.

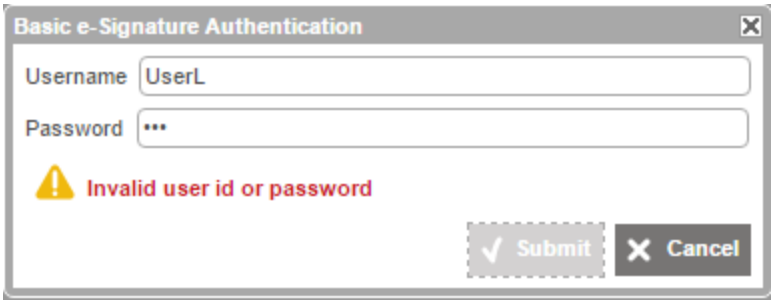
Using e-Signature in Web UI

e-Signature can be setup in workbench to be used on a transition in a workflow, though it can only be run in Web UI. Users attempting to call a business rule using e-Signature in the workbench will receive an error. Additionally, functionality cannot be used for any automated transition process or within any system processes that do not have direct user interaction (e.g. scheduled bulk update, etc).

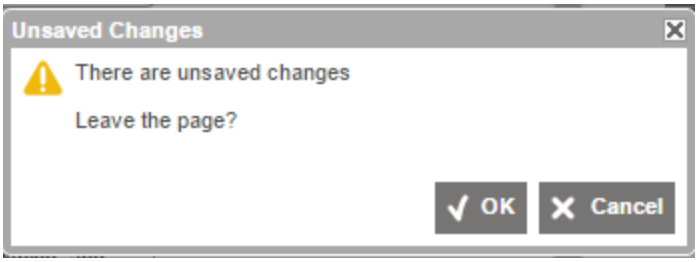
When used in Web UI, after the object is selected to move from one state to the next in the workflow, the re-authentication prompt appears, forcing the user to retype in their username and password.



This provides a more secure interface, as only the currently logged in user's username and password credentials, when properly entered, will allow the desired action to trigger and take place. If the incorrect username or password is entered, an error message will occur.



The user can then try and enter in the correct information and select submit, or press cancel. If the user presses cancel and then tries to leave the page, where an object has been edited without successfully re-authenticating, the user will be notified that there are unsaved changes



For more on how to configure e-Signature, see **Applying e-Signature to a Transition in a Workflow** in the **Business Rules** documentation.

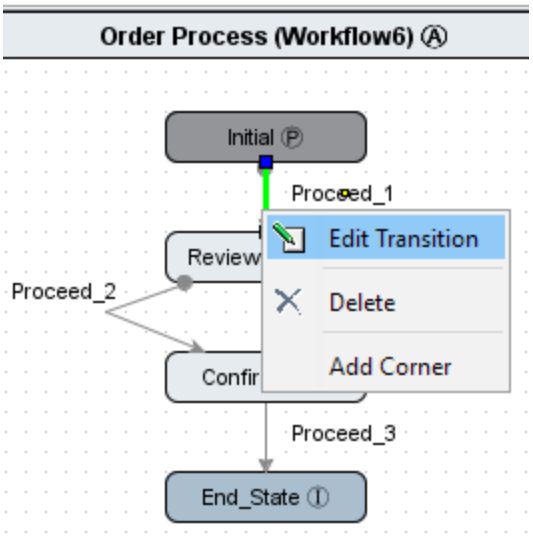
Applying e-Signature to a Transition in a Workflow

Configuring e-Signature sets the user re-authentication prompt in Web UI, though the setup is in workbench. Global business rules using e-Signature can be configured for transitions in a workflow. Users are able to customize how they want the re-authentication data to be captured using global business rules.

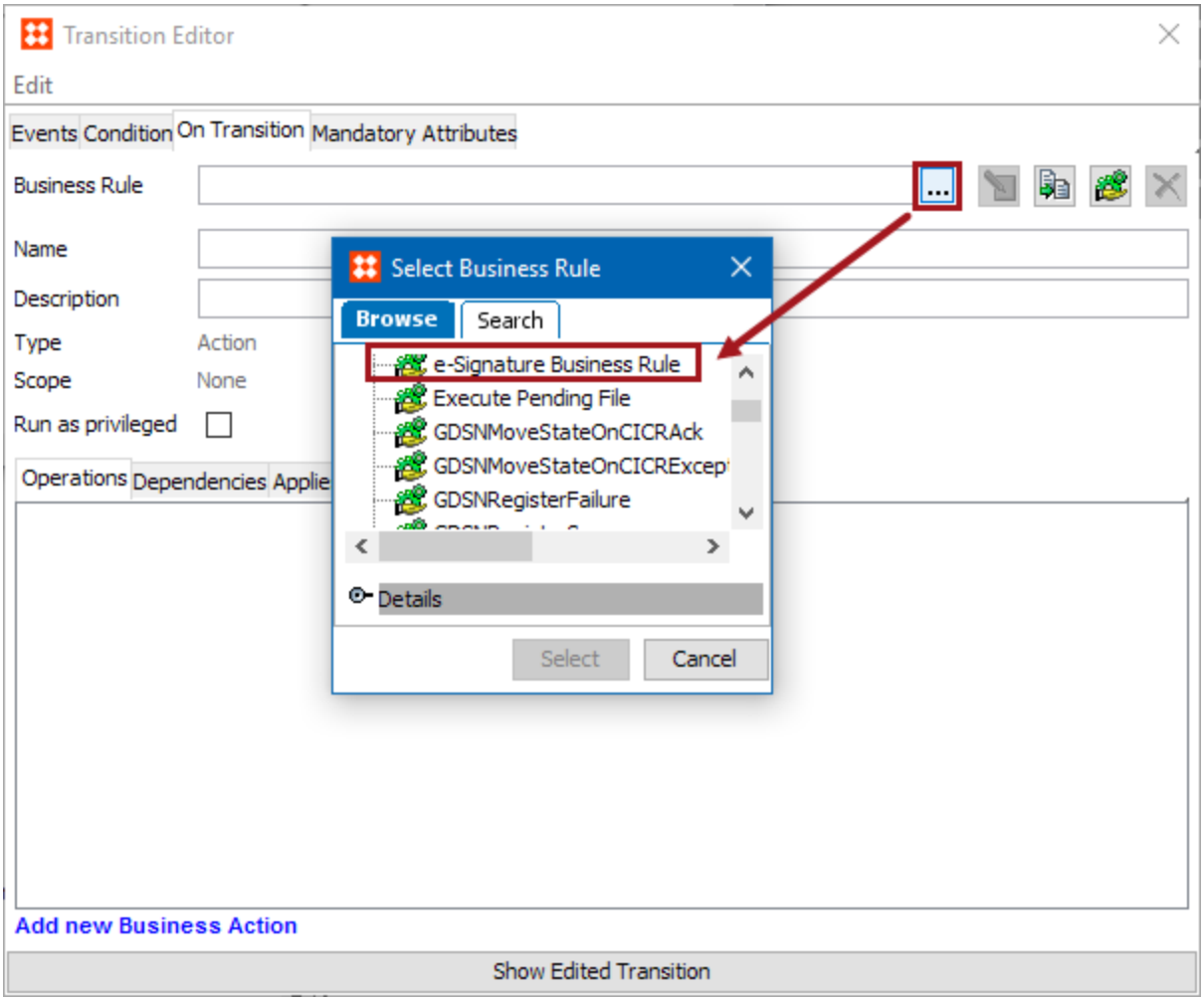
Note: Users attempting to call a business rule using e-Signature in the workbench will receive an error.

To add e-Signature to a transition in a workflow, follow the steps below:

1. In the STEP Workflow designer in workbench, right-click on the desired transition that needs an e-Signature global business action rule applied to it, and select **Edit transition**.

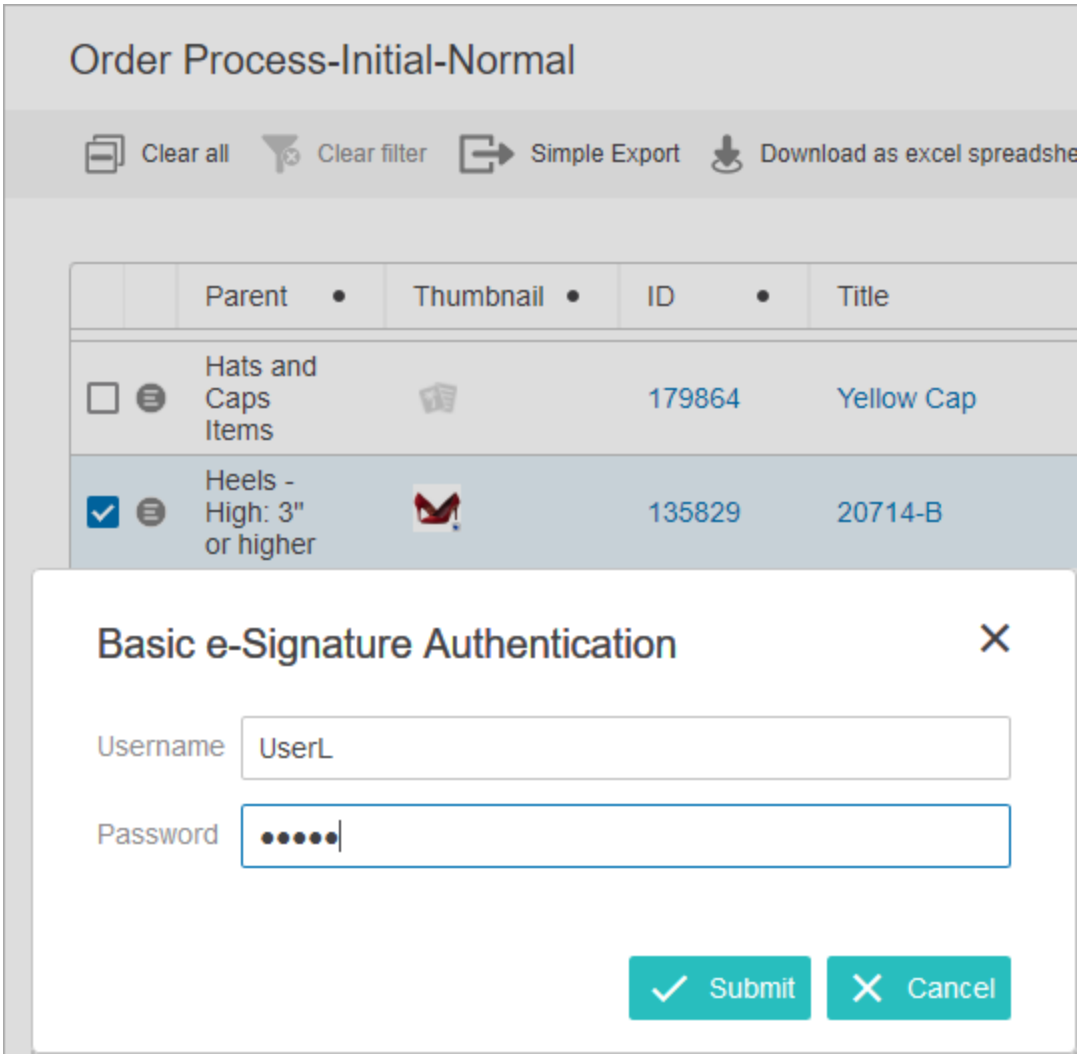


2. In the Transition Editor, select the On Transition tab and click the ellipsis button **(…)** next to the 'Business Rule' field. Select the appropriate e-Signature global business rule.



Note: Using e-Signature as a global business condition on a transition in a workflow is not supported. It is only supported as a business action.

Save and close the designer. Re-authentication and the capturing of data are now setup and ready to be used in Web UI.



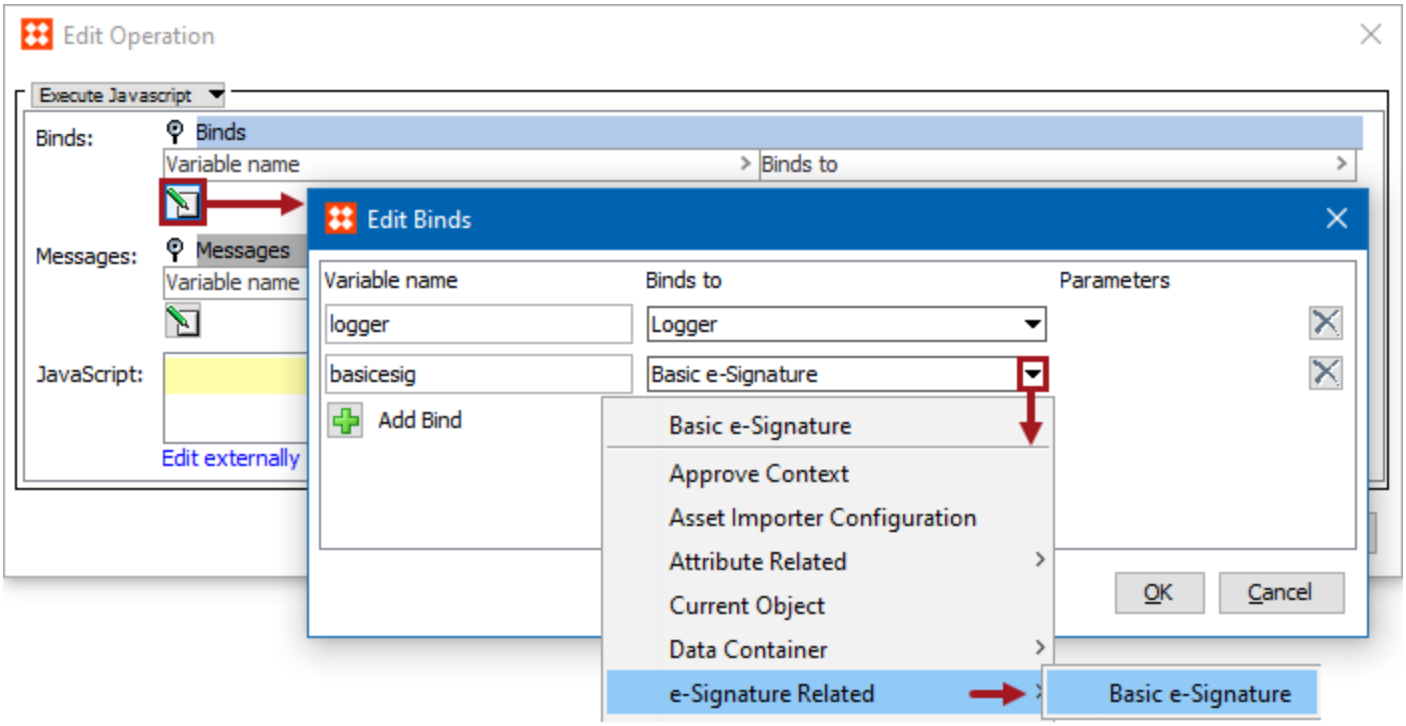
For information on business rules and how to create them, see the **Business Rules** documentation.

For an example of how to use e-Signature, see the **Using e-Signature in Web UI** topic in the **Business Rules** documentation.

For workbench configuration of e-signature using JavaScript, see the **Basic e-Signature Bind** in the **Business Rules** documentation.

Basic e-Signature Bind

The Basic e-Signature bind is used when re-authentication needs to occur. The bind can be found within the 'Binds to' dropdown, as shown below.



Configuration

To use any bind:

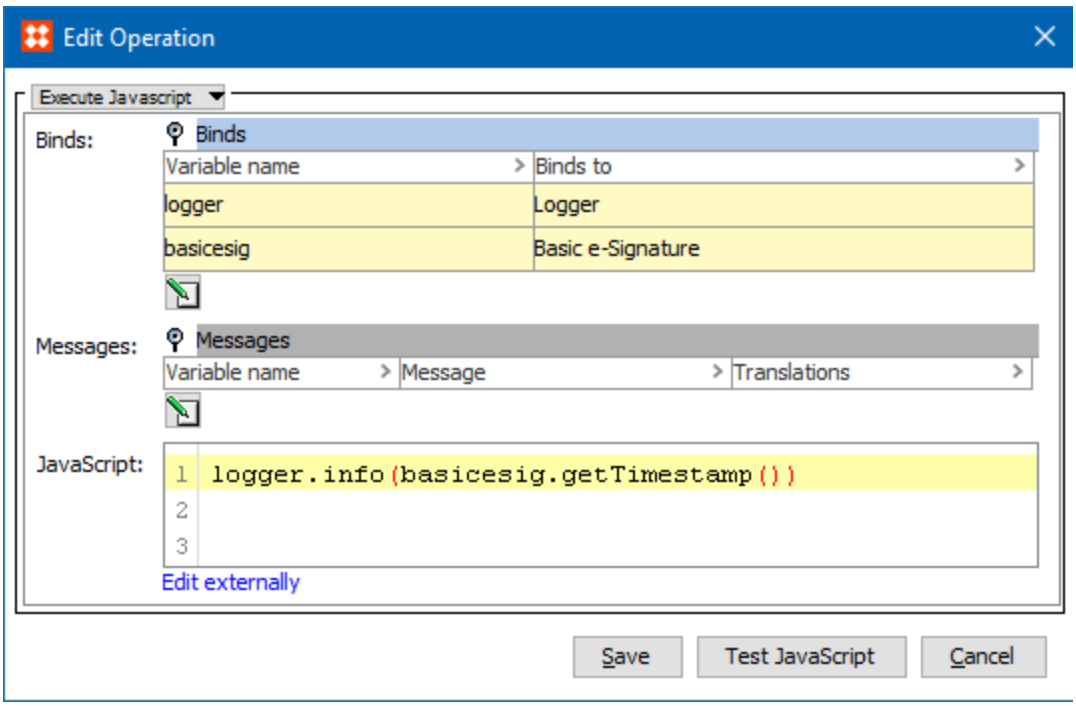
1. Create a business rule as defined in the **Creating a Business Rule or Library** topic.
2. Edit the business rule as defined in the **Editing a Business Rule** topic.
3. In the Edit Operation dialog, add the bind to a business rule, as defined in the **Adding a Bind** topic.
4. In the Edit Operation dialog, optionally add Messages, as defined in the **Adding a Localized Business Rule Message** topic.
5. In the Edit Operation dialog, add JavaScript to call the bind.

Example

The following is example JavaScript that uses this bind.

Important: The example scripts should not be used as-is without thorough testing, including updating the script to match object and link types that exist on your system.

Configuring e-Signature as shown below sets it up in workbench and makes it active for use in Web UI.

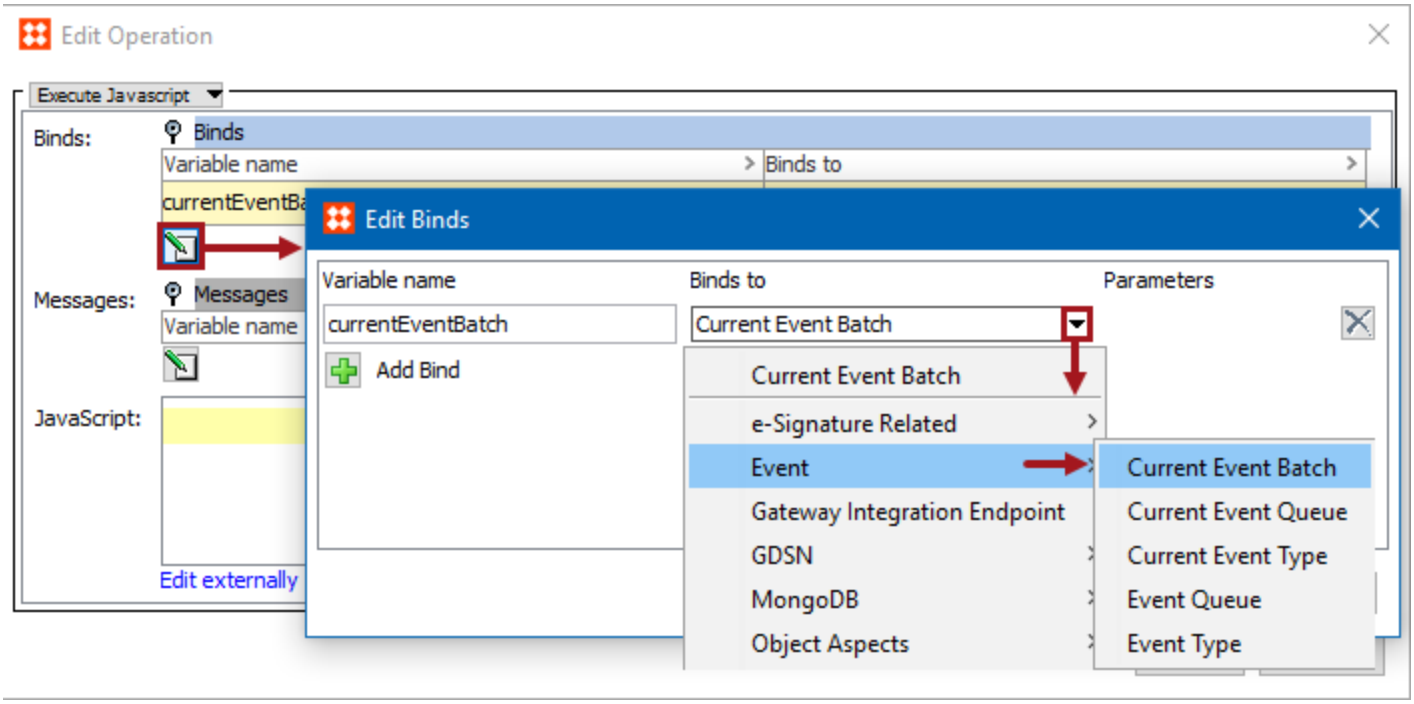


For an example of how to use e-Signature, see the **Using e-Signature in Web UI** topic in the **Business Rules** documentation.

Event Binds

Business rules can use any of the event binds listed below to access the selected event object. Binds that use the current object do not require a parameter selection. For the Event Queue and Event Type binds, a specific queue and type is selected in the Parameters field.

For additional examples of checking the current event and queuing a derived event via event binds and JavaScript, see the **Event Bind Examples** topic.



Important: The example scripts should not be used as-is without thorough testing, including updating the script to match object and link types that exist on your system.

Each bind is defined in the sections below.

Configuration

To use any bind:

1. Create a business rule as defined in the **Creating a Business Rule or Library** topic.
2. Edit the business rule as defined in the **Editing a Business Rule** topic.
3. In the Edit Operation dialog, add the bind to a business rule, as defined in the **Adding a Bind** topic.
4. In the Edit Operation dialog, optionally add Messages, as defined in the **Adding a Localized Business Rule Message** topic.
5. In the Edit Operation dialog, add JavaScript to call the bind.

Current Event Batch

This bind is only available for actions used as an OIEP Preprocessor and gives access to the batch of events currently being handled. This makes it possible to examine the events in the batch and to add new objects to the batch.

For example, this JavaScript adds new product / node '(newNode)' to the current event batch.

```
eventBatch.addAdditionalNode (newNode);
```

In this example, Current Event Batch is bound to the variable 'currentEventBatch'. For each event in the batch, the code accessed the object (node) for which the event was generated, checks if it is a product, and if so, adds the parent object to the batch.

```
eventsList = currentEventBatch.getEvents();
for (var i = 0; i < eventsList.size(); i++) {
    var node = eventsList.get(i).getNode();
    if (node instanceof com.stibo.core.domain.Product) {
        currentEventBatch.addAdditionalNode(node.getParent());
    }
}
```

Current Event Queue

This bind is available for conditions and actions used as OIEP event filters / event generators. It gives access to the event queue tied to the OIEP and from an action makes it possible to, for example, add derived events to the queue.

For example, this JavaScript queues new product / node '(newNode)' using ‘eventType’, which is bound through Event Type bind (discussed later in this topic).

```
eventQueue.queuedDerivedEvent (eventType, newNode);
```

In this example, Current Event Queue is bound to 'currentEventQueue', Current Object is bound to the variable 'currentObject', and a derived event is bound to the variable 'myEvent' via the Event Type bind. The code queues an event for the parent of current object if current object is a product.

```
if (currentObject instanceof com.stibo.core.domain.Product) {
    currentEventQueue.queueDerivedEvent(myEvent, currentObject.getParent());
}
```

Current Event Type

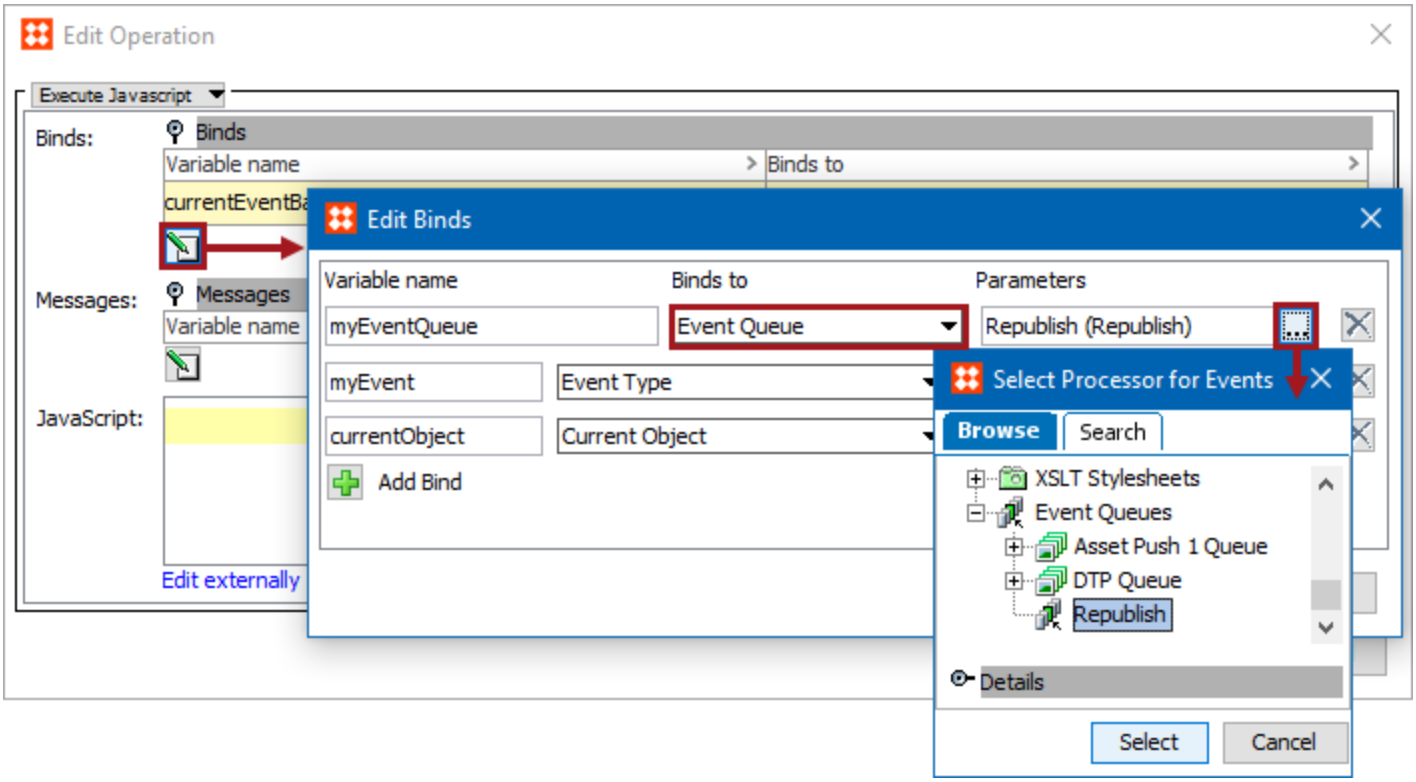
This binds the current event type and can be hooked into an endpoint as filters / generators to examine the current event. It is available for conditions and actions used as OIEP event filters / event generators. Use this to examine the type of the current event.

In this example, Current Event Type is bound to 'currentEventType' and the code checks whether the current event is a core 'modify' event.

```
if (currentEventType.equals
(com.stibo.core.domain.eventqueue.BasicEventType.Modify)) {
    // Do something if core modify event
}
```

Event Queue

This bind is available for conditions and actions, although typically used in actions. Use the bind from an action to queue an event on the selected event queue in STEP.



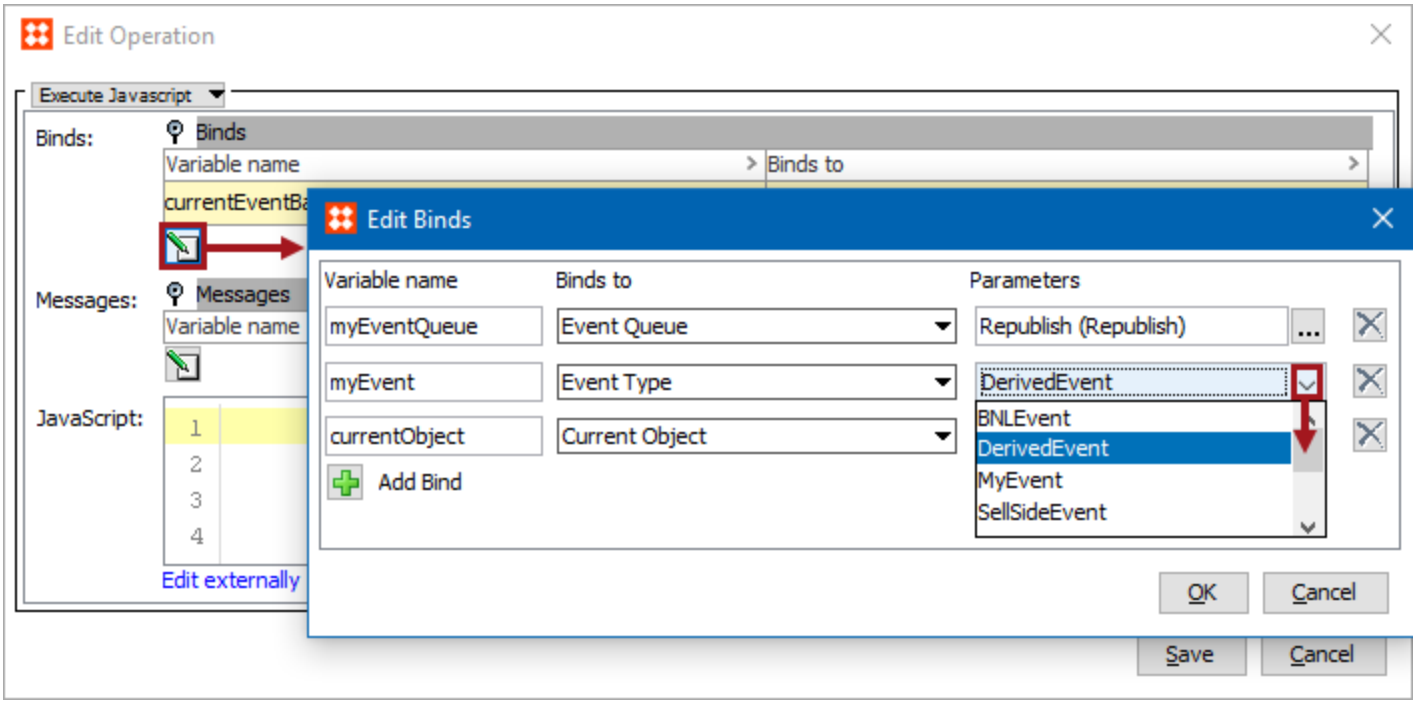
In this example, a derived event is bound to the variable 'myEvent' via the Event Type bind, an event queue is bound to the variable 'myEventQueue' via the Event Queue bind, and Current Object is bound to 'currentObject'. The example places a 'myEvent' event on the queue for current object.

```
myEventQueue.queueDerivedEvent(myEvent, currentObject);
```

Event Type

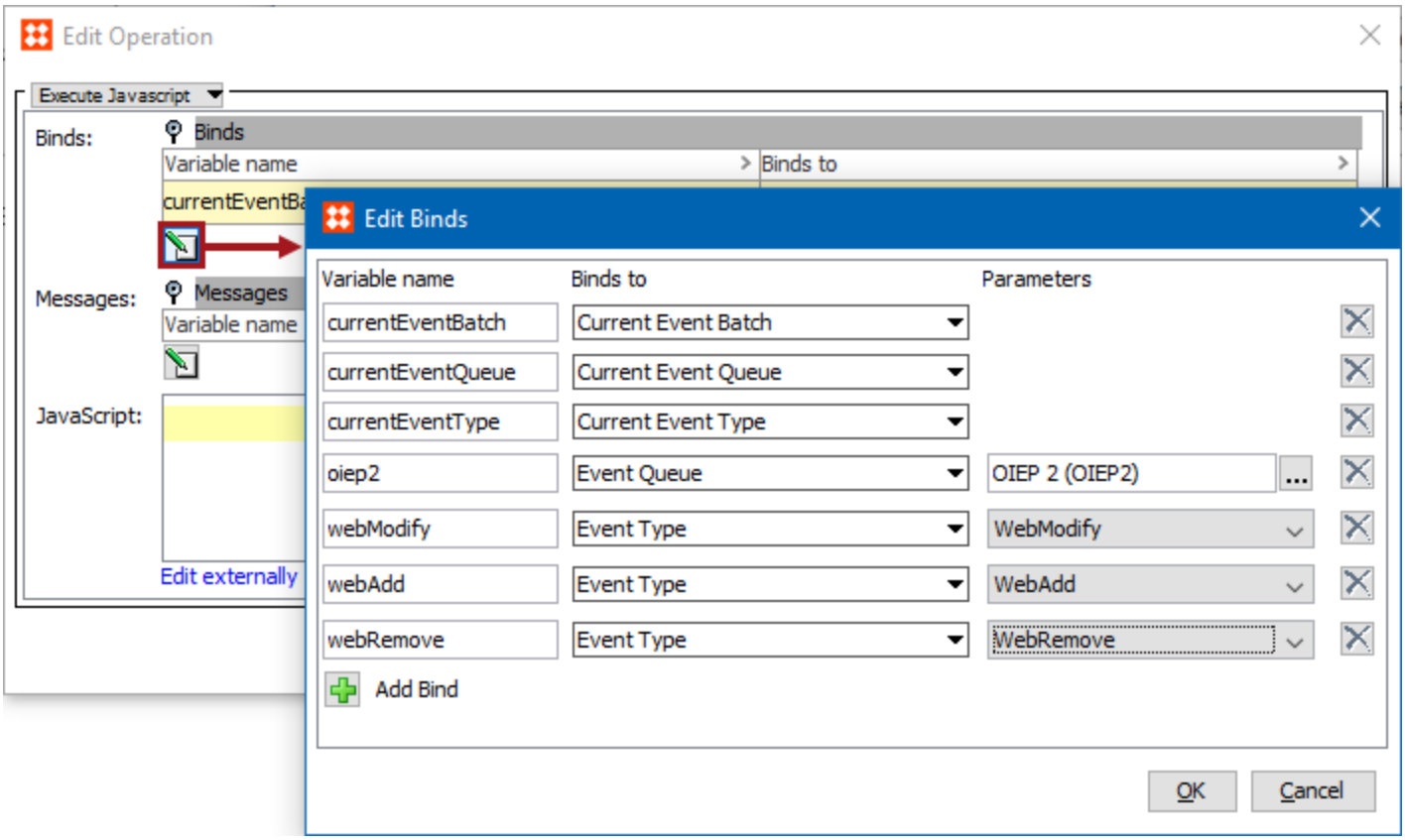
This bind is available for actions and conditions, although typically used in actions, binds a specified derived event to a JavaScript variable.

The same binds and variables are used in this example, as in the one above for Event Queue.


Edit Operation
Execute Javascript
Binds:
Binds
Variable name
Binds to
Messages:
Messages
Variable name
JavaScript:
1
2
3
4
Edit externally
Edit Binds
Variable name
Binds to
Parameters
myEventQueue
Event Queue
Republish (Republish)
myEvent
Event Type
DerivedEvent
currentObject
Current Object
Add Bind
BNLEvent
DerivedEvent
MyEvent
SellSideEvent
OK
Cancel
Save
Cancel

Event Bind Examples

The example code below expects these event binds:



Important: The example scripts should not be used as-is without thorough testing, including updating the script to match object and link types that exist on your system.

Checking Current Event

From actions and conditions used as filters / generators in an outbound integration endpoint, you can obtain the current event using a binding as shown above. The event will be either a core STEP event (BasicEventType Enum) or a derived event (DerivedEventType interface).

This code checks whether current event is a Core event:

```
if (currentEvent instanceof com.stibo.core.domain.eventqueue.BasicEventType) {
// It is a core event
}
```

This code checks whether current event is a core Modify event:

```
if (currentEvent.equals
(com.stibo.core.domain.eventqueue.BasicEventType.Modify)) {
```

```
// It is a core Modify event
}
```

This code checks whether current event is a Derived event:

```
if (currentEvent instanceof com.stibo.core.domain.eventqueue.DerivedEventType)
{
// It is a derived event
}
```

This code checks whether current event is derived 'WebModify' event:

```
if (currentEvent.equals(webModify)) {
// It is a derived WebModify event
}
```

Queuing a Derived Event

From an action, you can queue derived events using the following method on the EventQueue interface:

```
queueDerivedEvent(DerivedEventType event, Node node)
```

For a generator action hooked into an outbound integration endpoint, you can queue a derived event on a current event queue as follows (using bindings described above):

```
currentEventQueue.queueDerivedEvent(webModify, node);
```

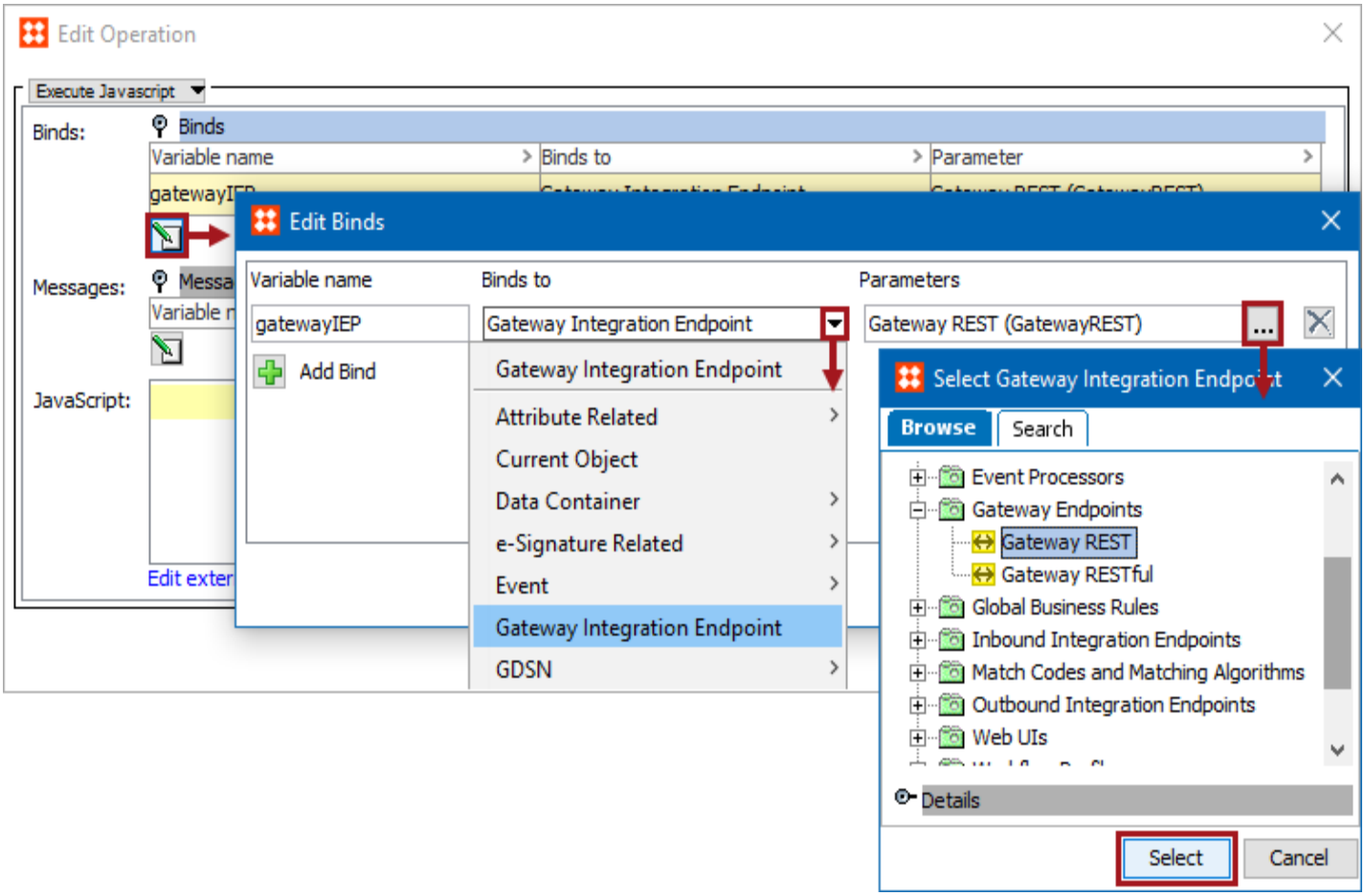
From any action, you can queue a derived event on an event queue via the 'Event Queue' and 'Event Type' bindings:

```
oiep2.queueDerivedEvent(webModify, node);
```

Gateway Integration Endpoint Bind

Gateway Integration Endpoints provide real-time inbound and outbound integration under STEP's control. The Gateway Integration Endpoint bind is accessed from JavaScript in business rule conditions and actions. The bind can work with the REST methods included in the **Gateway REST Methods** section below.

Note: A gateway IEP functions as an HTTP client and supports all of the standard HTTP methods. Therefore a sound basic understanding of the HTTP protocol is essential to be able to configure and use a gateway IEP.



Configuration

To use any bind:

1. Create a business rule as defined in the **Creating a Business Rule or Library** topic.
2. Edit the business rule as defined in the **Editing a Business Rule** topic.
3. In the Edit Operation dialog, add the bind to a business rule, as defined in the **Adding a Bind** topic.
4. In the Edit Operation dialog, optionally add Messages, as defined in the **Adding a Localized Business Rule Message** topic.
5. In the Edit Operation dialog, add JavaScript to call the bind.

Gateway REST Methods

This bind can work with the following REST methods. For more information about these methods, click the **STEP API Documentation** button on the STEP WebStart page.

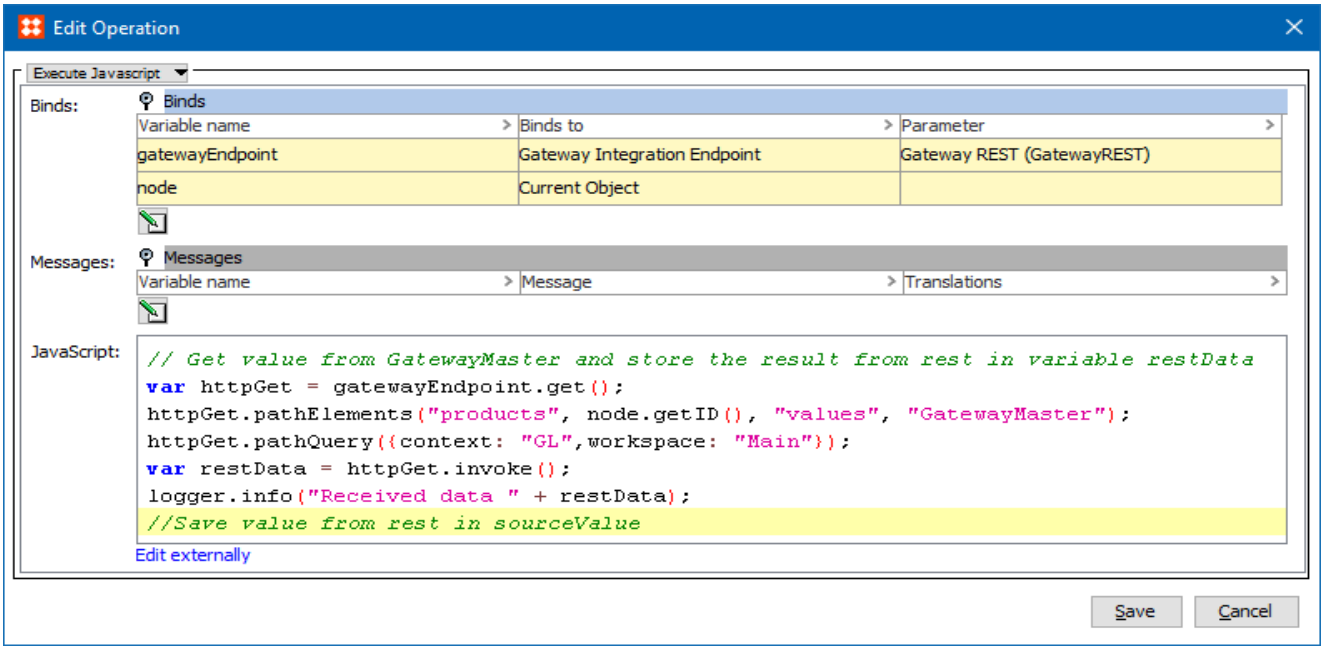
REST Method	Syntax Example
delete	`delete().path("products").pathQuery({ workspace: "Approved", context: "GL"}).body("Test").invoke();`
get	`get().path("products").pathQuery({ workspace: "Approved", context: "GL"}).invoke();`
head	`head().path("products").pathQuery({ workspace: "Approved", context: "GL"}).invoke();`
options	`options().path("products").pathQuery({ workspace: "Approved", context: "GL"}).invoke();`
post	`post().path("products").pathQuery({ workspace: "Approved", context: "GL"}).invoke();`
put	`put().path("products").pathQuery({ workspace: "Approved", context: "GL"}).body("Test").invoke();`

Example

The following is example JavaScript that uses this bind.

Important: The example scripts should not be used as-is without thorough testing, including updating the script to match object and link types that exist on your system.

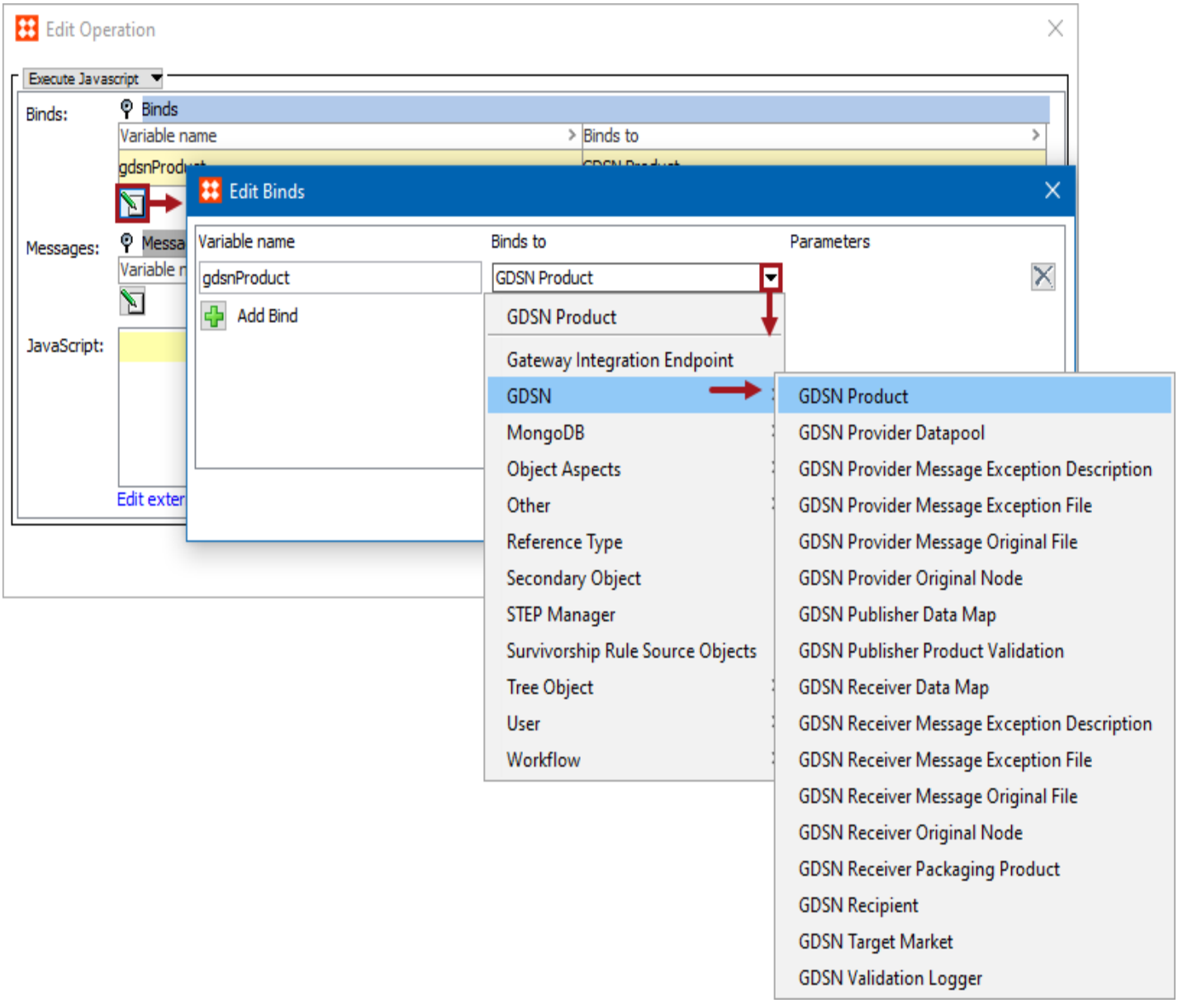
The following example uses a REST gateway integration endpoint to access the external server for information.



When the business action is executed, the gateway endpoint is accessed. The gateway endpoint Statistics tab displays an overview of executed REST methods. For more information, see the **Gateway Integration Endpoint Statistics** section.

GDSN Binds

Business actions used in the GDSN Provider and the GDSN Receiver solutions have additional parameters that can be configured to extract information from the GDSN XML files using XPath expressions.



Bind Name	Description
GDSN Product	Binds the selected GDSN product for use in the Pre action business action parameter or in the validation conditions.

Bind Name	Description
GDSN Provider Datapool	Binds the selected GDSN provider datapool for use in the Pre action business action parameter or in the validation conditions.
GDSN Provider Message Exception Description	In JavaScript business actions, this binds the GDSN Message Exception description which is stored in the background process.
GDSN Provider Message Exception File	For JavaScript business actions, this bind will provide the GDSN Message Exception file.
GDSN Provider Message Original File	This bind will provide the original outbound message file.
GDSN Provider Original Node	This bind will provide the node (the product) that was send to GDSN. Used when configuring GDSN failure message handling.
GDSN Publisher Data Map	For a business action used by the GDSN Provider, this bind can get the value of a specific XPath. The XPath value can subsequently be used to retrieve the corresponding string in the GDSN XML file.
GDSN Publisher Product Validation	Deprecated. Provides access to information about a product instance that is registered for a given data pool. For example, this could be used to write a script that can validate if all required attributes for a given data receiver have been populated on a given product. For more information about the Validation Condition option, see **GDSN in Web UI Overview**.
GDSN Receiver Data Map	For a business action used by the GDSN Receiver, this bind can get the value of a specific XPath. The XPath value can subsequently be used to retrieve the corresponding string in the GDSN XML file.
GDSN Receiver Message Exception Description	In JavaScript business actions, this binds the GDSN Message Exception description which is stored in the background process.
GDSN Receiver	For JavaScript business actions, this bind will provide the GDSN Message Exception file.

Bind Name	Description
Message Exception File	
GDSN Receiver Message Original File	This bind will provide the original outbound message file.
GDSN Receiver Original Node	This bind will provide the node (the product) that was send to GDSN. Used when configuring GDSN failure message handling.
GDSN Receiver Packaging Product	The bind will return the current product as a GDSN Receiver packaging member node. This makes it possible to set various CIC related information on the products in a java script.
GDSN Recipient	Binds the selected GDSN recipient for use in the Pre action business action parameter or in the validation conditions.
GDSN Target Market	Binds the selected GDSN target market for use in the Pre action business action parameter or in the validation conditions.
GDSN Validation Logger	Access the GDSN validation logger for use in the Pre action business action parameter or in the validation conditions.

XPath Expressions

Each XPath expression is bound to a parameter name. Use the Inbound format settings for business actions in the Workbench to configure XPath expressions. For more information, see **Configuring the Inbound Message Format** section of the **GDSN Provider** documentation or **GDSN Receiver** documentation.

Bindings use the following method to evaluate the XPath and extract the value of a parameter:

```
String get(String key)
```

For example, assume you have a business action with two parameter binds:

Parameter 1:

- name: ResponseCode
- xPath:

```
//catalogueResponse/documentException/publication/description/@code
```

Parameter 2:

- name: DocumentException
- xPath:

```
//catalogueResponse/documentException/publication/description/text()
```

In this example business action, the Data Map bind has the variable name 'dataMap'.

The following JavaScript code includes the new variable 'attributeValue', which is created by concatenating the value of the XPath for the ReponseCode parameter with the value of the XPath for the DocumentException parameter:

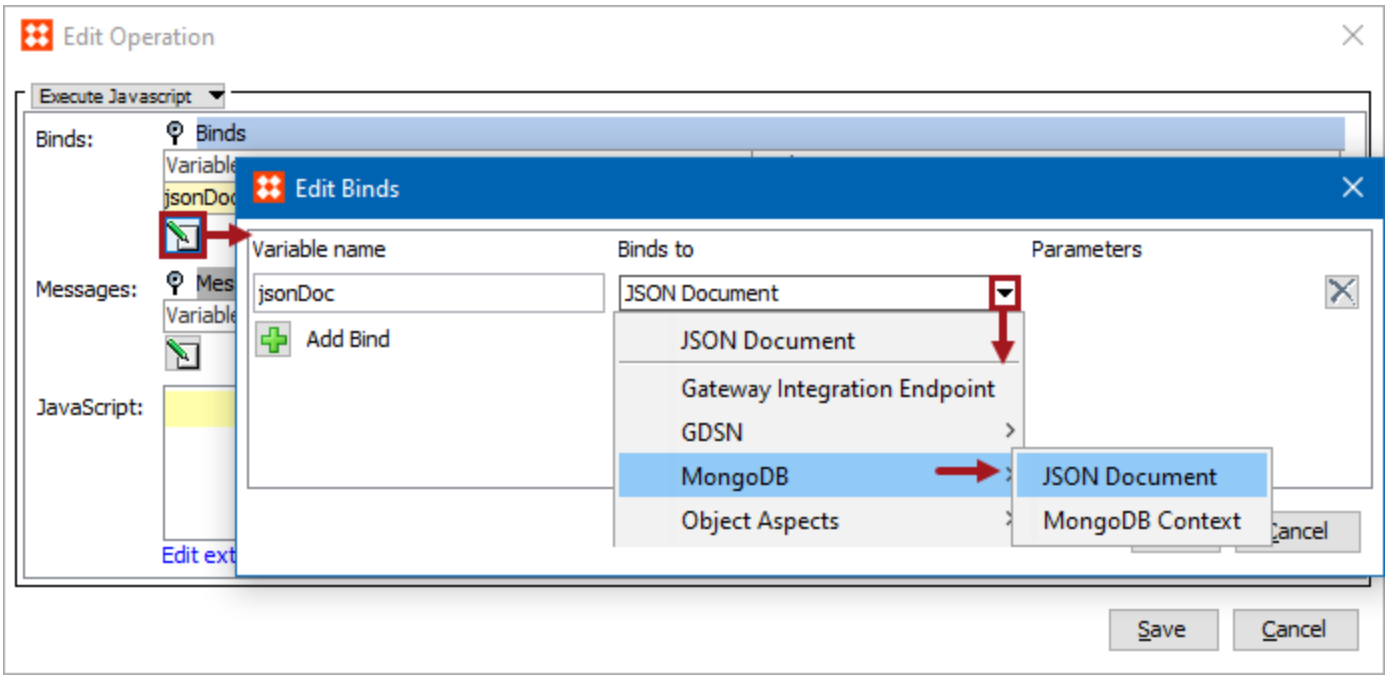
```
var attributeValue = dataMap.get("ResponseCode") + ": " + dataMap.get
("DocumentException");
```

The GDSN Publisher Data Map bind has a getBlockValues method that can fetch a list of strings from a repeated XML block. The parameter is configured on the business action in the Inbound format configuration. The configuration is an XPath that specifies a repeated XML block and an XPath that extracts the needed content from the repeated XML block:

```
List<String> getBlockValues(String blockKey)
```

MongoDB Binds

Business rules can use the following binds for the integration with MongoDB.



Each bind is defined in the sections below.

Configuration

To use any bind:

1. Create a business rule as defined in the **Creating a Business Rule or Library** topic.
2. Edit the business rule as defined in the **Editing a Business Rule** topic.
3. In the Edit Operation dialog, add the bind to a business rule, as defined in the **Adding a Bind** topic.
4. In the Edit Operation dialog, optionally add Messages, as defined in the **Adding a Localized Business Rule Message** topic.
5. In the Edit Operation dialog, add JavaScript to call the bind.

JSON Document

When used in a business action referenced from the 'Mongo delivery' plugin, this bind gives access to a JSON object for the data that has just been persisted.

MongoDB Context

This bind gives access to an object with the following methods:

```
getMongo()
```

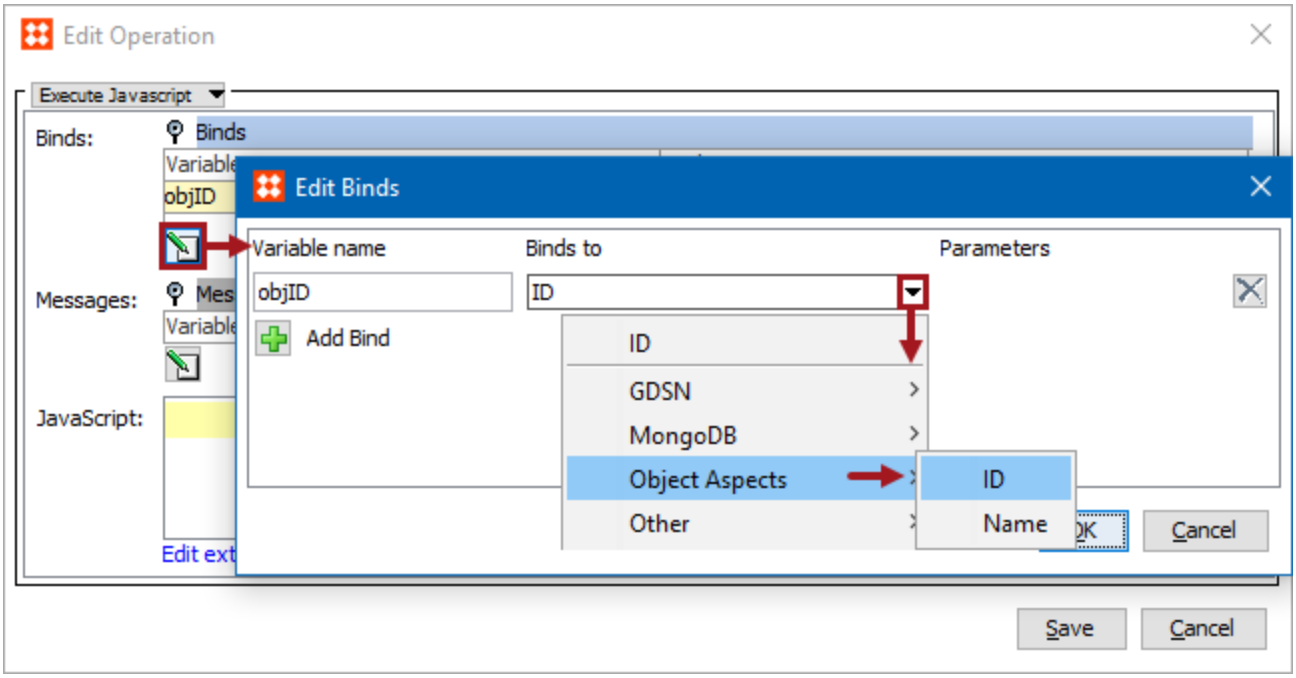
The MongoDB database is accessed via the com.mongodb.Mongo object. For more information, see http://api.mongodb.org/java/current/com/mongodb/Mongo.html.

```
getContext()
getContextID()
getRawDBName()
```

Object Aspects Binds

Business rules can use the following object aspects binds to get the ID and Name of STEP object that the business rule is being evaluated or executed against.

For information on accessing the current object, see the **Current Object Bind** documentation.



Each bind is defined in the sections below.

Configuration

To use any bind:

1. Create a business rule as defined in the **Creating a Business Rule or Library** topic.
2. Edit the business rule as defined in the **Editing a Business Rule** topic.
3. In the Edit Operation dialog, add the bind to a business rule, as defined in the **Adding a Bind** topic.
4. In the Edit Operation dialog, optionally add Messages, as defined in the **Adding a Localized Business Rule Message** topic.
5. In the Edit Operation dialog, add JavaScript to call the bind.

Important: The example scripts should not be used as-is without thorough testing, including updating the script to match object and link types that exist on your system.

ID

This bind resolves to the ID of current object (Java String).

The following code checks if the ID is not ‘Product hierarchy root’ then sets the ID to attribute NODEID.

```
//node = bind to current node.

if (!id.equals(“Product hierarchy root”)) {
    node.getValue("NODEID").setSimpleValue(id);
}
```

Name

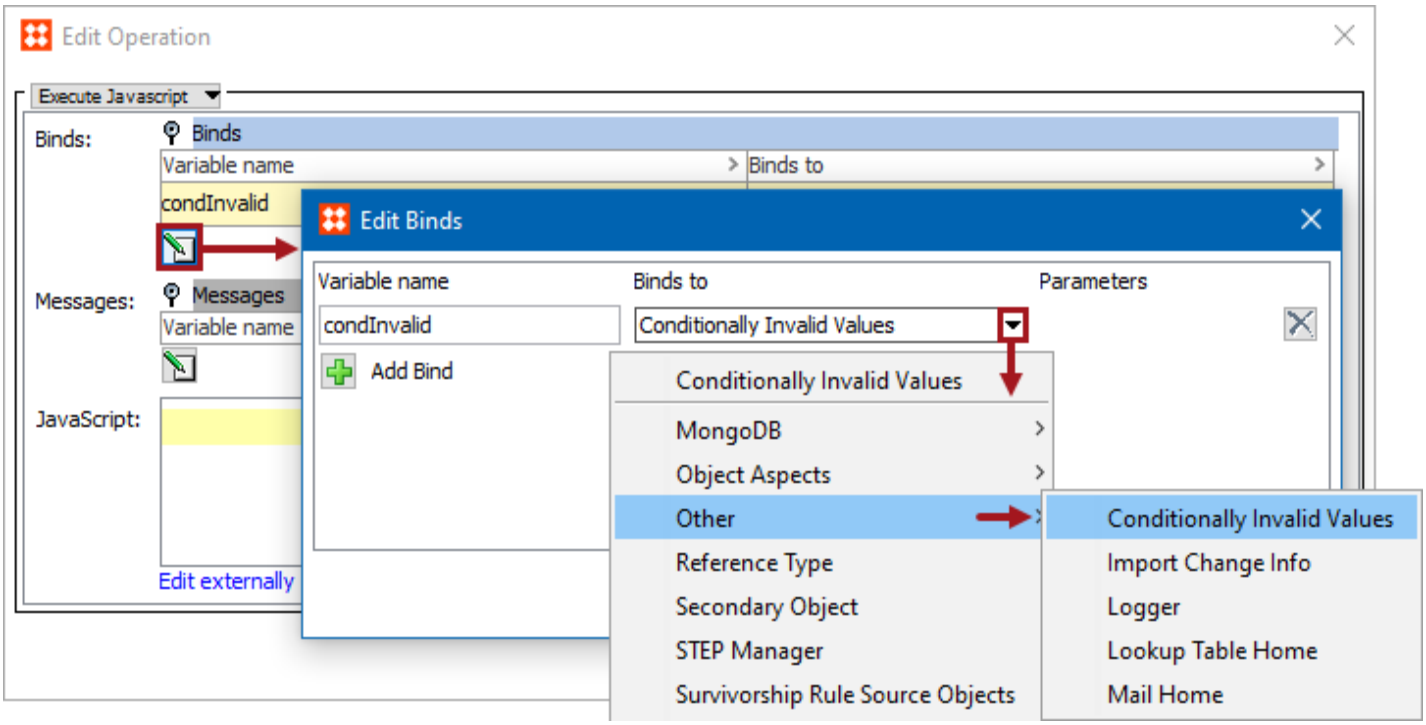
This bind resolves to the name of current object (Java String or null if no name).

In this example, 'Name' is bound to the variable 'name'. The code checks if there is a name.

```
//node = bind to current node.

if (name) {
    node.getValue("NODENAME").setSimpleValue(name);
}
```

Other Binds

Business rules can use any of the following 'Other' binds.



Each bind is defined in the sections below.

Configuration

To use any bind:

1. Create a business rule as defined in the **Creating a Business Rule or Library** topic.
2. Edit the business rule as defined in the **Editing a Business Rule** topic.
3. In the Edit Operation dialog, add the bind to a business rule, as defined in the **Adding a Bind** topic.
4. In the Edit Operation dialog, optionally add Messages, as defined in the **Adding a Localized Business Rule Message** topic.
5. In the Edit Operation dialog, add JavaScript to call the bind.

Important: The example scripts should not be used as-is without thorough testing, including updating the script to match object and link types that exist on your system.

Conditionally Invalid Values

This bind is used with Conditional Attributes. For more information, see the **Business Rules with Conditional Attributes** documentation.

For information about creating conditional attributes, see the **Conditional Attribute Display** section of the **System Setup / Super User Guide**.

Import Change Info

Use this bind when the business rule is running in an import. It accesses what has changed with the import (for example, new / existing objects, updated attributes). This object will be populated for each object in the import.

The object contains two members:

```
isUnmodified()
```

- true when an object is unchanged by the import
- false when the object is new or modified

```
getChanges()
```

- returns a 'changes' object with a getAttributes() method that lets you see which Attribute values are changed by the import

Logger

This is shorthand for the Java Logger class. Logger writes to the main STEP log file on the application server (step.*.log) when the condition or action runs. This variable is always bound in, gives access to a java.util.logging.Logger object, and allows logging of messages at the following severity levels:

```
logger.fine("Message to log");
logger.info("Message to log");
logger.warning("Message to log");
logger.severe("Message to log");
```

By default, the STEP log does not register messages at the 'fine' severity level.

After development and debugging a new Business Rule, including the Business Rule ID and the context in the log message aids in troubleshooting when required.

Note: When testing a business rule, Logger writes directly to the test dialog, not to the STEP log file.

Lookup Table Home

The interface has a single public method that returns a String containing either the lookup value or the original value if no match could be made:

```
getLookupTableValue(String assetID, String value)
```

- The first argument is the ID of the lookup table (an asset)
- The second argument is the string for which you want to get the lookup value.

For more information, see the **Using JavaScript with Lookup Tables** documentation and the **Transformation Lookup Tables** topic in the **System Setup / Super User Guide**.

Mail Home

This bind allows the JavaScript to send emails. The bound class is the API Mail Home object.

Binds to the MailHome domain interface available for actions and conditions (although typically used with actions). After mail server settings are configured on STEP, this bind allows sending of emails that support full HTML.

In this example, 'Mail Home' is bound to the variable 'mailHome'.

```
mailHome.mail()
      .from("Address of Sender")
      .addTo("Address of Recipient", "Common Name")
      .htmlMessage("Full HTML message:)
      .subject("Subject of Email")
      .send();
```

Business Rules with Conditional Attributes

For information on setting up a conditionally valid attribute, see the **Conditional Attribute Display** documentation in the **System Setup / Super User Guide**.

For information on adding a bind, see the **Adding a Bind** topic.

JavaScript business rule bind for Conditional Validity

It is possible to make use of the Conditionally Invalid Values bind in a JavaScript business action. This will resolve to a (possibly empty) set of values, and this set will contain all values which has value conditions that for the current node evaluates to false.

Note: Note that this binding is unrestricted in the sense that it is always available, not just from workflows or imports, etc.

This is a JavaScript example for clearing conditionally invalid values.

Required binds:

- invalid:"Conditionally Invalid Values"-binding
- logger:"Logger"-binding

```
//acquire an iterator - easiest way to deal with Sets
var iter = invalid.iterator();
//iterate over conditionally invalid values
while (iter.hasNext()){
//get the value
var val = iter.next();
//test if there is a value and if it is local
var clean = val.isLocal() && val.getSimpleValue();
//do some chatting to the log
var msg
if (clean)
msg = "Cleaning up";
else
msg = "Ignoring";
msg += " conditionally invalid ";
if (val.isLocal())
msg += "local value";
else
msg += "non-local value";
```

```
msg+= val.getAttribute().getID() + " [" + val.getSimpleValue() + "]";

logger.info( msg );

//in case of local value - delete it

if (clean) val.setSimpleValue("");//empty string handled as null and deletes
local values

}
```

JavaScript business rule bind of Workflow State

It is possible to make use of the workflow state bindings for conditions and actions that, on the import, can resolve and be used for general evaluation and handling of the product being imported.

This is an example business condition that be applied on import of a maintenance Smartsheet, exported from the Web UI Task List.

Required binds:

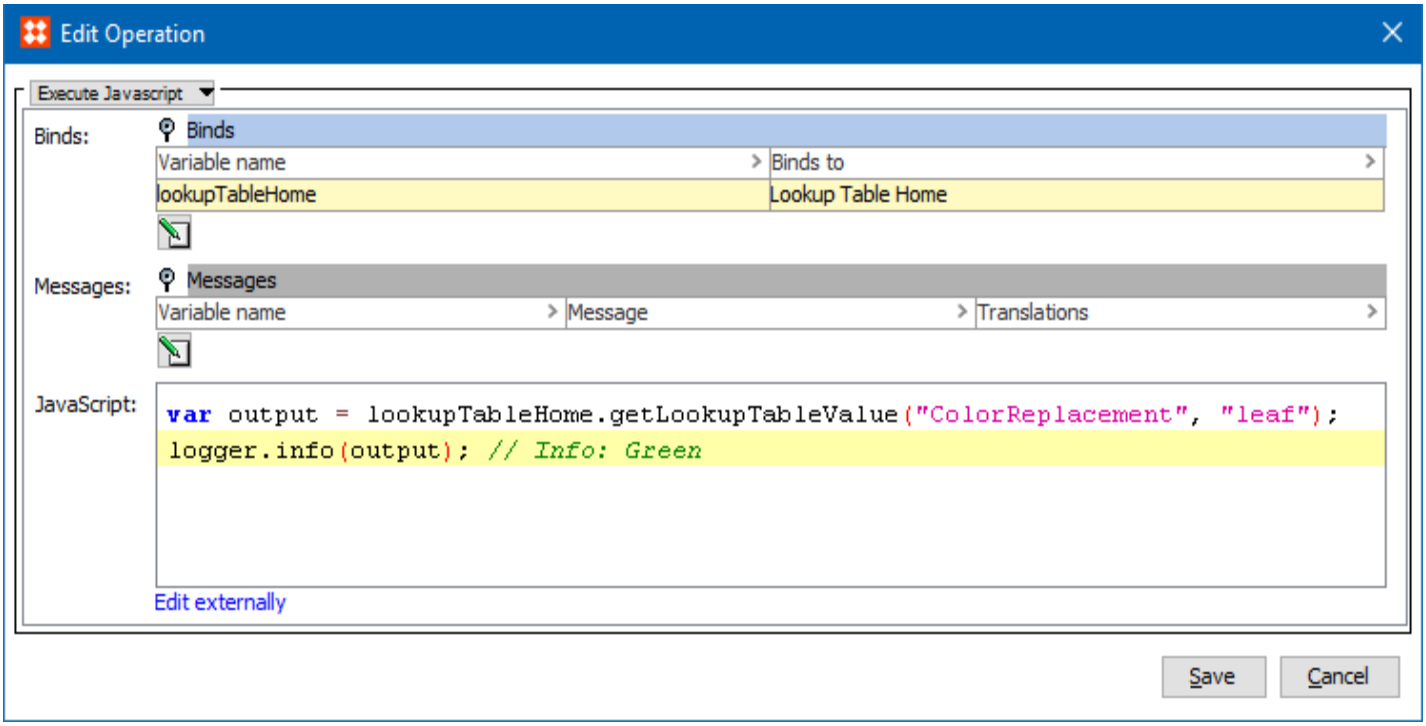
- state:"Workflow state"-binding
- node:"Current Object"-binding
- logger:"Logger"-binding

```
//up front test if the import carries expectation of specific workflow state

if (!state) {

logger.info("No import state provided");

//could have instead return some rejection to say no import if a state is not
supplied

return true;

}

var instance, task, msg;

//acquire the workflow instance for the node and workflow

instance = node.getWorkflowInstance(state.getWorkflow());

if (!instance) {

//simply reject since workflow has been terminated or never started

msg = "Rejected : Workflow " + state.getWorkflow().getTitle() + " not started
for " + node.getTitle();

logger.info(msg);

return msg

}

//now look for actual task for node in supplied state

task = instance.getTask(state);

if (!task){
```

```
//no task also causes rejection
msg = "Someone might have changed data " + node.getTitle() + " has no task for
state " + state.getID() + " in worflow " + state.getWorkflow().getTitle();
logger.info(msg);
return msg;
}
logger.info('Accepted: we have a task and will now continue with import');
return true;
```

Using JavaScript with Lookup Tables

Transformation Lookup Tables are used to make replacements via the Other > Lookup Table Home bind. For details about this bind, see the **Other Binds** documentation.

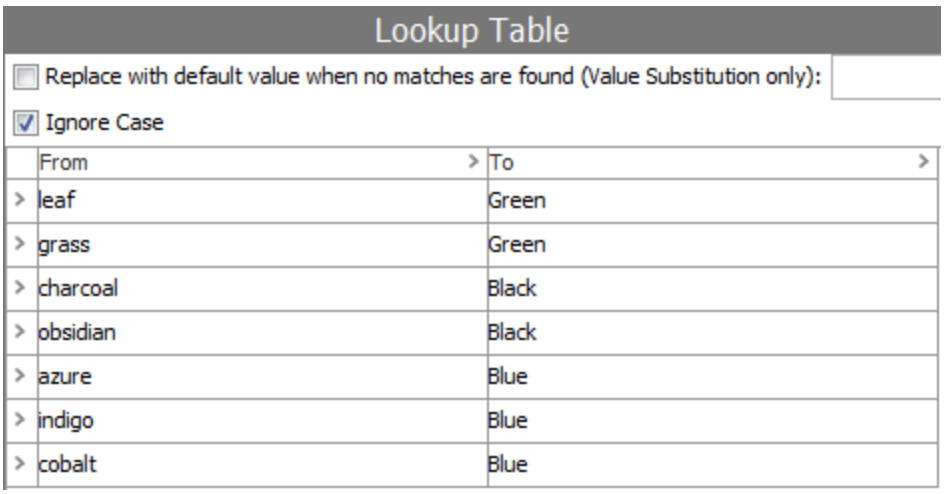


Important: The example scripts should not be used as-is without thorough testing, including updating the script to match object and link types that exist on your system.

Lookup Table

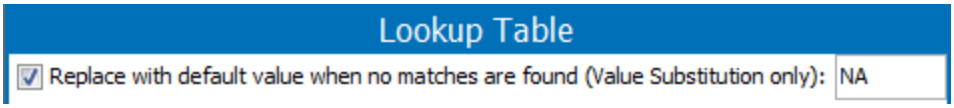
Lookup tables can be used by transformations to make text replacements. For more information, see the **Transformation Lookup Tables** topic in the **System Setup / Super User Guide**.

This example lookup table has an ID of 'ColorReplacement':



From	To
leaf	Green
grass	Green
charcoal	Black
obsidian	Black
azure	Blue
indigo	Blue
cobalt	Blue

Setting the 'Replace with default...' option on the Lookup Table returns the substitution value when a match is not found:



JavaScript using a Lookup Table

The following JavaScript accesses the ColorReplacement lookup table and returns 'Green':

```
var output = lookupTableHome.getLookupTableValue("ColorReplacement", "leaf");
logger.warning(output); // WARNING: Green
```

The 'getLookupTableValue' method returns the original string if a match is not found. In this example, the original string 'orange' is returned:

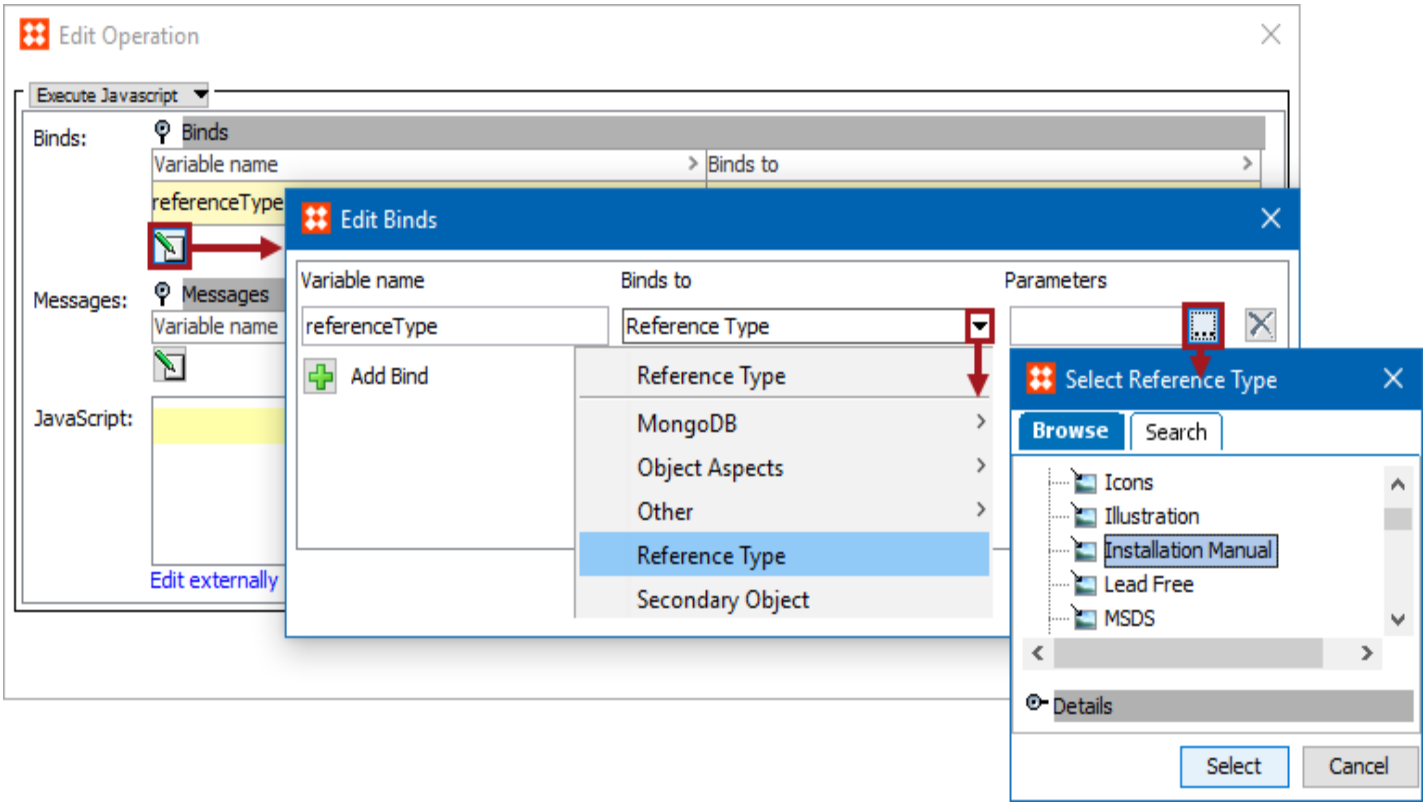
```
var output = lookupTableHome.getLookupTableValue("ColorReplacement", "orange");
logger.warning(output); // WARNING: orange
```

Using the 'Replace with default...' option on the Lookup Table set to NA as shown above, returns NA:

```
var output = lookupTableHome.getLookupTableValue("ColorReplacement", "rose");
logger.warning(output); // WARNING: NA
```

Reference Type Bind

Business rules can use the Reference Type bind to access the Reference Type or Link Type selected from the dialog. The bind can be found within the 'Binds to' dropdown, as shown below.



Configuration

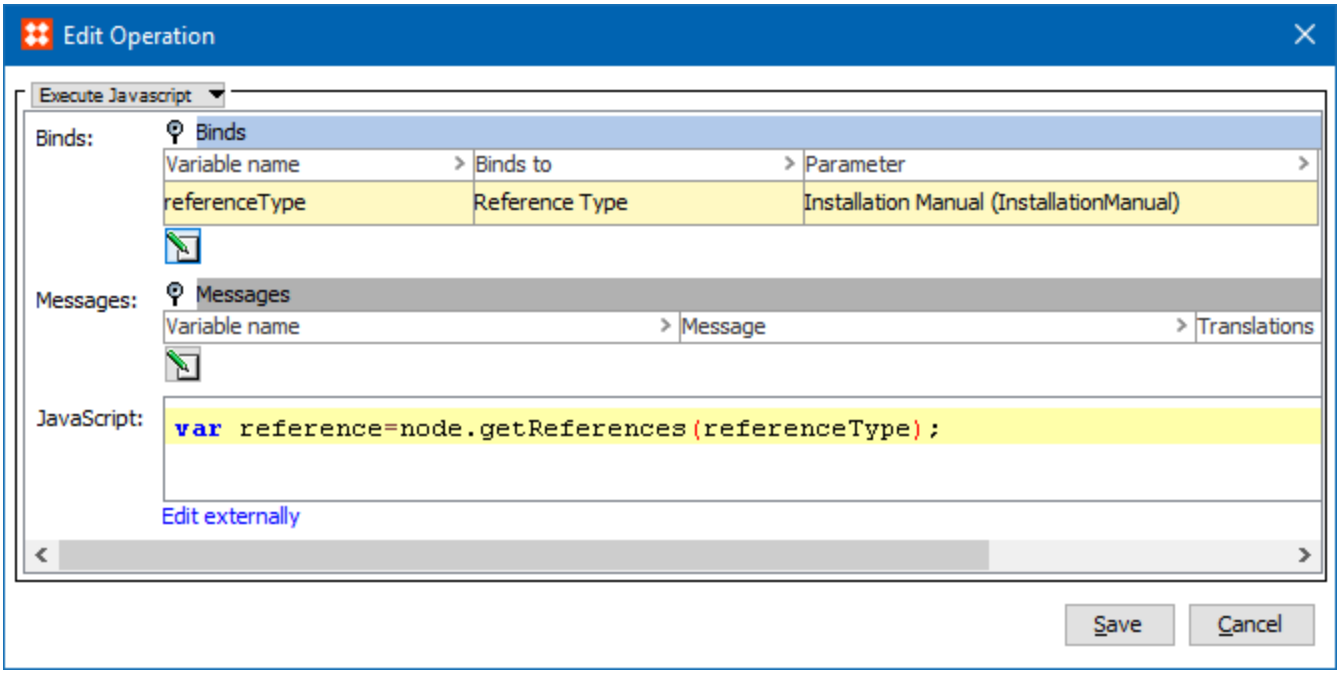
To use any bind:

1. Create a business rule as defined in the **Creating a Business Rule or Library** topic.
2. Edit the business rule as defined in the **Editing a Business Rule** topic.
3. In the Edit Operation dialog, add the bind to a business rule, as defined in the **Adding a Bind** topic.
4. In the Edit Operation dialog, optionally add Messages, as defined in the **Adding a Localized Business Rule Message** topic.
5. In the Edit Operation dialog, add JavaScript to call the bind.

Example

The following is example JavaScript that uses this bind.

Important: The example scripts should not be used as-is without thorough testing, including updating the script to match object and link types that exist on your system.



The following code gets the value of the reference type 'Installation Manual' on the current object and stores it in the variable reference.

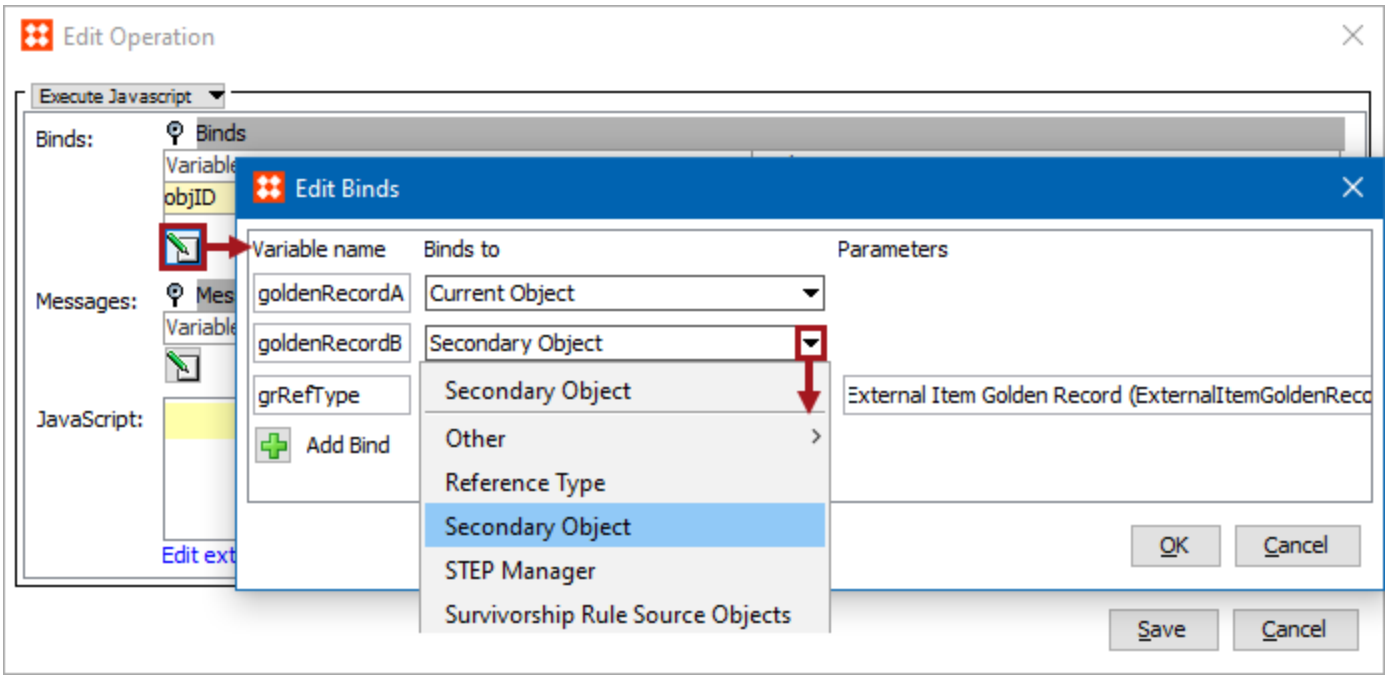
```
var reference=node.getReferences(referenceType);
```

Secondary Object Bind

Business rules can use the Secondary Object bind in conjunction with the Matching, Linking, and Merging functionality when there is a need for binding another object in addition to the Current Object. A number of the golden record match action handlers require access to two objects (for example, two golden records in merge and split cases).

For example, a business action selected in the Matching Algorithm's Match Action when merging two golden records could work on the two golden records to be merged using the Current Object and Secondary Object binds. For more information about golden record handlers, see **Updating Golden Records**.

The bind can be found within the 'Binds to' dropdown, as shown below.



Configuration

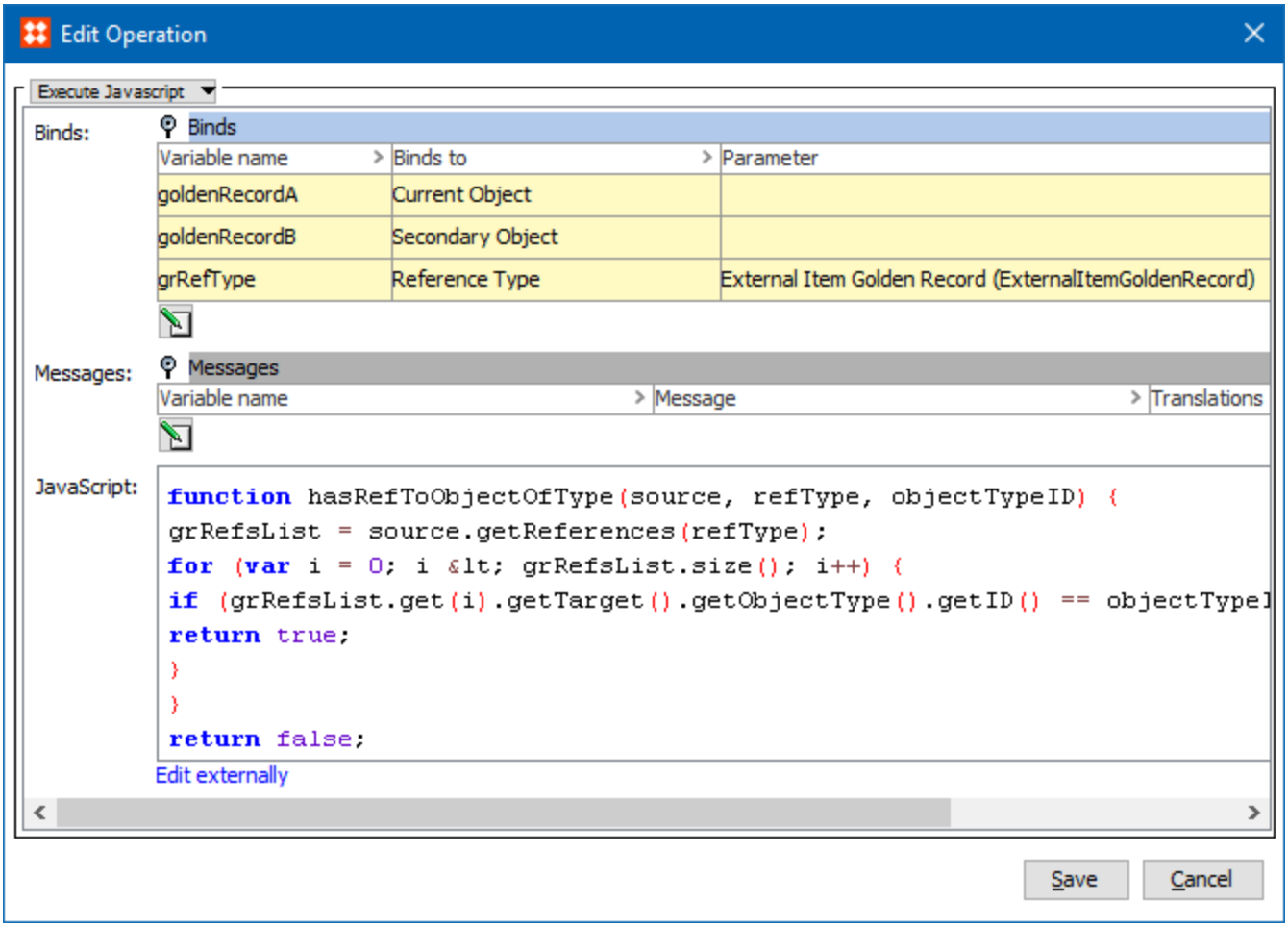
To use any bind:

1. Create a business rule as defined in the **Creating a Business Rule or Library** topic.
2. Edit the business rule as defined in the **Editing a Business Rule** topic.
3. In the Edit Operation dialog, add the bind to a business rule, as defined in the **Adding a Bind** topic.
4. In the Edit Operation dialog, optionally add Messages, as defined in the **Adding a Localized Business Rule Message** topic.
5. In the Edit Operation dialog, add JavaScript to call the bind.

Example

The following is example JavaScript that uses this bind.

Important: The example scripts should not be used as-is without thorough testing, including updating the script to match object and link types that exist on your system.



A simple merge action deletes the enrichment object for a golden record that is to be deleted as the consequence of a merge.

```
function hasRefToObjectOfType(source, refType, objectTypeID) {
grRefsList = source.getReferences(refType);
for (var i = 0; i < grRefsList.size(); i++) {
if (grRefsList.get(i).getTarget().getObjectType().getID() == objectTypeID) {
return true;
}
}
return false;
}
function deleteERForGR(gr, grRefType, erObjectTypeID) {
grRefsList = gr.getReferences(grRefType);
```

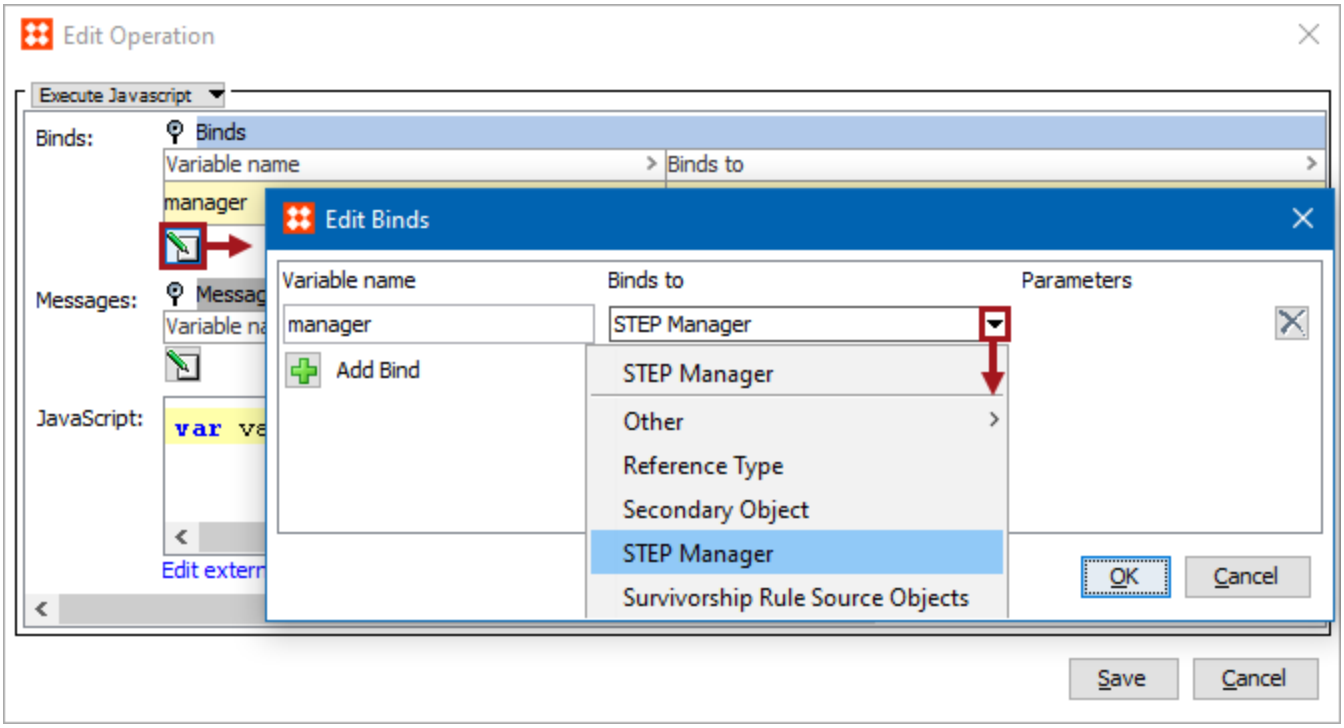
```
for (var i = 0; i < grRefsList.size(); i++) {
if (grRefsList.get(i).getTarget().getObjectType().getID() == erObjectTypeID) {
var er = grRefsList.get(i).getTarget();
try {
grRefsList.get(i).delete();
er.delete();
}
catch(e) {
throw e.getMessage();
}
return;
}
}
}
var soObjectTypeID = "ExternalItem";
var erObjectTypeID = "ExternalItemEnrichmentRecord";
var aStays = hasRefToObjectOfType(goldenRecordA, grRefType, soObjectTypeID);
var bStays = hasRefToObjectOfType(goldenRecordB, grRefType, soObjectTypeID);
if (aStays && bStays) {
throw "Something is wrong. Both GRs to be merged have references to source
objects.";
}
if (aStays) {
deleteERForGR(goldenRecordB, grRefType, erObjectTypeID);
}
if (bStays) {
deleteERForGR(goldenRecordB, grRefType, erObjectTypeID);
}
*/
erFunctions.deleteERForGR(goldenRecordB, grRefType, erObjectTypeID);
}
```

STEP Manager Bind

The STEP Manager binds current context and workspace, and is a general gateway into the API. This binding enables access to information about the current user, current context, and objects not available using the Current Object binding.

Important: For business conditions run On Approve, the STEP Manager binding applies to the current context and the main workspace. When run After Approval, the STEP Manager binding applies to current context and the approved workspace.

The bind can be found within the 'Binds to' dropdown, as shown below.



Configuration

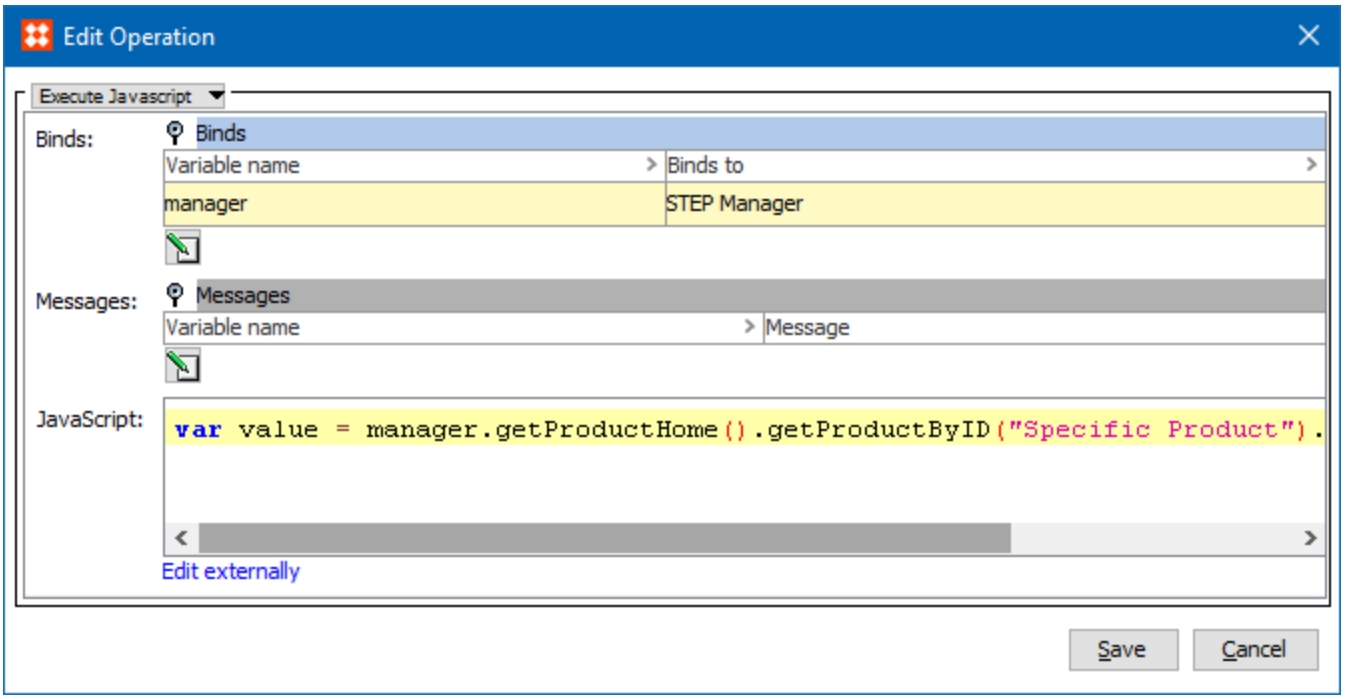
To use any bind:

1. Create a business rule as defined in the **Creating a Business Rule or Library** topic.
2. Edit the business rule as defined in the **Editing a Business Rule** topic.
3. In the Edit Operation dialog, add the bind to a business rule, as defined in the **Adding a Bind** topic.
4. In the Edit Operation dialog, optionally add Messages, as defined in the **Adding a Localized Business Rule Message** topic.
5. In the Edit Operation dialog, add JavaScript to call the bind.

Example

The following is example JavaScript that uses this bind.

Important: The example scripts should not be used as-is without thorough testing, including updating the script to match object and link types that exist on your system.



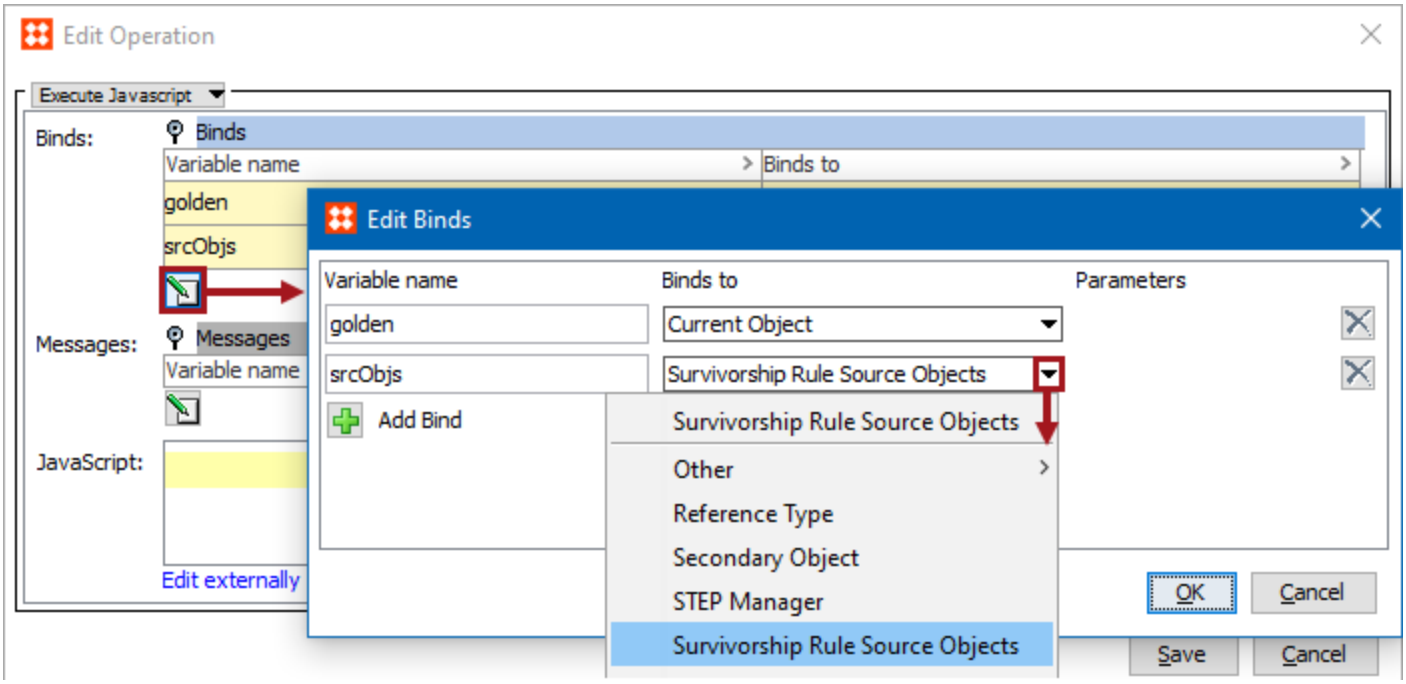
The STEP Manager is bound in as **manager**, and gets the value of the Attribute 'Description' on a specific product in current context / workspace.

```
var value = manager.getProductHome().getProductByID("Specific
Product").getValue("Description").getSimpleValue();
```

Survivorship Rule Source Objects Bind

The Survivorship Rule Source Objects bind can be used from business actions referenced from a 'business action survivorship rule' to access the source object for a golden record. For the 'merge approach', this is the only way to access source records from a survivorship rule business action.

The bind can be found within the 'Binds to' dropdown, as shown below.



Configuration

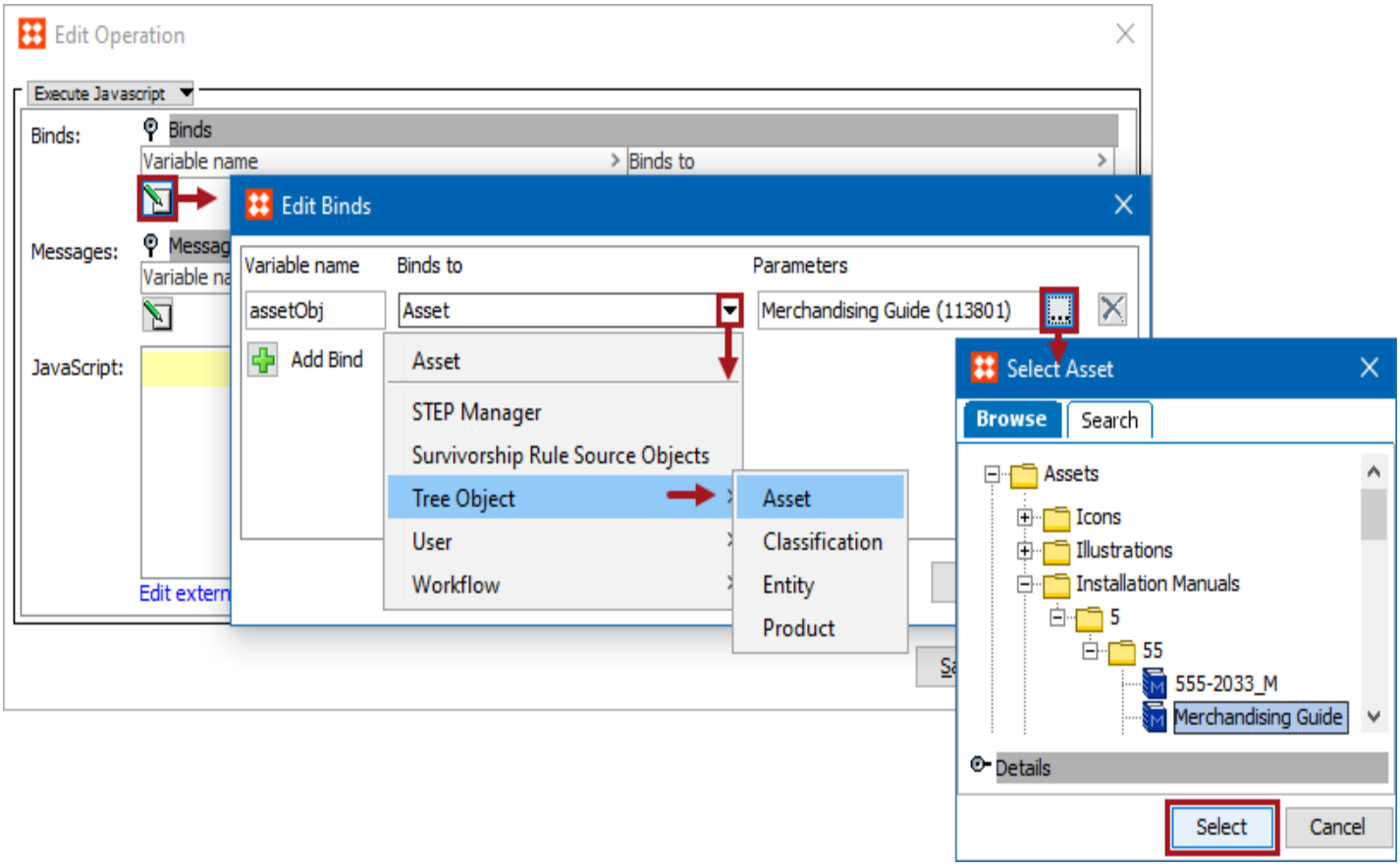
To use any bind:

1. Create a business rule as defined in the **Creating a Business Rule or Library** topic.
2. Edit the business rule as defined in the **Editing a Business Rule** topic.
3. In the Edit Operation dialog, add the bind to a business rule, as defined in the **Adding a Bind** topic.
4. In the Edit Operation dialog, optionally add Messages, as defined in the **Adding a Localized Business Rule Message** topic.
5. In the Edit Operation dialog, add JavaScript to call the bind.

For a JavaScript example of this bind, please see the online version of this topic.

Tree Object Binds

Business rules can use any of the following simple binds to gain access to an object on the Tree. These binds are especially useful when a particular object is needed repeatedly in the JavaScript.



Each bind is defined in the sections below.

Configuration

To use any bind:

1. Create a business rule as defined in the **Creating a Business Rule or Library** topic.
2. Edit the business rule as defined in the **Editing a Business Rule** topic.
3. In the Edit Operation dialog, add the bind to a business rule, as defined in the **Adding a Bind** topic.
4. In the Edit Operation dialog, optionally add Messages, as defined in the **Adding a Localized Business Rule Message** topic.
5. In the Edit Operation dialog, add JavaScript to call the bind.

Asset

This binds the selected asset to the action or condition variable. Click the ellipsis button **(…)** in the Parameters field to display the Select Asset dialog. Use Browse or Search to locate an asset, select it, and click the **Select** button.

Classification

This binds the selected classification to the action or condition variable. Click the ellipsis button **(…)** in the Parameters field to display the Select Classification dialog. Use Browse or Search to locate a classification , select it, and click the **Select** button.

Entity

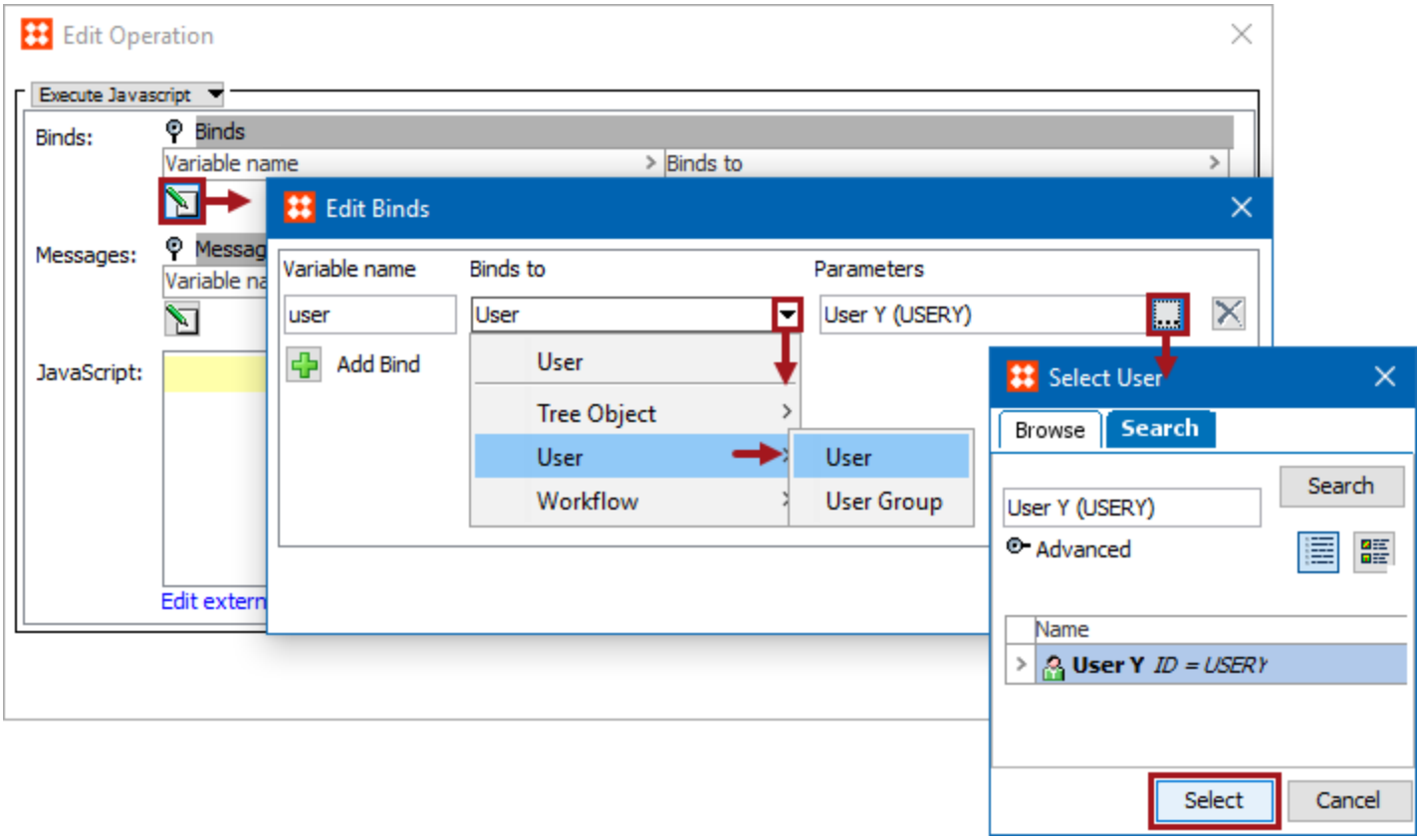
This binds the selected entity to the action or condition variable. Click the ellipsis button **(…)** in the Parameters field to display the Select Entity dialog. Use Browse or Search to locate an entity, select it, and click the **Select** button.

Product

This binds the selected product to the action or condition variable. Click the ellipsis button **(…)** in the Parameters field to display the Select Product dialog. Use Browse or Search to locate a product, select it, and click the **Select** button.

User Binds

Business rules can use the following binds to access a specified user or user group. For example, this is useful when a particular user or user group is required to approve a specific object type. The User bind can be used to provide the condition by requiring a specific user.



Each bind is defined in the sections below.

Configuration

To use any bind:

1. Create a business rule as defined in the **Creating a Business Rule or Library** topic.
2. Edit the business rule as defined in the **Editing a Business Rule** topic.
3. In the Edit Operation dialog, add the bind to a business rule, as defined in the **Adding a Bind** topic.
4. In the Edit Operation dialog, optionally add Messages, as defined in the **Adding a Localized Business Rule Message** topic.
5. In the Edit Operation dialog, add JavaScript to call the bind.

User

This binds the selected user to the action or condition variable. Click the ellipsis button **(…)** in the Parameters field to display the Select User dialog. Use Browse or Search to locate a user, select it, and click the **Select** button.

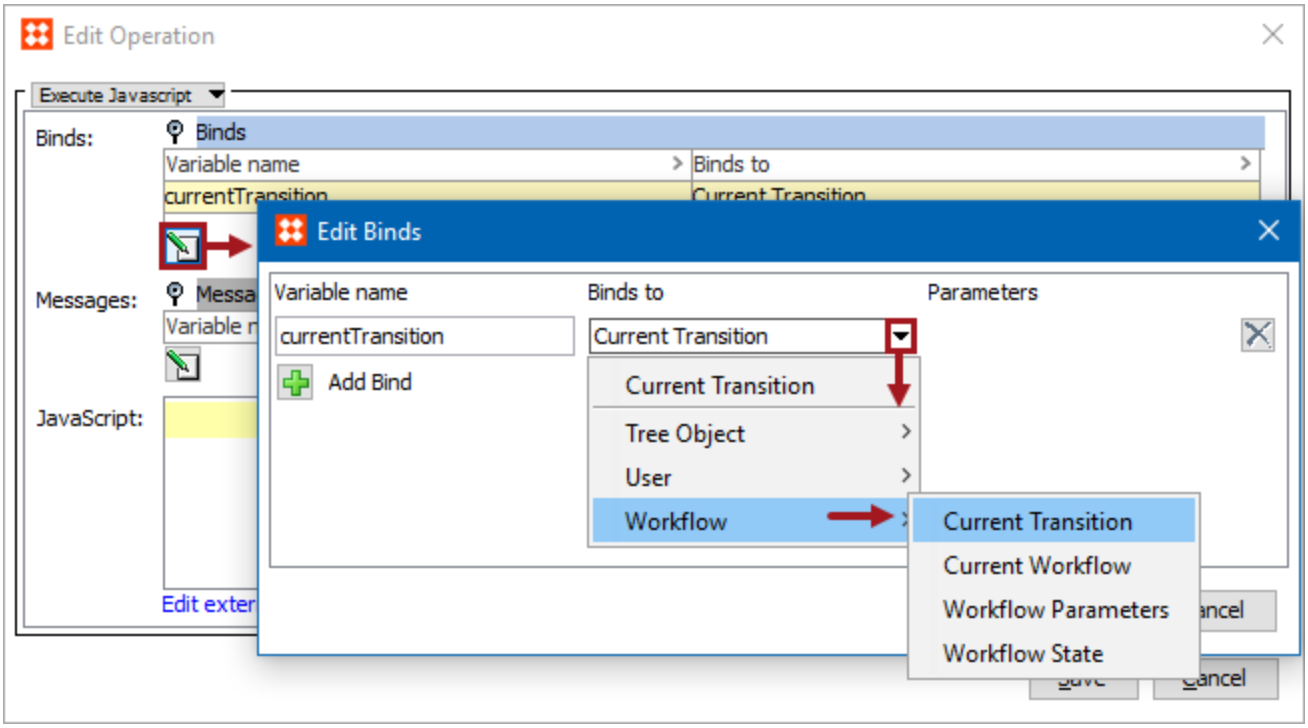
User Group

This binds the selected user group to the action or condition variable. Click the ellipsis button **(…)** in the Parameters field to display the Select User Group dialog. Use Browse or Search to locate a user group, select it, and click the **Select** button.

Workflow Binds

Business rules can use any of the following workflow binds to get workflow-related information like ID, states, title, etc. This bind is only valid in a workflow.

The binds can be found within the 'Binds to' dropdown as shown below.



For an additional bind that is used on a transition in a workflow, see the **e-Signature** documentation. This bind forces users to re-authenticate prior to taking action in a workflow in Web UI, and ensures the security of the data being committed.

Each bind is defined in the sections below.

Configuration

To use any bind:

1. Create a business rule as defined in the **Creating a Business Rule or Library** topic.
2. Edit the business rule as defined in the **Editing a Business Rule** topic.
3. In the Edit Operation dialog, add the bind to a business rule, as defined in the **Adding a Bind** topic.
4. In the Edit Operation dialog, optionally add Messages, as defined in the **Adding a Localized Business Rule Message** topic.
5. In the Edit Operation dialog, add JavaScript to call the bind.

Important: The example scripts should not be used as-is without thorough testing, including updating the script to match object and link types that exist on your system.

Current Transition

This bind is available to track properties of a transition currently being performed in a STEP workflow. This allows JavaScript business rules access to information about the event that started the transition and to the message entered as part of submitting the event.

The bind is only available from JavaScript-based business conditions and actions executed from within the workflow. Those are actions executed 'On Entry,' 'On Transition,' or 'On Exit' and conditions on transitions. The bind is not available from workflow start conditions, actions executed when deadlines are reached, or conditions on conditionally mandatory attributes.

JavaScript action stores the submit message in a workflow variable

Below is an example of a JavaScript action that stores the submit message in a workflow variable. Setup includes the following prerequisites:

- Binding on Current Object named: node
- Binding on Current Workflow named: workflow
- Binding on Current Transition named: currentTranstion
- Simple Workflow Variable named: lastMessage

```
var workflowInstance = node.getWorkflowInstance(workflow);
var msg = currentTransition.getMessage();
if (!msg)
        msg = '[none]';
workflowInstance.setSimpleVariable('lastMessage',msg);
```

JavaScript condition checks for a message when a specific event is submitted

This is an example of a JavaScript condition that checks for presence of message when a specific event is submitted. Setup includes the following prerequisites:

- Binding on Current Transition named: currentTransition
- Business rule Message named: messageRequired

```
if (currentTransition.getEvent() &&
   'submit' == currentTransition.getEvent().getID() &&
   !currentTransition.getMessage()){
        return new messageRequired();
} else {
        return true;
}
```

getEvent() will return the event (object) used to trigger the current transition; should be null if there is no event; getMessage() will return the submit message as a string or null if no message has been supplied.

Current Workflow

This bind is available for conditions evaluated and actions executed from a workflow, giving access to the current workflow. In this example, Current Object is bound to 'currentObject' and Current Workflow is bound to 'currentWorkflow'. The code stores the assignee for the 'Enrich' task in a variable. Notice that this code will only work if the JavaScript is executed from within the workflow and current object is in the 'Enrich' state.

```
var assignee = currentObject.getTaskByID(currentWorkflow.getID
(),"Enrich").getAssignee();
```

Workflow Parameters

This bind allows parameters to be passed to a workflow when it is started. It is typically used for Flatplanner / page planner workflows started from the Flatplanner component

Workflow State

This bind allows access to Workflow State information from a Smartsheet.

JavaScript Considerations

JavaScript is a powerful tool that allows an extensive range of functionality that may be applied to manage and maintain STEP data. The following considerations should be observed to ensure the best use of this tool.

- **Version control:** Since JavaScripts written in STEP are not version controlled, it is important to manually keep a backup of the scripts in a text file, updating them regularly. Saving scripts with a meaningful file name can allow you to revert to a previous version as needed. For example, incrementing a number within the file name like PriceValidation_01.txt.
- **Variables:** Although the Rhino engine used for executing JavaScript is reinitialized prior to the execution of each JavaScript plugin script fragment, it is considered best practice to declare all script variables using the “var” keyword.
- **Code Reuse:** If you plan to construct more than just a few JavaScript-based business rules, business libraries should be used to hold common functions. If libraries are not used, it will be very hard to maintain the business rules as time goes on.
- **Error Handling:** While most of the simple scripts shown as examples do not include error handling, for scripts to be run on a production system, error handling is essential. Common cases that should be handled include attributes that do not exist, or an attribute that does exist but has no value. Both of these cases would return an exception error. For more information, see the **JavaScript Exception Handling** topic.
- **Java vs. JavaScript:** When writing JavaScript Business Rules it is important to understand that the public STEP API is a Java API. Thus all returned objects will be Java objects, not JavaScript objects. For more information, see the **Java vs. JavaScript** topic.

JavaScript Examples

The following examples of JavaScript are used to set an attribute value and to send asset content.

Example

The following is example JavaScript that uses this bind.

Important: The example scripts should not be used as-is without thorough testing, including updating the script to match object and link types that exist on your system.

Set an Attribute Value

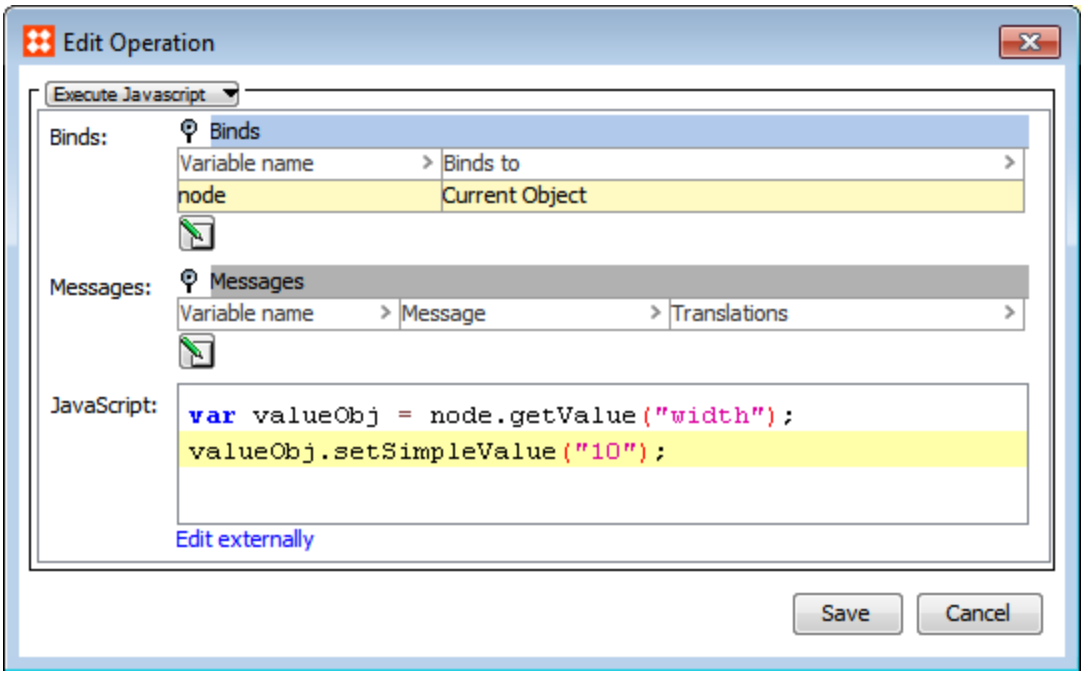
As a simple example, you could use the following JavaScript to set an attribute value:

1. Select Execute JavaScript from the dropdown menu, and add any binds needed for the workflow. In this case, 'node' is being bound to 'Current Object'.
2. Copy and paste, or type in the JavaScript field, to add the business action script.

```
var valueObj = node.getValue("Attribute ID");
valueObj.setSimpleValue("Value for the attribute");
```

   - Replace 'Attribute ID' in the script with the ID of a text attribute that is valid for the object type for which the workflow is made valid.
   - Replace 'Value for the attribute' with the value that the attribute on the object should receive when it passes through the state.

In the following example, 'width' is the attribute ID and '10"' is the value that will be written to the attribute.



With this JavaScript, the users gets the value object for the attribute via the getValue() method called on 'node'. This is a script shorthand for getting the object in the object-in-workflow relation. This value object is then stored in the variable 'valueObj'. On the second line, the setSimpleValue() method is called on the value object, which sets the value.

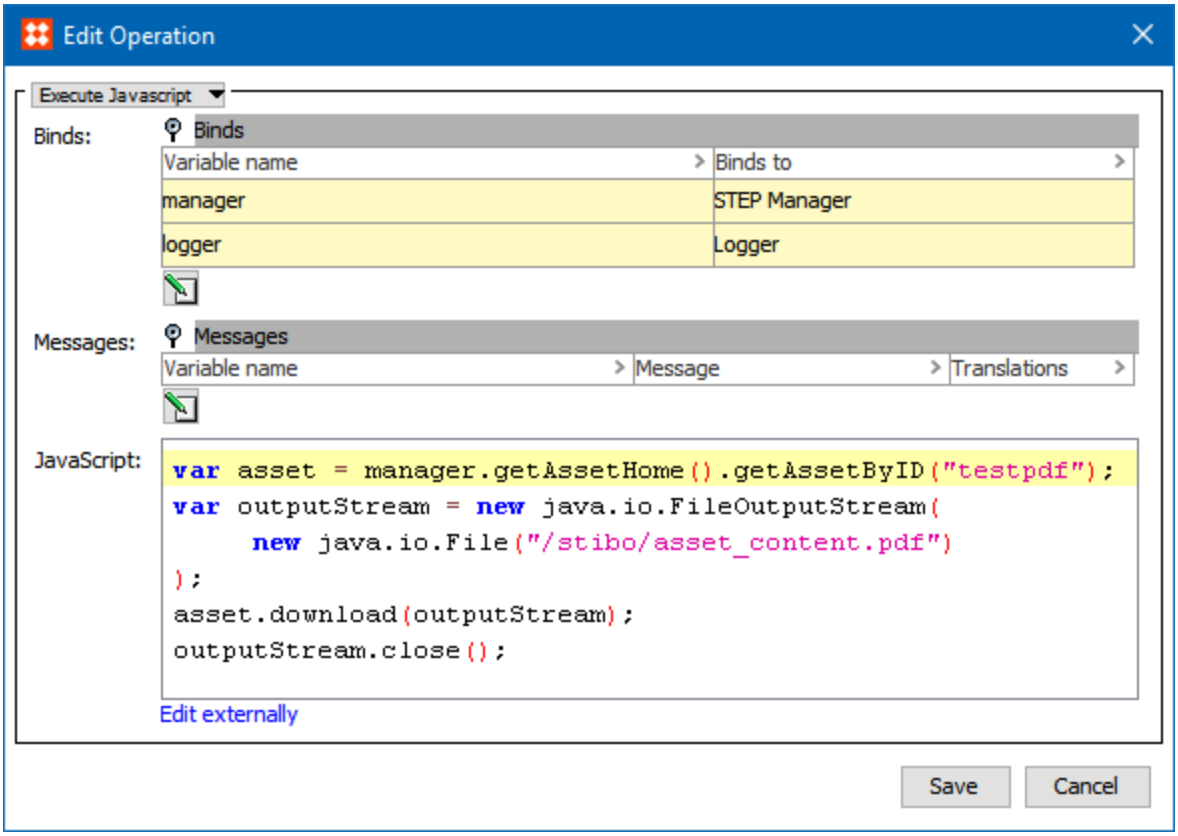
Send Asset Content

Note: This JavaScript operation will not create folders at the destination. To successfully download the file specified by this code, ensure that the needed folders exist. In this example, our STEP application server has a folder called **stibo**.

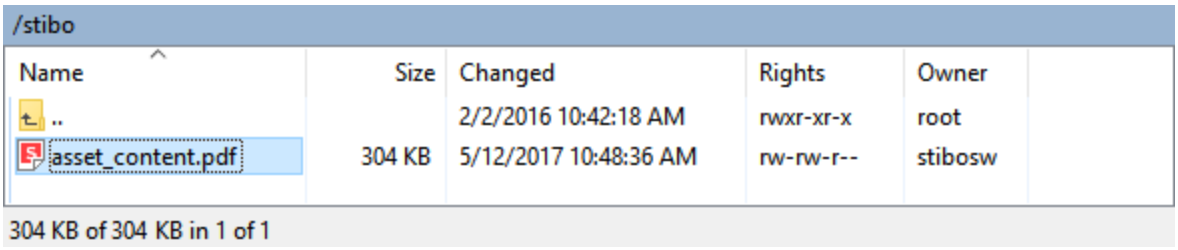
1. As before, select Execute JavaScript from the dropdown menu, and add any binds that are needed for the workflow. In this case, 'manager' is being bound to 'STEP Manager'.
2. Copy and paste, or type in the JavaScript field, to add the business action script.

```
var asset = manager.getAssetHome().getAssetByID("Asset ID");
var outputStream = new java.io.FileOutputStream(
      new java.io.File("Destination")
);
asset.download(outputStream);
outputStream.close();
```

- Replace "Asset ID" in the script with the ID of an asset for which the workflow and business rule is valid.
- Replace "Destination" with hierarchy location on the STEP server of where the file will be downloaded.



After executing this business rule, the file will be downloaded and located where specified on the STEP server.



For information on handling exceptions, see **JavaScript Exception Handling**.

JavaScript Exception Handling

When writing JavaScript business rules, it is important that 'try…catch' statements are designed correctly. 'Try...catch' statements should not swallow (catch and ignore) exceptions that should cause changes made by the script to be rolled back and cause the business rule to fail.

Important: Carelessly swallowing exceptions will lead to derived errors, which usually makes it very hard to determine the root cause.

The sample code in the table below demonstrates a way to log exceptions in JavaScript without swallowing those that should be handled by the framework.

Correct	Incorrect

`try {`
`  // Some code`
`} catch (e) {`
`  logger.info(e);`
`  throw(e); // REQUIRED`
`}`
	`try {`
	`  // Some code`
	`} catch(e) {`
	`  logger.info(e);`
	
	`}`

The only types of JavaScript exceptions that may be caught and handled locally are:

- Checked exceptions thrown by an invoked method (including any subclasses of the specified checked exception) (see the example below)
- Exceptions not generated by calls to the STEP API

Important: Any other exception must always be re-thrown if caught, or not caught at all.

Correct Handling of Checked Exceptions

The code snippet below exemplifies how a checked exception can be caught and handled locally. Notice the necessary Rhino specific '**e.javaException**' notation.

```
try {
     currentObject.createReference(targetAsset, primaryImageRefType.getID());
} catch (e) {
     if (e.javaException instanceof
com.stibo.core.domain.UniqueConstraintException) {
          logger.info("Reference could not be created");
     } else {
          throw(e); // All other exceptions MUST be re-thrown
     }
```

```
}
```

Localized Messages for JavaScript Business Rules

When adding an **Evaluate JavaScript** business condition or an **Execute JavaScript** business action, messages and translations can be specified. The JavaScript 'Return' or 'Throw' statements are used to display a business rule message.

Return Statement

The return statement is for conditions used to indicate whether the condition is true or false. When Boolean true is returned, the condition evaluates to true. Any other return value will make the condition evaluate to false.

To have a message displayed to the user when a condition evaluates to false, two options are available. The first and most simple is to return a String, as shown in the following example:

```
return "Object is not ready";
```

The other option is to return a translatable error message object, making it possible to have the message displayed in a language matching the UI locale. For more information on adding messages, see **Adding a Localized Business Rule Message** in the **Business Rules** documentation.

The Return statement is not available in a business action.

Throw Statement

It is possible to deliberately throw exceptions from both actions and conditions if error states are encountered. The exception message can be a translatable message object. For more information on adding messages, see **Adding a Localized Business Rule Message** in the **Business Rules** documentation.

Adding a Localized Business Rule Message

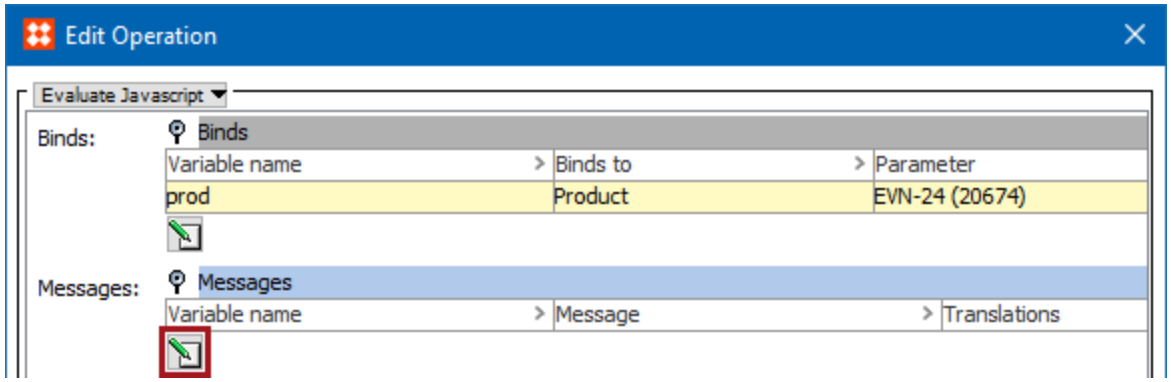
A message can display static text or dynamic text, based on the value of a variable.

Note: This option uses locales. It is not context-dependent. Contact your Stibo Systems representative to display additional language locales on your Web UI or workbench.

1. Open the Edit Operation dialog for an existing JavaScript business rule. For more information, see the **Editing a Business Rule** topic.

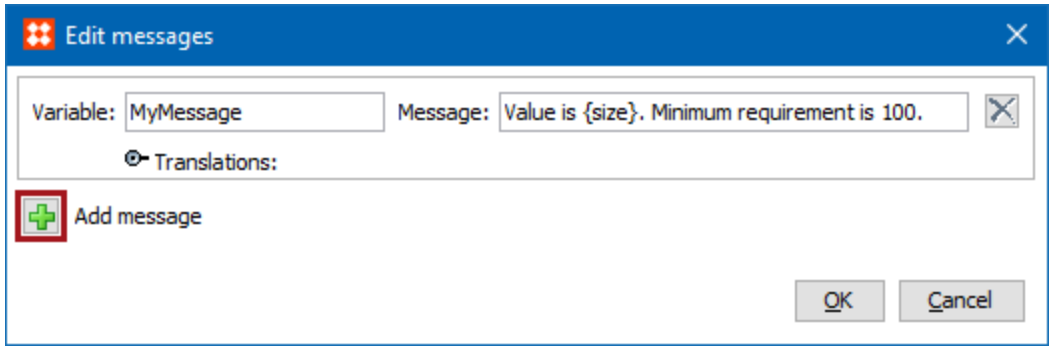
   In this example, we have selected a condition and are using the 'return' statement to display the message.

2. On the Edit Operation dialog, for the Messages parameter, click the **Edit** button ( ) to display the 'Edit messages' dialog.



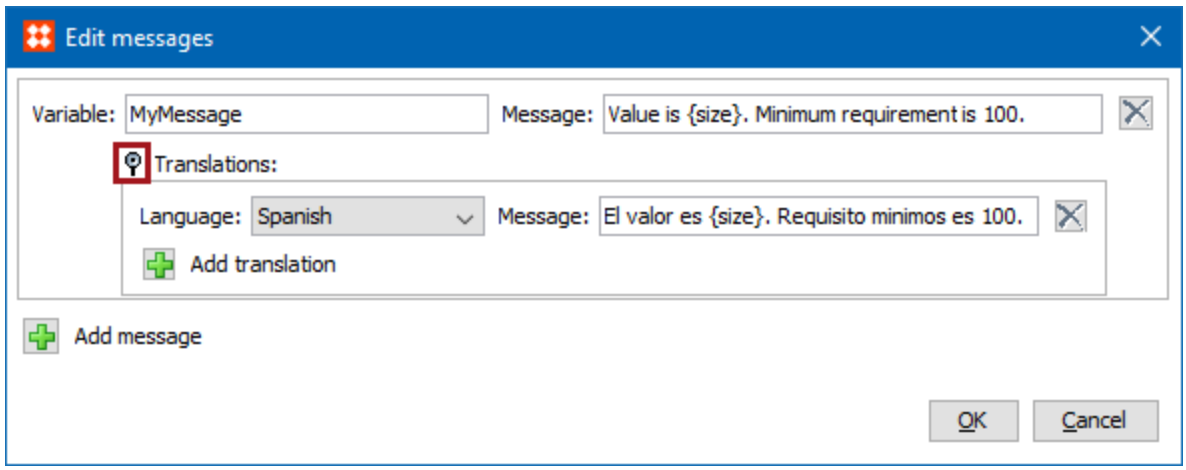
3. Click the **Add message** button ( ) and add a variable name and the message text.

   A variable message can include the name of the business rule or any other data that would be helpful in resolving the problem reported by the message.

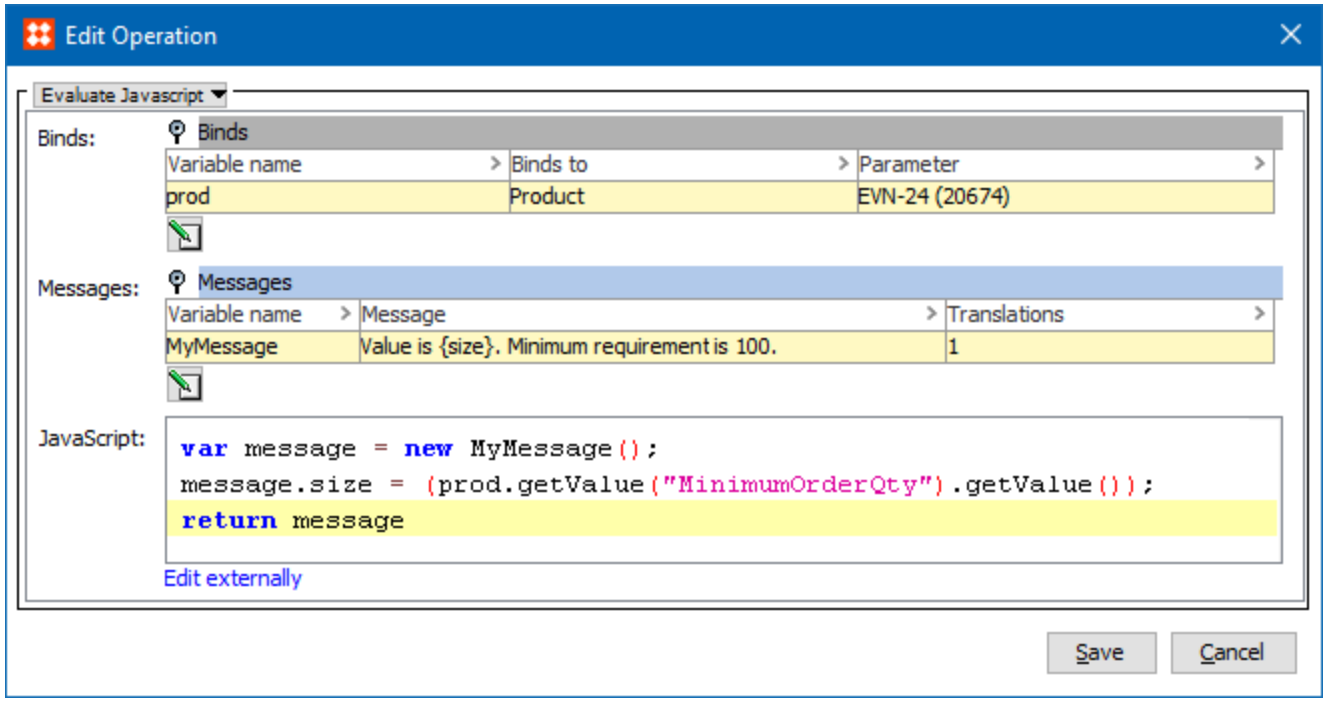


   A variable tag within the message text is optional. In this example, the tag is {size} and will return the value of the attribute from the bound product.

4. Click the Translations flipper ( ) to open it ( ).
5. Click the **Add translation** button ( ) to add the translated text. Multiple translations can be added. Do not translate variable tags.

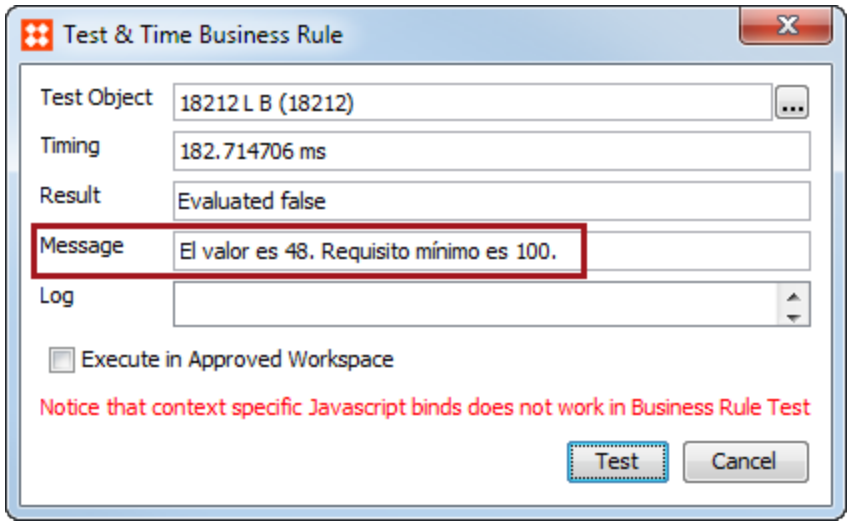


6. Click **OK** to close the 'Edit messages' dialog.
7. On the Edit Operation dialog, for the JavaScript parameter, add JavaScript code to display the message. In this example, our business condition will use the 'return' statement.



8. If you added a translation, test the results by logging in to the locale that matches the language for the message using the STEP Workbench or the Web UI. For our example, we select the Spanish locale.

9. Test the business rule or run a workflow that uses it to display the translated message. For information, see the **Testing a Business Rule** topic.



Business Action: Generate Match Codes

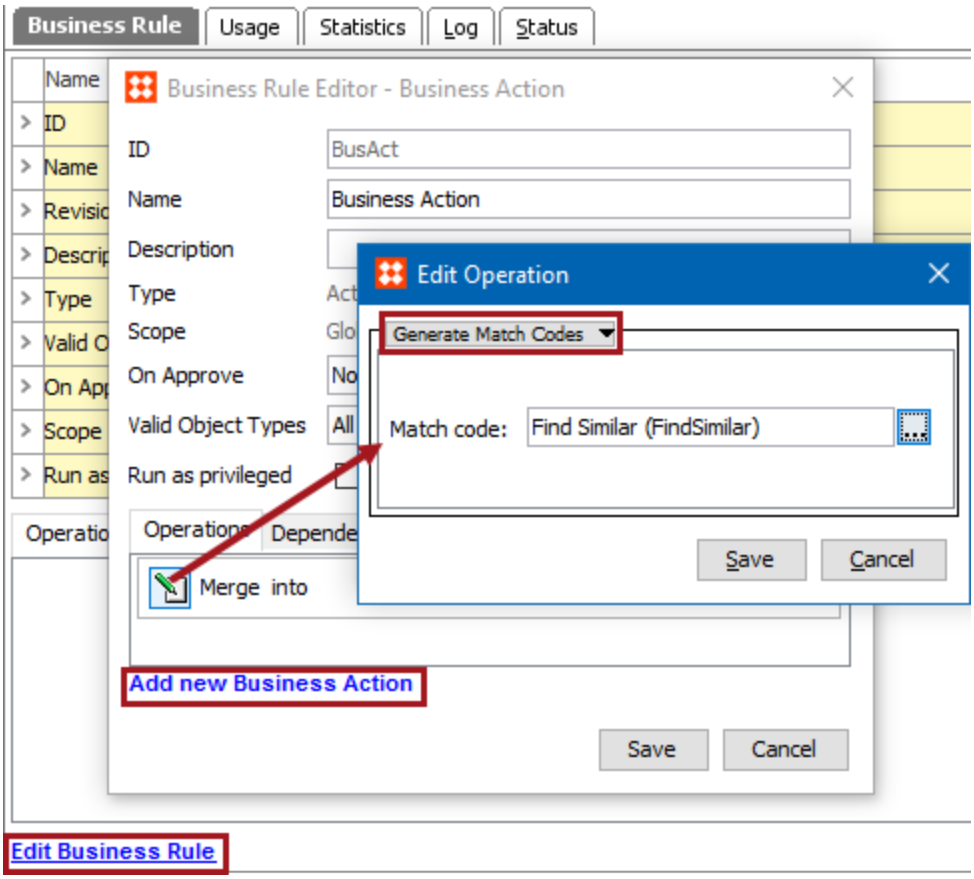
This action generates match code values for objects or nodes. Match codes have a formula, that when executed by a business rule, creates match codes for objects. The match codes are displayed on the System Setup match code object, on the Match Code Values tab.

Prerequisites

Before using this action:

1. Ensure match code objects exist as defined in the **Configuring Match Codes** section of the **Matching, Linking, and Merging** documentation.
2. Create a business rule as defined in the **Creating a Business Rule or Library** topic.

Configuration



1. On the Edit Operation dialog, select the Data Quality > **Generate Match Codes** option from the dropdown.
2. For **Match Code**, click the ellipsis button **(…)** to display the Select a Match Code dialog, and the relevant match code.
3. Click the **Save** button to add the operation to the business rule editor.

Business Action: Initiate Items In STEP Workflow

This action can initiate a valid object or objects into the selected workflow.

For example, consider an object that is in State A, when it moves to State B within the workflow, the object should also be initiated to another workflow. This can be achieved by the Initiate Items in STEP Workflow action by configuring it at the 'On entry' of a state of the workflow where it is already present.

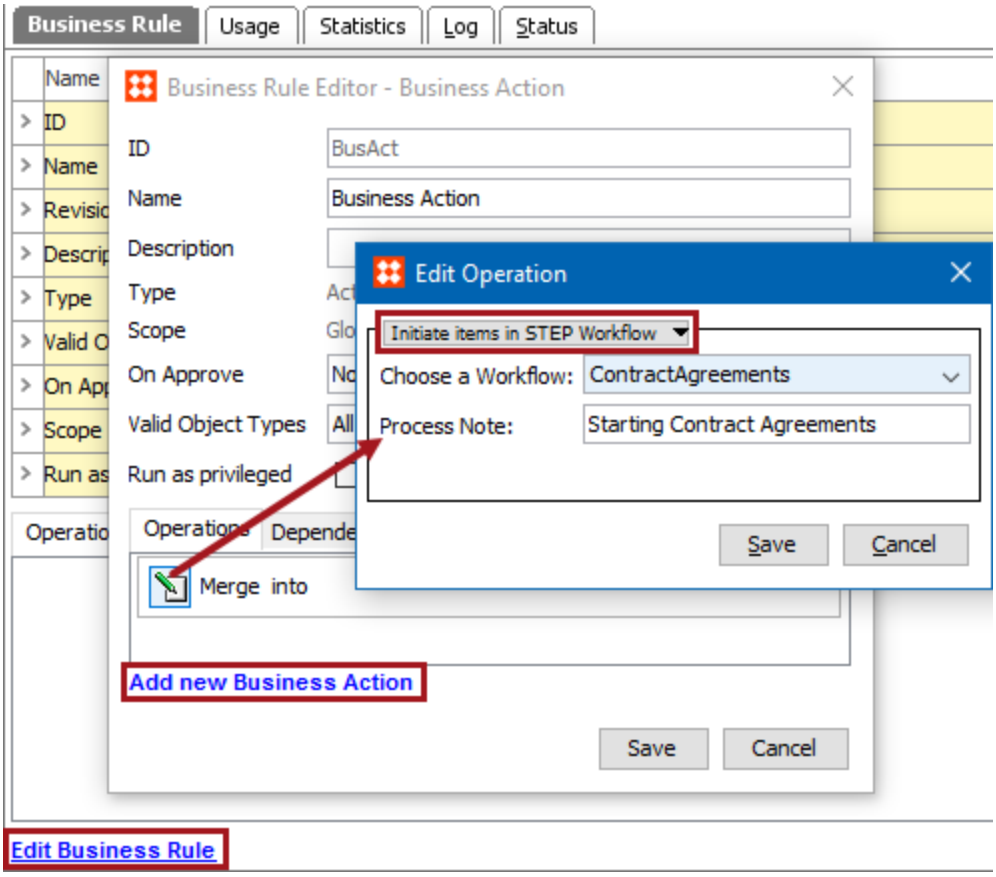
For more information, see the **Business Rules and Workflows** topic in the **Workflows** documentation.

Prerequisites

Before using this action:

1. Ensure the workflow exists, and the object is valid for the workflow.
2. Create a business rule as defined in the **Creating a Business Rule or Library** topic.

Configuration



1. On the Edit Operation dialog, select the Workflow > **Initiate Items in STEP Workflow** option from the dropdown.

2. For **Choose a Workflow**, select a workflow from the dropdown.
3. For **Process Note**, if required add a note to be displayed in the state log for the objects and the workflow.
4. Click the **Save** button to add the operation to the business rule editor.

Business Action: Merge Attribute Values

This action copies the value of one attribute into another attribute. You can specify whether to keep or overwrite existing values. This action can also be used for local attribute link metadata values.

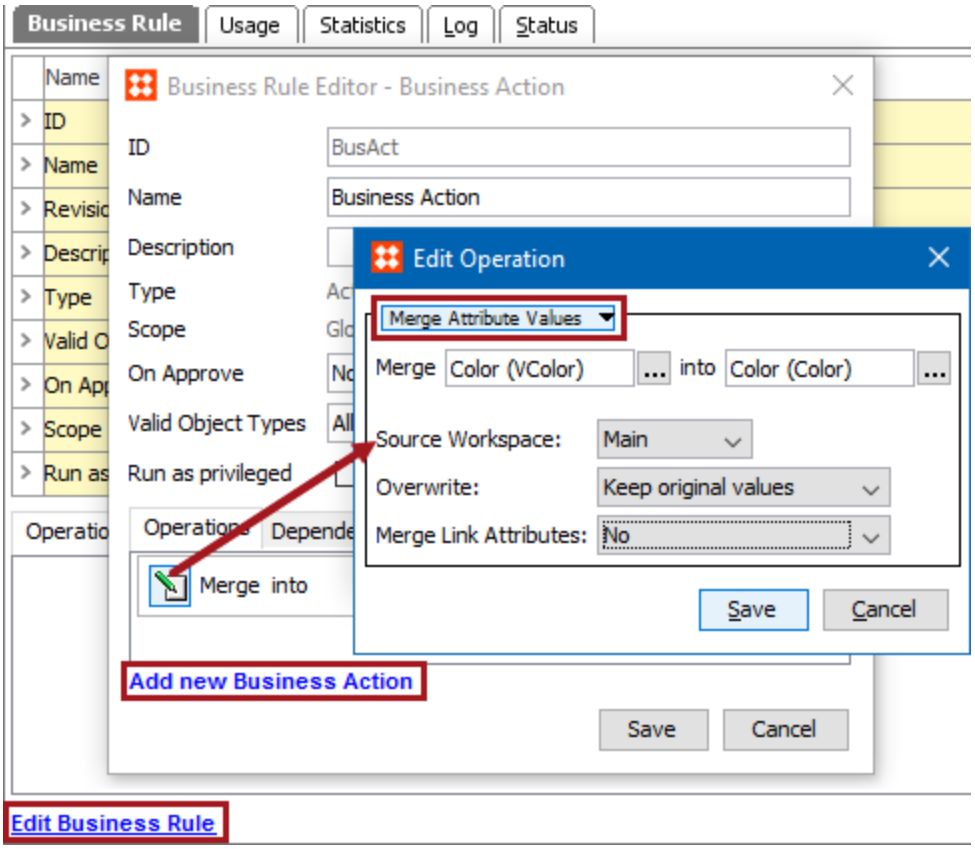
For example, while performing data cleanup activities, to consolidate duplicate attributes, this action can be used to copy values from the duplicate attribute to the attribute that should be used to store the value.

Prerequisites

Before using this action:

1. Ensure the source and target attributes are valid for all of the same object types, and for the object on which the business action will be run.
2. Ensure the source and target attributes are valid for all of the same product links and classification links.
3. Ensure the source and target attributes are valid for all of the same reference types.
4. Ensure the source and target attributes have the same dimension dependencies.
5. Create a business rule as defined in the **Creating a Business Rule or Library** topic.

Configuration



Note: The Merge Attribute Values operation does not delete any attributes, it can only update values. If desired, manually delete the source attribute once you confirm the merged data.

1. On the Edit Operation dialog, select the Workflow > **Merge Attribute Values** option from the dropdown.
2. For the **Merge** parameter, click the ellipsis button **(…)** to display the Select Attribute dialog, choose the attribute to copy values from, and click the **Select** button.
3. For the **into** parameter, click the ellipsis button **(…)** to display the Select Attribute dialog, choose the attribute to copy values to, and click the **Select** button.
4. For the **Source Workspace** parameter, use the dropdown to select the workspace that holds the values to be copied. Values are always pasted into the Main workspace.
5. For the **Overwrite** parameter, use the dropdown to select one of the following options. Validation errors are displayed when a value cannot be modified. See an illustration of the results in the **Merge Results** section below.
   - **Overwrite existing values** means the source value overwrites the target value, except when the source attribute is blank. If the source attribute is blank, the target value is not changed.
   - **Keep Original values** means the source value is only copied to the target attribute when the target attribute is blank. If the target attribute contains a value, the target value is not changed.
6. For **Merge Link Attributes**, use the dropdown to select one of the following options. If the selected attributes are both used as an attribute on a reference, a product classification link, or an attribute link, merging of those values are based on the **Yes** or **No** set in this parameter, as well as the other parameters on the operation. This option is not relevant when values do not exist on references for the selected attributes.
7. Click the **Save** button to add the operation to the business rule editor.

Merge Results

As indicated in the following table, the source attribute value, target attribute values, and the overwrite setting together determine the end result of this action.

Source Attribute Value	Target Attribute Value	Overwrite Setting	Target After Merge
<empty>	OriginalValue	<any>	OriginalValue
NewValue	<empty>	<any>	NewValue
NewValue	OriginalValue	[Overwrite Existing]	NewValue
NewValue	OriginalValue	[Keep Original Values]	OriginalValue

For an example of merging attribute values with this action, see the **Merge Attribute Values Operation** section of the **Attribute Values Operations for Bulk Updates** topic in the **Bulk Updates** documentation.

Business Action: Overlap Analysis

This action compares records of predefined format to each record with the same object type, by specifying a matching algorithm.

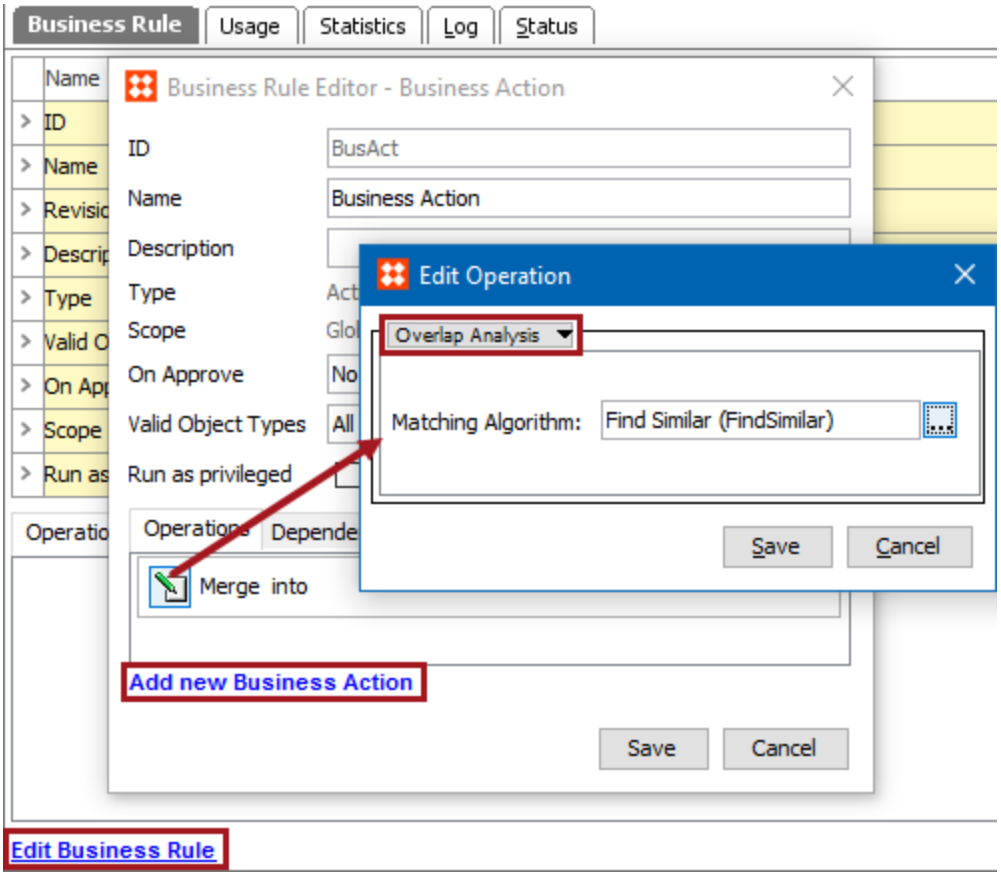
Important: This operation is only intended to be used together with an import running in Test mode. In this mode, statistics are reported on how many objects in import file will match existing objects in STEP, without actually importing the data.

Prerequisites

Before using this action:

1. Create a business rule as defined in the **Creating a Business Rule or Library** topic.

Configuration



1. On the Edit Operation dialog, select the **Overlap Analysis** option from the dropdown.
2. For the **Matching Algorithm** parameter, click the ellipsis button **(...)** to display the Select a Matching Algorithm dialog, select an algorithm, and click the **Select** button.
3. Click the **Save** button to add the operation to the business rule editor.

Business Action: Reference Other Business Action

This action can create a reference to another business action without additional configuration.

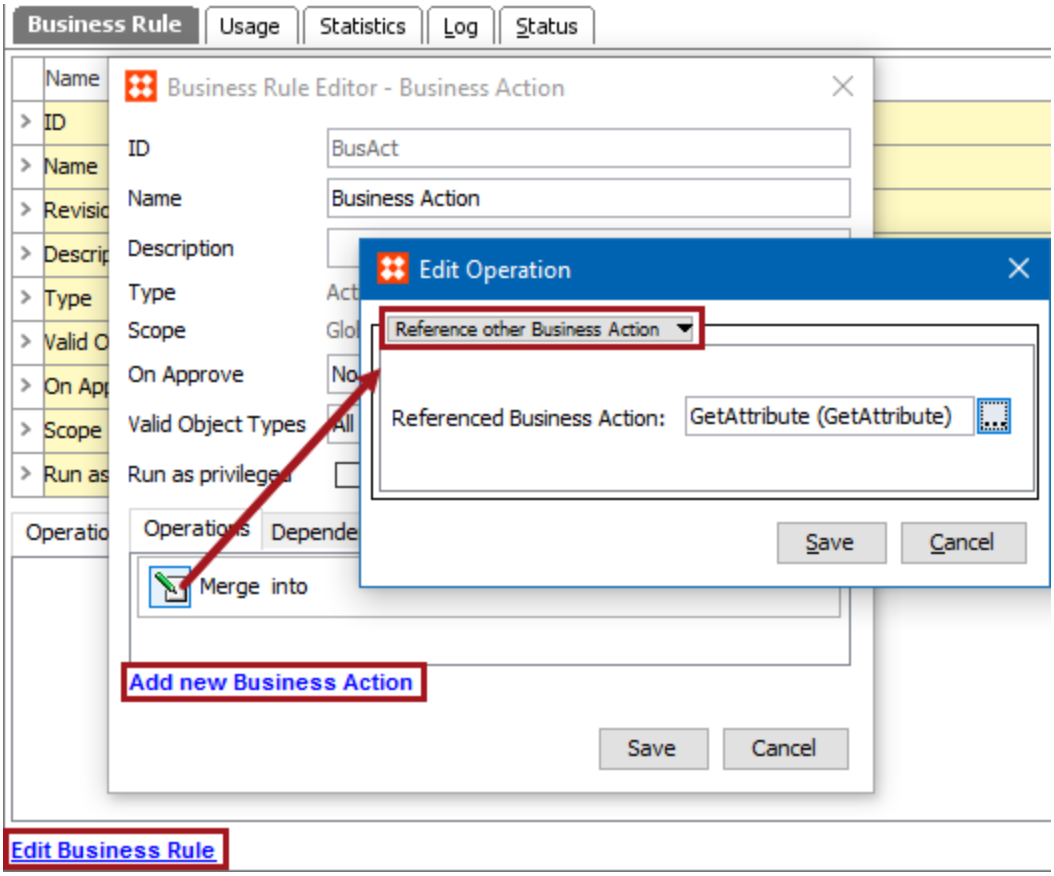
For example, a business action that updates an object name needs to be executed within other business rules run by the Import Manager or in a Web UI.

Prerequisites

Before using this action:

1. Ensure the business action is valid for the node(s) where it will be executed.
2. Create a business rule as defined in the **Creating a Business Rule or Library** topic.

Configuration



1. On the Edit Operation dialog, select the **Reference Other Business Action** option from the dropdown.
2. In the **Referenced Business Action** parameter, click the ellipsis button **(…)** to display the Select Action dialog, select a business action, and click the **Select** button.
3. Click the **Save** button to add the operation to the business rule editor.

Business Action: Remove Attribute Link

This action removes an attribute link from a product.

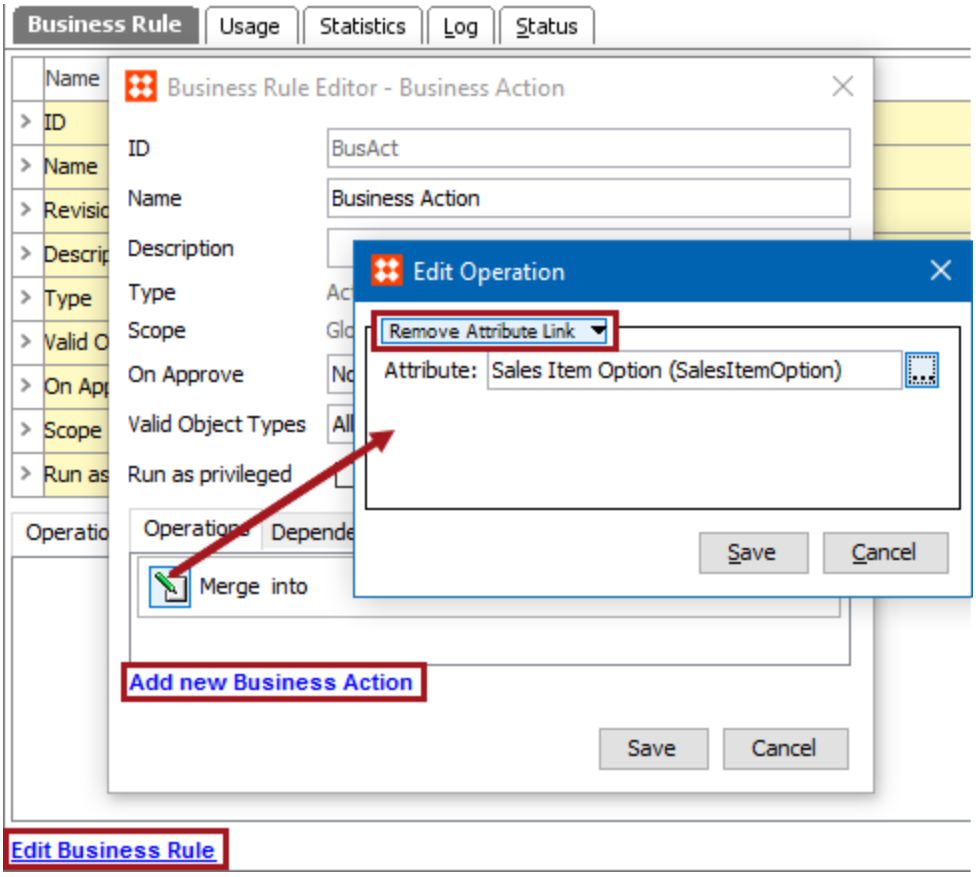
For example, the 'Offer' attribute is linked to a number of products during a seasonal special offer promotion. Once the promotion has ended, the attribute should immediately be removed from all products. This can be achieved by creating a collection of all nodes that include the attribute, and then running a bulk update with the Remove Attribute Link business action.

Prerequisites

Before using this action:

1. Ensure the valid attribute is linked to the product. For more information, see the **Product Attribute Link Type** topic in the **System Setup / Super User Guide** documentation.
2. Create a business rule as defined in the **Creating a Business Rule or Library** topic.

Configuration



1. On the Edit Operation dialog, select the References and Links > **Remove Attribute Link** option from the dropdown.

2. For the **Attribute** parameter, click the ellipsis button **(…)** to display the Select Attribute dialog, choose the attribute that should be removed and click the **Select** button.
3. Click the **Save** button to add the operation to the business rule editor.

Business Action: Remove Object from STEP Workflow

This action can remove the object or objects from the selected workflow. For example, the object should be removed from the workflow when it reaches the final state within a workflow.

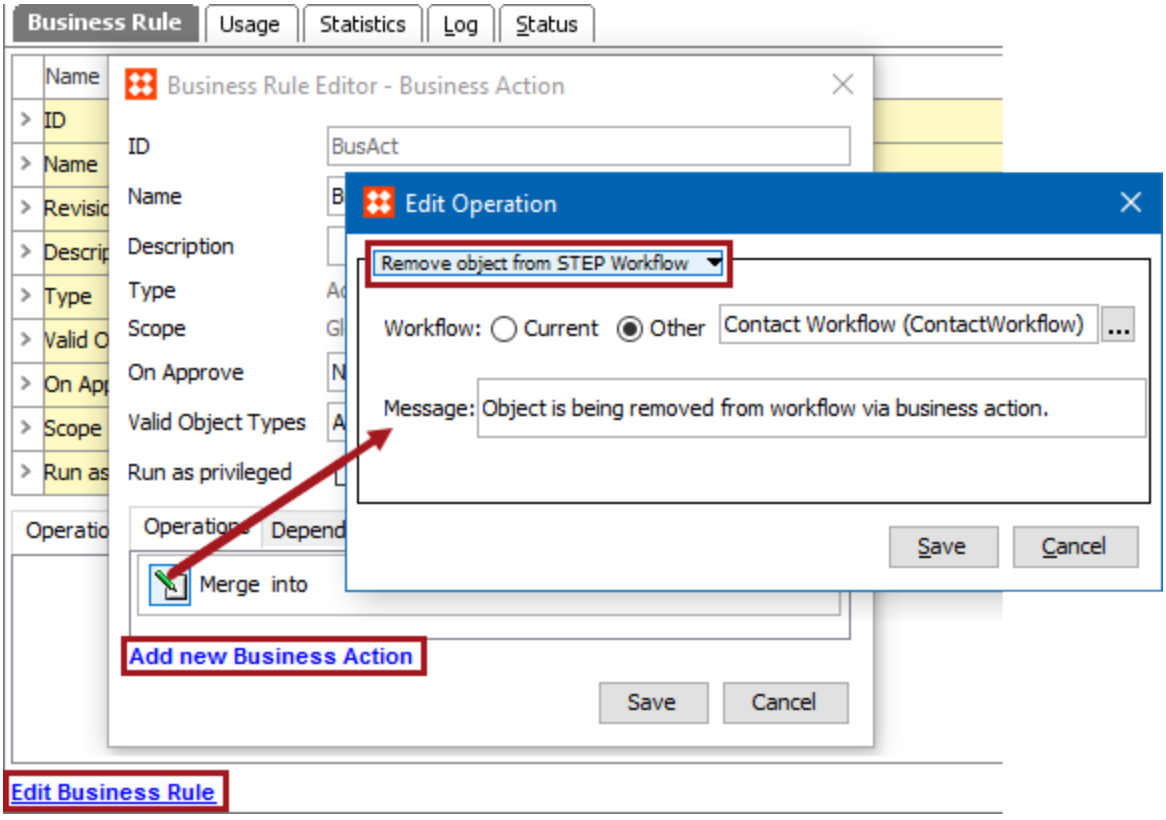
Business actions can also be added within the STEP Workflow Designer. Edit the workflow state or transition for the business rule, and select it on the appropriate tab. For a state, a business rule can be selected on the following tabs: On Entry, On Exit, and Escalation. For a transition, a business rule can be selected on the following tabs: Condition and On Transition.

Prerequisites

Before using this action:

1. Review information about using business rules in workflows as addressed in the **Business Rules and Workflows** topic of the **Workflows** documentation.
2. Create a business rule as defined in the **Creating a Business Rule or Library** topic.

Configuration



1. On the Edit Operation dialog, select the Workflow > **Remove object from STEP Workflow** option from the dropdown.
2. For the **Workflow** parameter, select to use the current workflow or a different one as follows:
   - Click the **Current** radio button to indicate that the workflow in progress when the business rule is executed is used.
   - Click the **Other** radio button, and then click the ellipsis button **(…)** to display the Select Workflow dialog, choose the workflow that includes the object that should be removed. Click the **Select** button.
3. For the **Message** parameter, type a message that will be displayed in the state log.
4. Click the **Save** button to add the operation to the business rule editor.

Business Action: Remove Reference

This action removes a reference to a specific target or all references of a specific type.

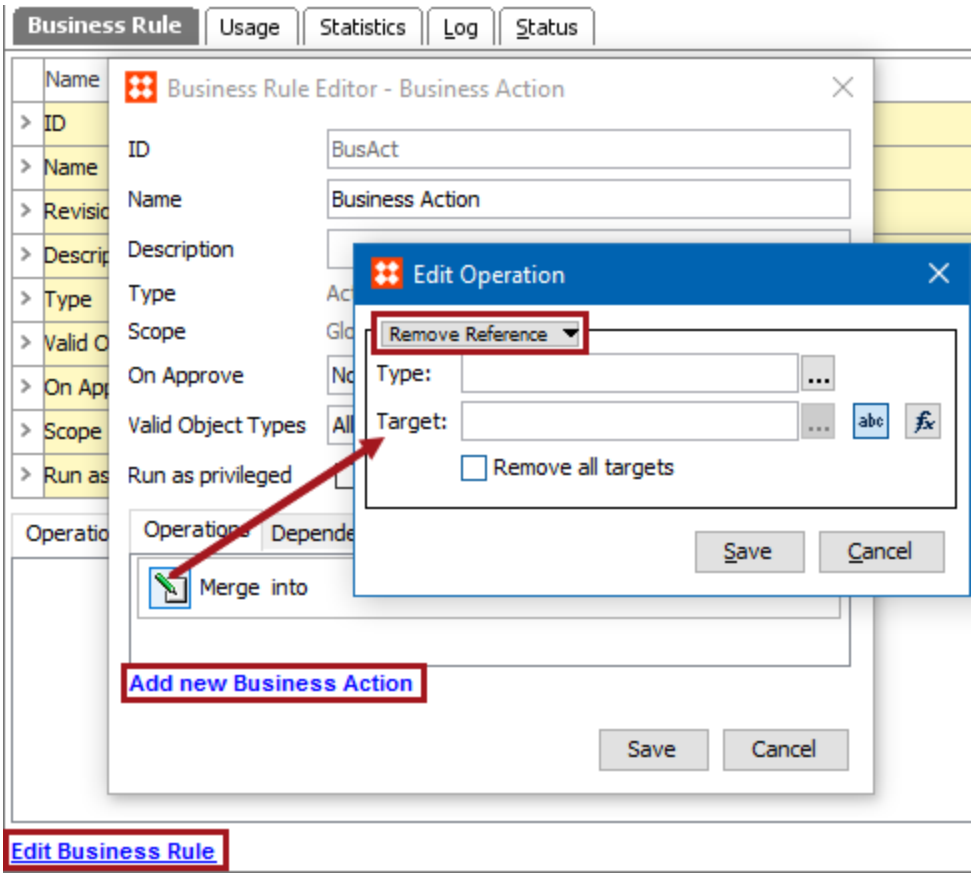
For example, the 'free_objects' reference type is used to indicate a seasonal sales offer items. When the sale period has ended, removal of all 'free_objects' references must be removed. This can be achieved by creating a collection of all nodes that include the reference, and then running a bulk update with the Remove Reference business action.

Prerequisites

Before using this action:

1. Create a business rule as defined in the **Creating a Business Rule or Library** topic.

Configuration



1. On the Edit Operation dialog, select the References and Links > **Remove Reference** option from the dropdown.
2. For the **Type** parameter, click the ellipsis button **(…)** to display the Select Reference Type dialog, choose the reference type that should be removed and click the **Select** button.

3. For the **Target** parameter, use one of these methods to select the relevant target. Leave this parameter blank to remove all references of a specific type without limiting removal to a node. The target must be in accordance with the selected type, otherwise an error is displayed.
   - Click the 'abc' button (), click the ellipsis button **(…)** to display the Select Reference Target picker, select a valid target, and click the **Select** button.
   - Click the 'fx' button (), click the ellipsis button **(…)** to display the Function Editor dialog, write a calculation to select a valid target, and click the **Select** button. For more information, see the **Using Function Editor** topic in the **System Setup / Super User Guide** documentation.
4. For the **Remove all targets** parameter, set the checkbox as follows:
   - Checked with a blank Target parameter means all references of the specified type are removed from the objects on which the business rule runs.
   - Unchecked means references of the specified type are removed from the object on which the business rule runs.
5. Click the **Save** button to add the operation to the business rule editor.

Business Action: Send Email

This action can send emails without requiring JavaScript. Multiple email addresses are allowed and are resolved when the business action is run.

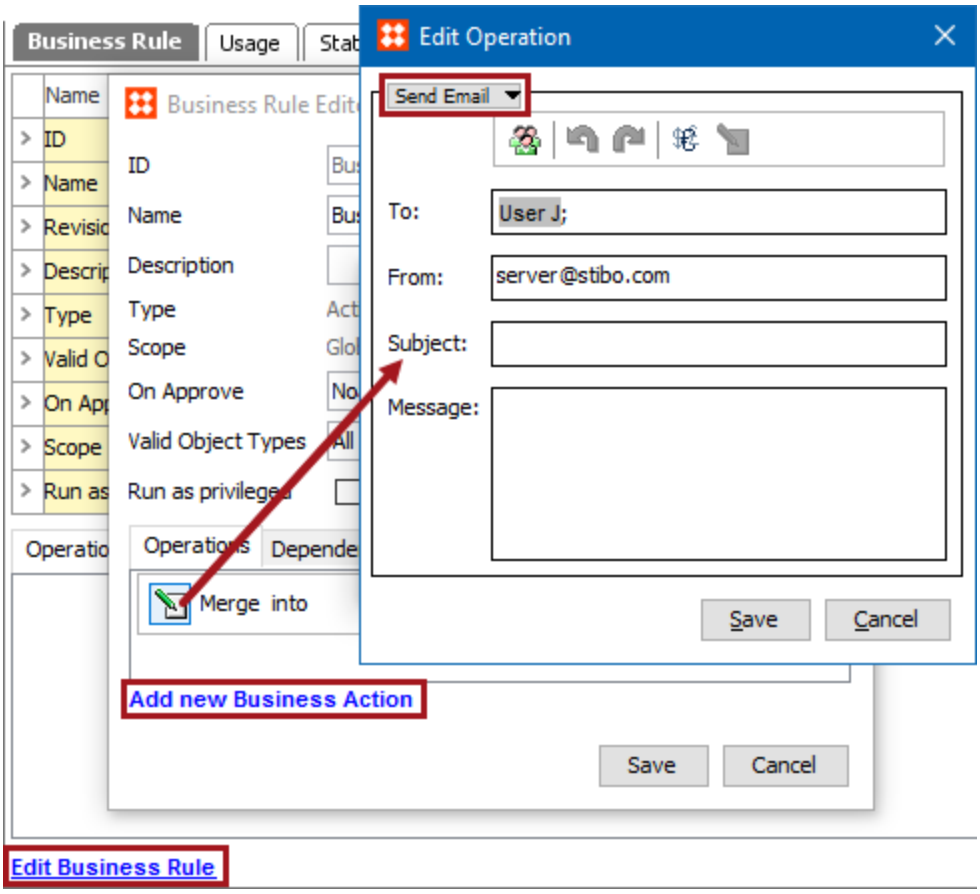
For example, an email can be sent when an object moves from one state of a workflow to another, or upon successful approval of an object.

Prerequisites

Before using this action:

1. Ensure a Simple Mail Transfer Protocol (SMTP) server is configured in the executing system.
2. Ensure any STEP users who are to receive email have an accurate email address in STEP. This is stored on System Setup > User & Groups > open the group > select the user to display the editor. On the User tab, under the Description flipper, the E-Mail parameter should display the email to be used.
3. Create a business rule as defined in the **Creating a Business Rule or Library** topic.

Configuration



1. On the Edit Operation dialog, select the **Send Email** option from the dropdown.

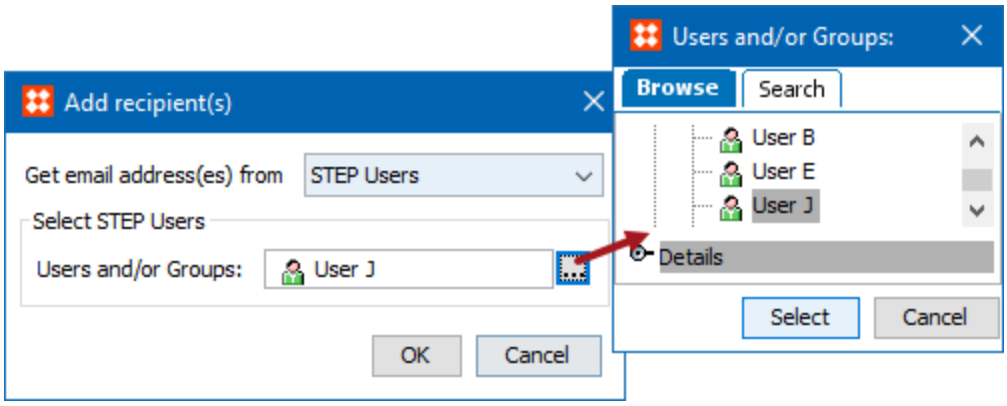
 Action buttons are enabled as the appropriate field is selected or action is performed.

2. For the **To** parameter, click into the field and choose a method to provide the relevant email addresses, using the required format: user@domain.com. If an address uses the right format, errors are not detected until the business action is executed.

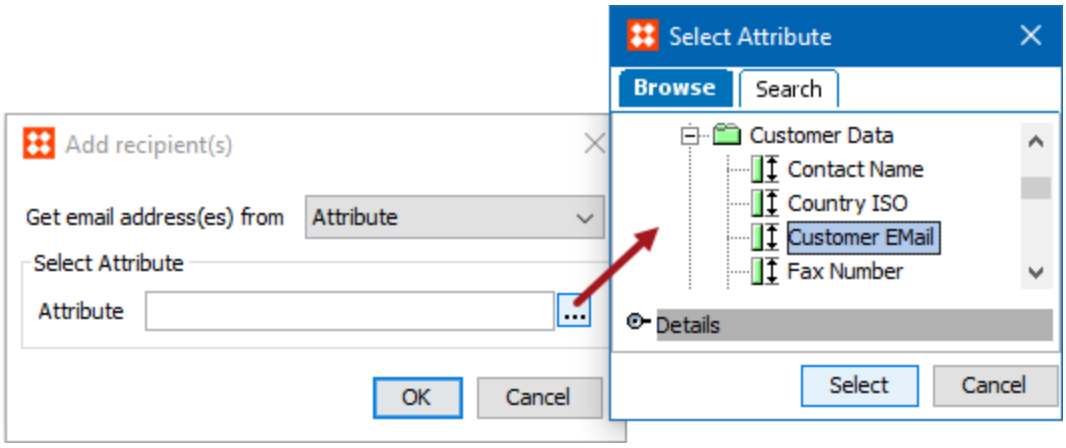
   - Manually type one or more email addresses, separated by semicolon (;), comma (,), or a space ( ). Use the undo and redo buttons () as needed. Special characters can be added using the rich text editor ().
   - Click the **Add recipient** button () to display the Add Recipients dialog.

   On the Add Recipients dialog, for the **Get email address(es) from** parameter, select an option and supply the required information:

   - When **STEP Users** is selected, the **Users and/or Groups** parameter is displayed. Click the ellipsis button **(…)** to display the Users and/or Groups dialog. Use Browse or Search to select one or more users and/or groups, and click the **Select** button.

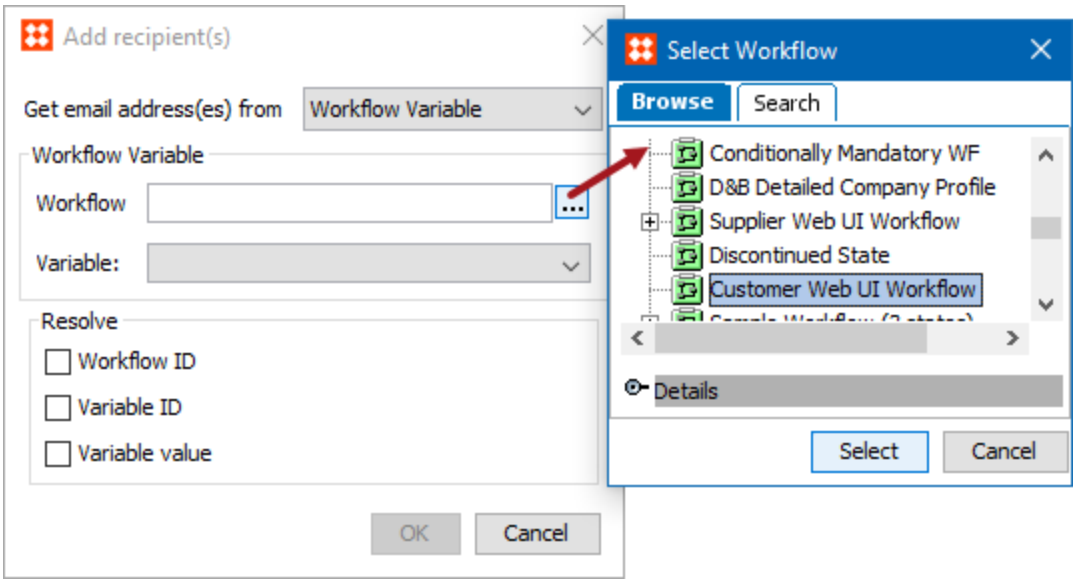


   - When **Attribute** is selected, the Select Attribute parameter is displayed. An email address can be obtained from an attribute of an object on which the business action is executed. Click the ellipsis button **(…)** to display the Select Attribute dialog. Select an email attribute, and click the **Select** button.



- When **Workflow Variable** is selected, the Workflow Variable and Resolve groups of parameters are displayed. An email address can be read from a workflow variable. This requires that the object the business action is executed on has been initiated in the specified workflow. For more information on workflow variables, see the **Workflow Variables** topic in the **Workflows** documentation.

  Click the ellipsis button **(…)** to display the Select Workflow dialog. Choose a workflow with variables and click the **Select** button. The available variables are displayed in the Variable dropdown.

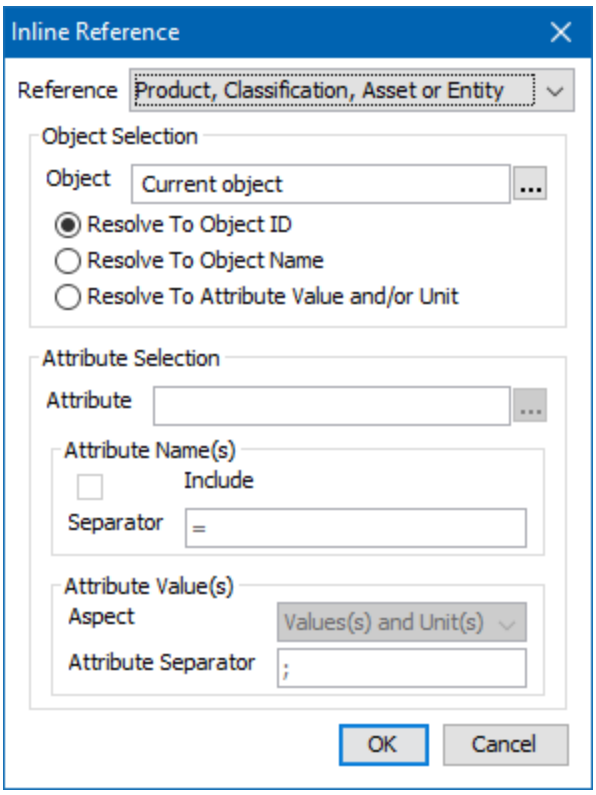


3. For the **From** parameter, the default address specified in the configuration properties file is displayed. If necessary, change the email address. Use the undo and redo buttons ( ) as needed.
4. For the **Subject** parameter, optionally type the text for the email subject. No error is returned when the subject is left blank. Use the undo and redo buttons ( ) as needed.
5. For the **Message** parameter, the following methods are availalbe to add text for the email message. Use the undo and redo buttons ( ) as needed.
   - **Special characters** can be added using the rich text editor button ( ).
   - **Inline references** can provide the recipient with information about which change caused the email to be sent. Click the insert inline reference button ( ) to display the Inline Reference dialog.

Note: It is possible to refer to data that may not be relevant for all executions of the business action.

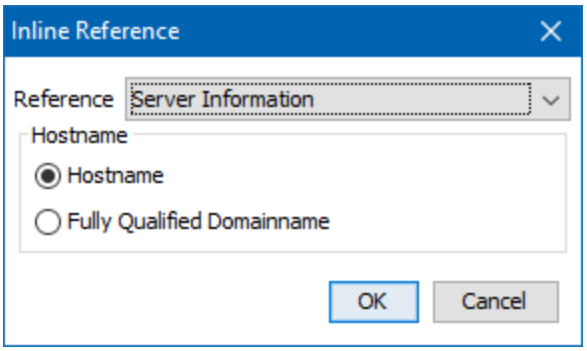
For the **Reference** dropdown, make a selection to update the dialog with the necessary parameters.

For **Product, Classification, Asset or Entity** references, the Object Selection and Attribute Selection groups of parameters are displayed. You can select the current object (the object the business action is executed on) as well as other specific objects.



For **Workflow Variable** references, the Workflow Variable and Resolve groups of parameters are displayed. The object the business action is executed on must have been initiated in the specified workflow. Refer to the Workflow Variable section above for details on the parameters.

For **Server Information** references, the Hostname group of parameters are displayed.



6. Click the **Save** button to add the operation to the business rule editor.

Configuration Notes

- If an invalid email address is found during execution, the action fails and results in an error. This can happen when part of an address has been left out by mistake, or if an attribute contains an unexpected value.

- If during execution no recipient is specified or if no valid email address can be found for any of the specified recipients, an error occurs.
- If an inline reference cannot be resolved during execution, an error occurs.

Business Action: Send Republish Event

This action sends a 'republish' event to an event queue. This allows events to be generated on demand.

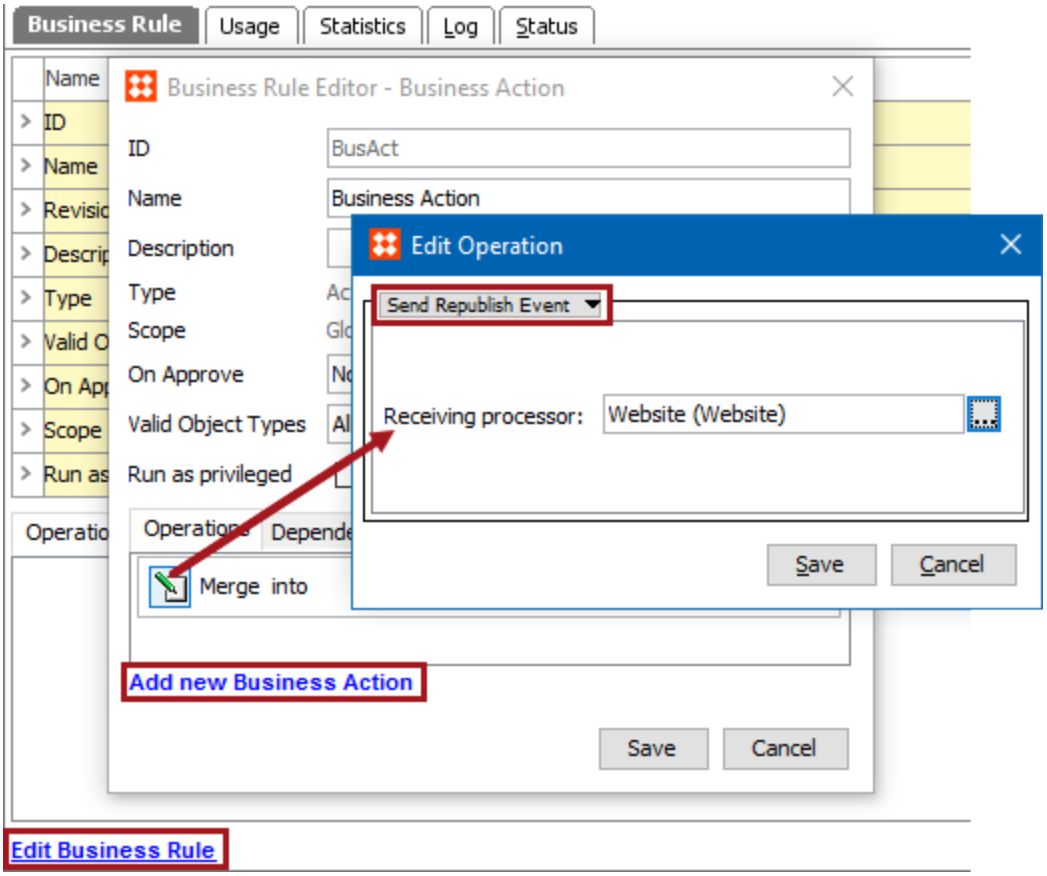
For example, events are required for an external system to initially publish an entire hierarchy. This can be achieved by running a business action with this operation on the hierarchy. The generated events are sent to the specified queue.

Prerequisites

Before using this action:

1. Ensure the event queue is enabled and set to Read Event.
2. Create a business rule as defined in the **Creating a Business Rule or Library** topic.

Configuration



1. On the Edit Operation dialog, select the **Send Republish Event** option from the dropdown.
2. For the **Receiving Processor** parameter, click the ellipsis button **(…)** to display the Select Processor for Events dialog, choose an OIEP or Event Processor that has an event queue, or choose an actual event queue object. Click the **Select** button.

3. Click the **Save** button to add the operation to the business rule editor.

Business Action: Set Attribute Value

This action can set the value of an attribute on the object where the action is being executed to a static value, match another attribute value on the same object, or as a workflow variable value.

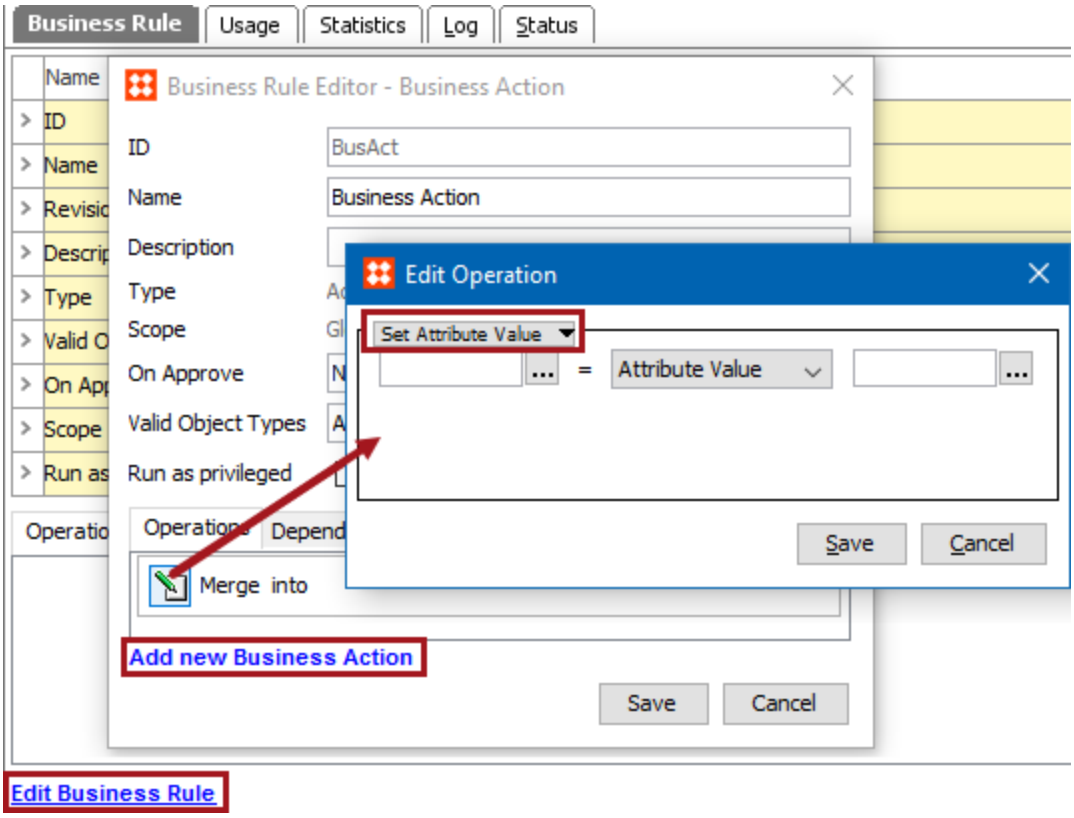
For example, a product being published to a downstream system can first be checked to ensure all mandatory attributes are populated. If so, this business action sets the value of the 'Ready for Sale' attribute to 'Yes' indicating that the product can be approved.

Prerequisites

Before using this action:

1. Create a business rule as defined in the **Creating a Business Rule or Library** topic.

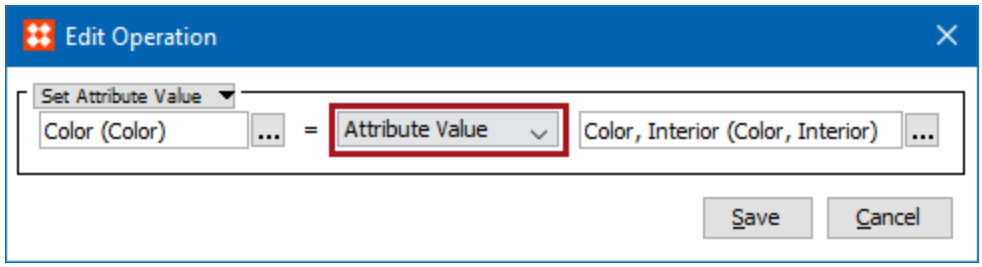
Configuration



1. On the Edit Operation dialog, select the **Set Attribute Value** option from the dropdown.
2. Select the target attribute by typing into the field on the left to use the auto-complete search functionality, or click the ellipsis button **(…)** to display the Select Attribute dialog. Use search or browse to select the attribute, then click the **Select** button. The value on this attribute will be changed to be equal to the attribute value that exists in the source.
3. In the center dropdown parameter, make a selection to display additional required parameters.

- When **Attribute Value** is selected, the select source attribute parameter is displayed. The attribute must be externally maintained and a description attribute.

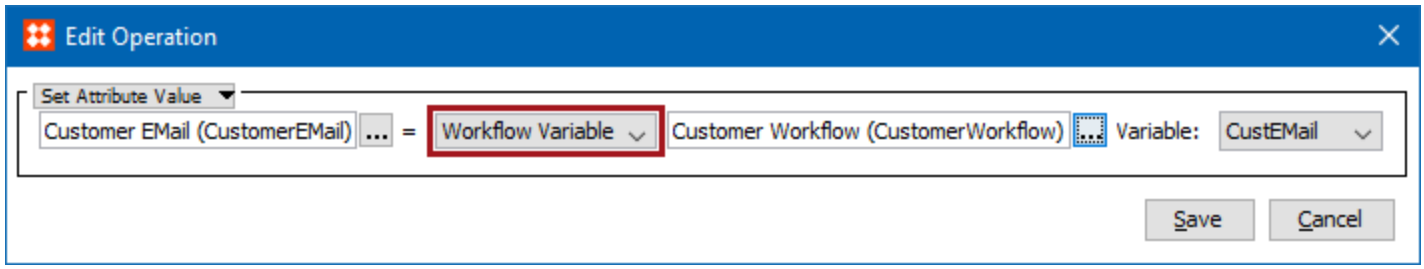
  Select the source attribute by typing into the field on the right to use the auto-complete search functionality, or click the ellipsis button **(…)** to display the Select Attribute dialog. Use search or browse to select the attribute, then click the **Select** button.



- When **Workflow Variable** is selected, the select source workflow and select source variable parameters are displayed. For more information on workflow variables, see the **Workflow Variables** topic in the **Workflows** documentation.

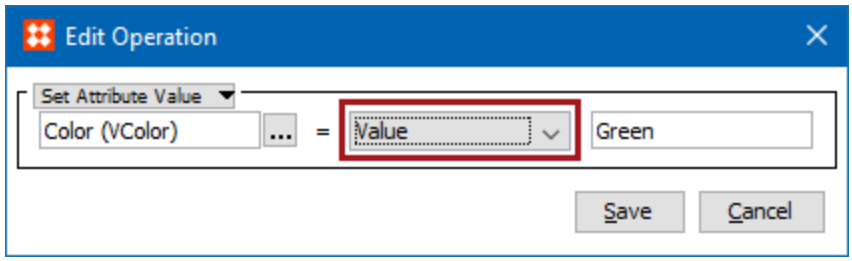
  Select the source workflow by typing into the field on the right to use the auto-complete search functionality, or click the ellipsis button **(…)** to display the Select Workflow dialog. Use search or browse to select the workflow, then click the **Select** button.

  Use the Variable dropdown to select the variable that holds the value used to update the target attribute value.



- When **Value** is selected, a source value text box is displayed.

  Set the source value by typing into the field on the right. This text will be used to update the target value.



4. Click the **Save** button to add the operation to the business rule editor.

Business Action: Set Name

This action can set the name of the current object to a constant value or to the result of a STEP function.

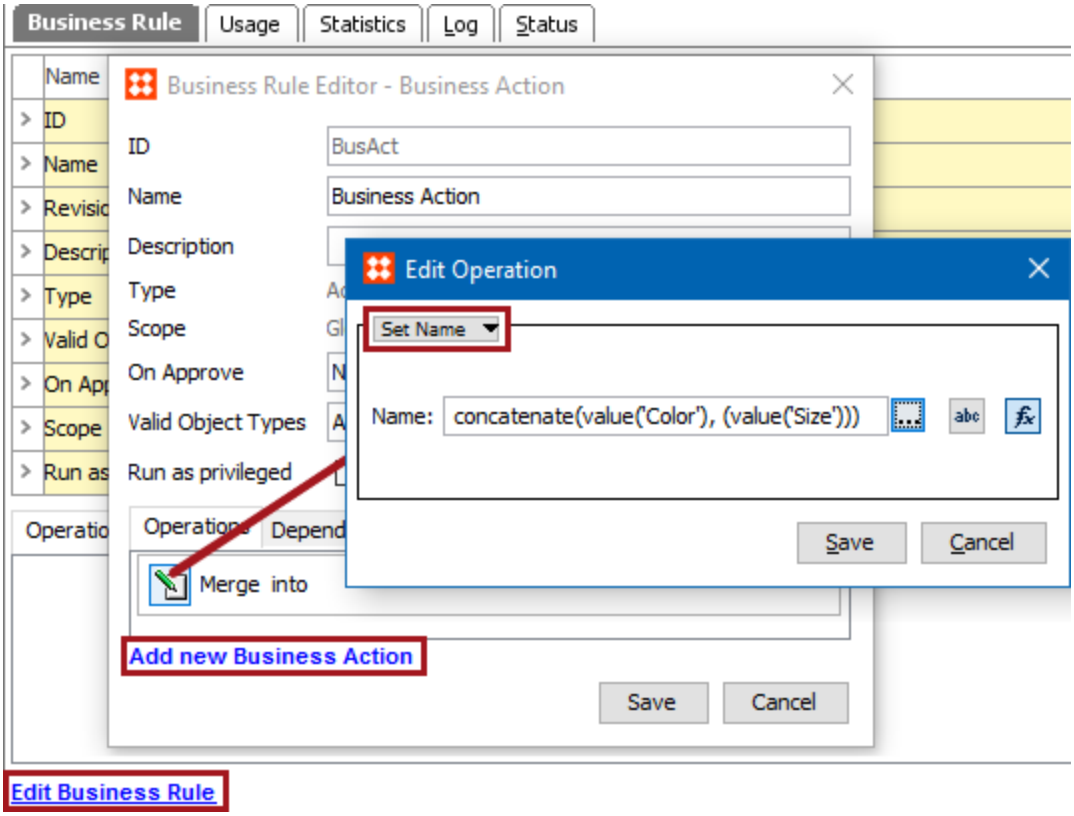
For example, when the color and size attribute values are concatenated to make the name of the object, use this action and a STEP function to generate that name.

Prerequisites

Before using this action:

1. Create a business rule as defined in the **Creating a Business Rule or Library** topic.

Configuration



1. On the Edit Operation dialog, select the **Set Name** option from the dropdown.
2. In the **Name** parameter, use one of these methods to set the relevant name.
   - Click the 'abc' button (), type in a constant value that will be used to update the name.
   - Click the 'fx' button (), click the ellipsis button **(...)** to display the Function Editor dialog, write a calculation to select a valid target, and click the **Select** button. For more information, see the **Using Function Editor** topic in the **System Setup / Super User Guide** documentation.

3. Click the **Save** button to add the operation to the business rule editor.

Business Action: Set Object Type

This action can change the object type of the current object based on a selection or the result of a STEP function.

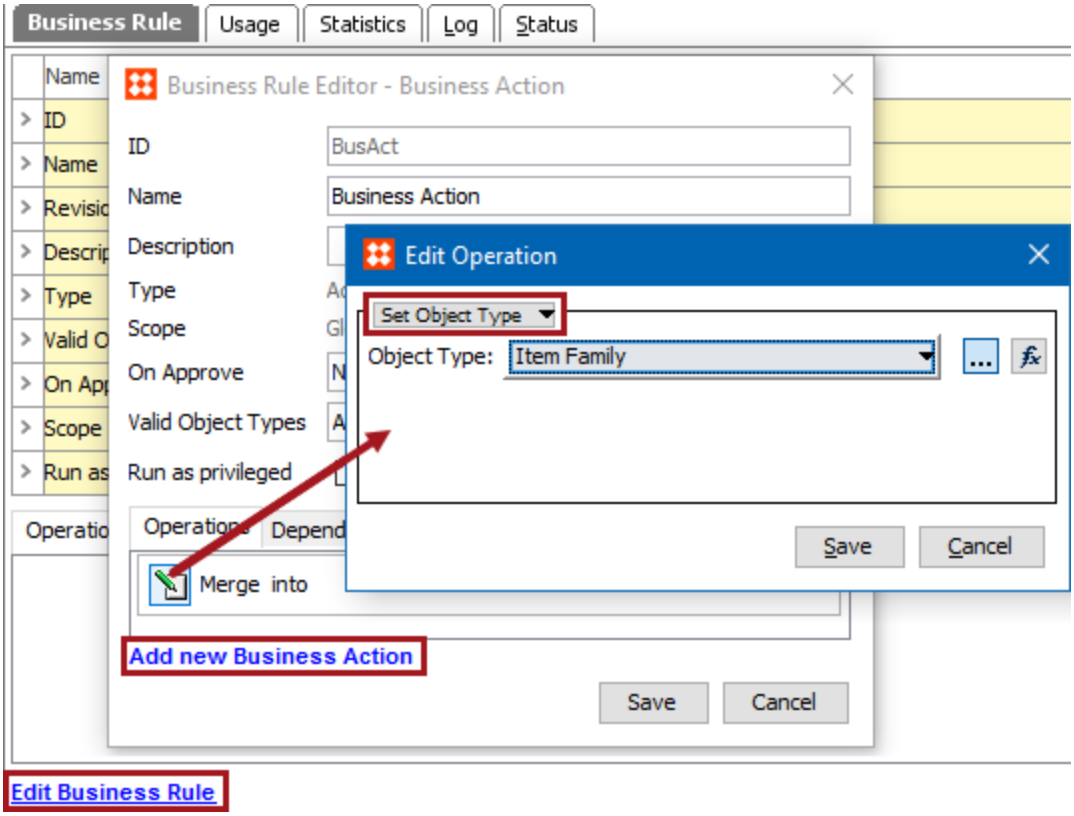
For example, when a group of objects should be updated to a different object type, use this action to choose the new object type and run it via a business rule.

Prerequisites

Before using this action:

1. Ensure the objects being modified are valid for the new object type.
2. Create a business rule as defined in the **Creating a Business Rule or Library** topic.

Configuration



1. On the Edit Operation dialog, select the **Set Object Type** option from the dropdown.
2. In the **Object Type** parameter, use one of these methods to set the relevant name.
   - Click the ellipsis button **(…)** to display the available object types in your system in the dropdown. Choose an item from the dropdown, or when available, choose the **Other...** option to display the Choose Object Type dialog.

- Click the 'fx' button (), click the ellipsis button **(…)** to display the Function Editor dialog, write a calculation to select an object type, and click the **Select** button. For more information, see the **Using Function Editor** topic in the **System Setup / Super User Guide** documentation.

3. Click the **Save** button to add the operation to the business rule editor.

Business Action: Set Product to Classification Link Type

This action can change product-to-classification link types for products that are linked into a given classification, and can assign a new object type and classification link type to the current product. Because a classification object type cannot be the target of more than one product-to-classification link type, the operation changes both the classification object type, and the link type.

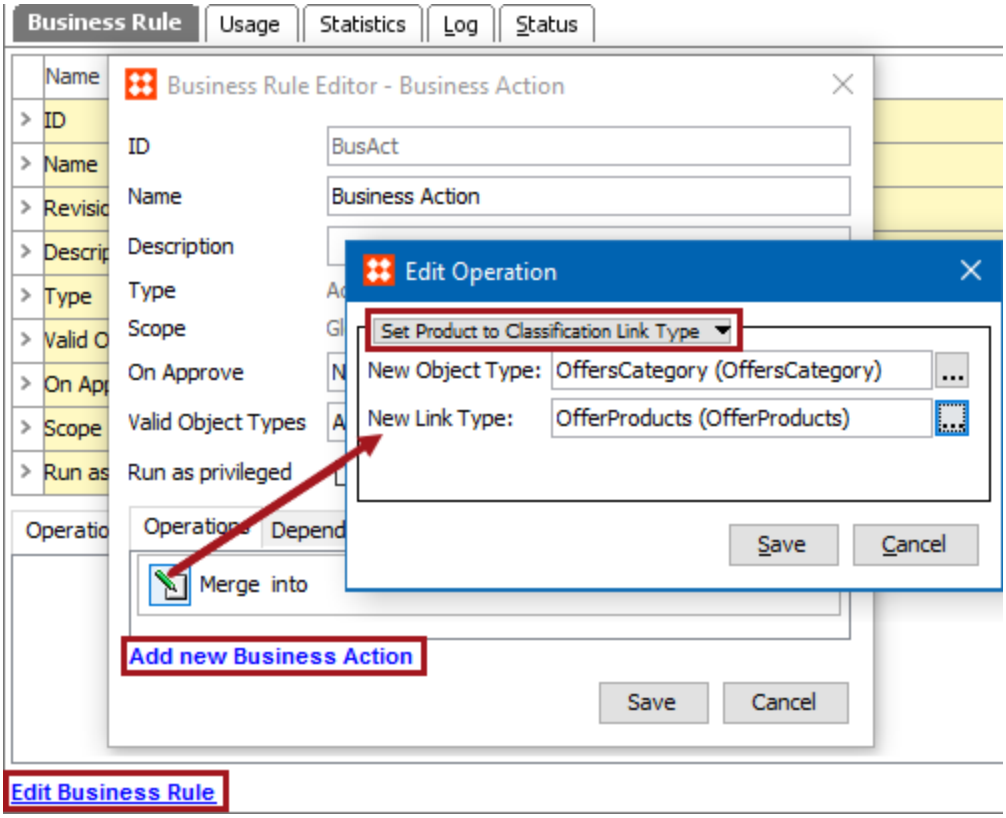
For example, while importing products, select a business rule with this action to automatically link the products to a specific classification. In workflows, when a product enters a particular state, a business rule can link the product to a specific classification reference type. In bulk updates, the same business rule could be run on a collection of products that require the link type.

Prerequisites

Before using this action:

1. Ensure a valid Classification Link Type and the Target Classification Object Type exists. For more information, see the **Reference and Link Types** topic in the **System Setup / Super User Guide** documentation.
2. Ensure a valid classification object type should be linked to the same parent classification of the object type being changed to.
3. Classification Link Type should have the object type valid.
4. Create a business rule as defined in the **Creating a Business Rule or Library** topic.

Configuration



1. On the Edit Operation dialog, select the References and Links > **Set Product to Classification Link Type** option from the dropdown.
2. In the **New Object Type** parameter, click the ellipsis button **(…)** to display the Select Object Type dialog, select the classification, and click the **Select** button.
3. In the **New Link Type** parameter, click the ellipsis button **(…)** to display the Select Product to Classification Link Type dialog, select the link type, and click the **Select** button. The target must comply with the selected type. If the link type does not match the object type selected, an error message displays.
4. Click the **Save** button to add the operation to the business rule editor.

Business Action: Set Workflow Variable

This action can copy a static value, an attribute value, or a workflow variable value, to a workflow variable value.

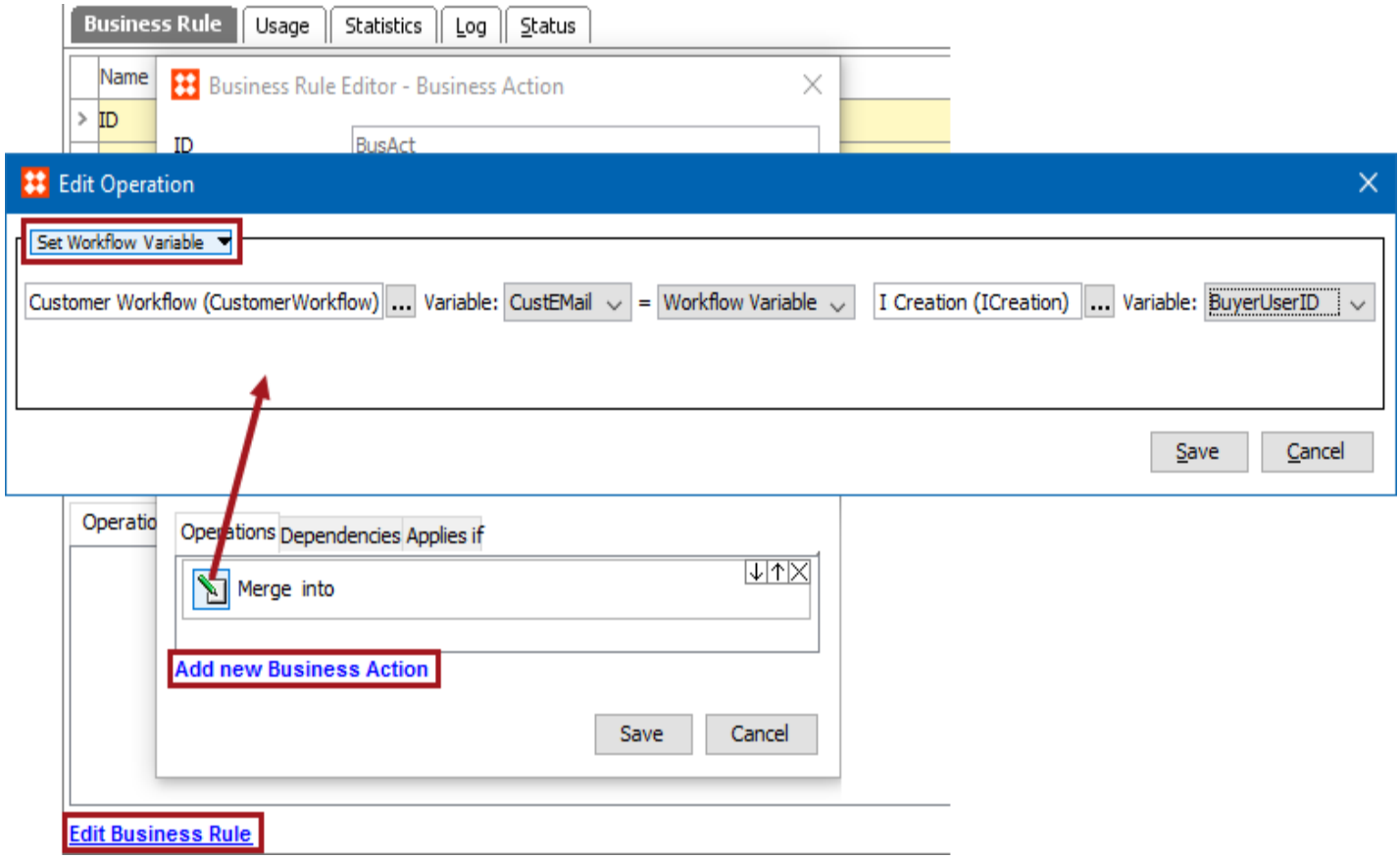
For example, a group of five users have access to a workflow, and can submit a product from the Edit state to the Review state. There requirement is to identify which of these users submitted a particular product. This action is added to a business rule to save the details to an attribute. The business action is set on the desired state in the State Editor on the 'On Exit' tab.

Prerequisites

Before using this action:

1. Ensure at least one variable exists on the workflow. For more information on workflow variables, see the **Workflow Variables** topic in the **Workflows** documentation.
2. Create a business rule as defined in the **Creating a Business Rule or Library** topic.

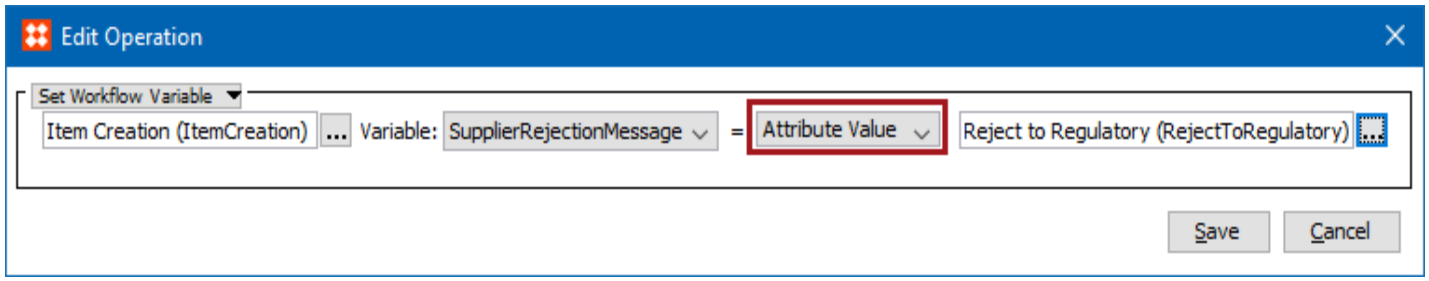
Configuration



1. On the Edit Operation dialog, select the **Set Workflow Variable** option from the dropdown.
2. Select the target workflow by typing into the field on the left to use the auto-complete search functionality, or click the ellipsis button **(…)** to display the Select Workflow dialog. Use search or browse to select the workflow, then click the **Select** button. The selected workflow must have variables.
3. Select the target workflow variable from the Variable dropdown. The selected variable for this workflow will be changed to be equal to the value that exists in the source.
4. In the center dropdown parameter, make a selection to display additional required parameters.

   - When **Attribute Value** is selected, the source attribute parameter is displayed. The attribute must be externally maintained and a description attribute. Use this option when an attribute has a different value in different states of the workflow. This allows you to store the values inside the workflow for access when the workflow reaches the final state.

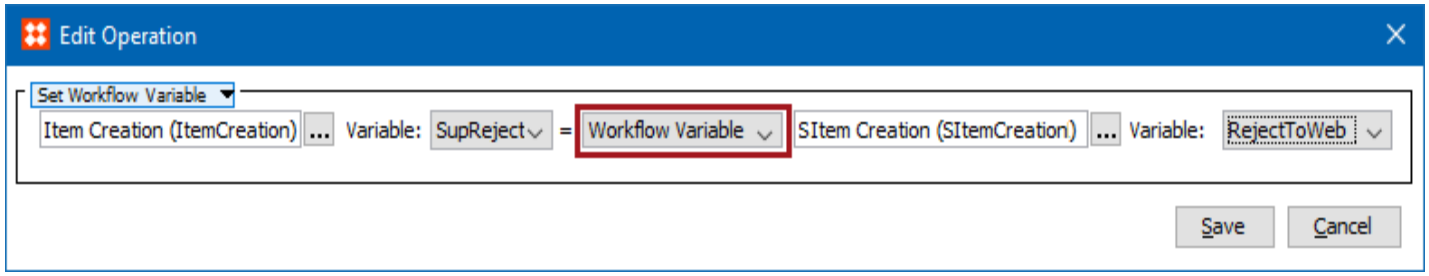
     Select the source attribute by typing into the field on the right to use the auto-complete search functionality, or click the ellipsis button **(…)** to display the Select Attribute dialog. Use search or browse to select the attribute, then click the **Select** button.



   - When **Workflow Variable** is selected, the source select workflow and source select variable parameters are displayed. Use this option when two workflows are dependent on each other, and one workflow needs values which are stored inside the other workflow.

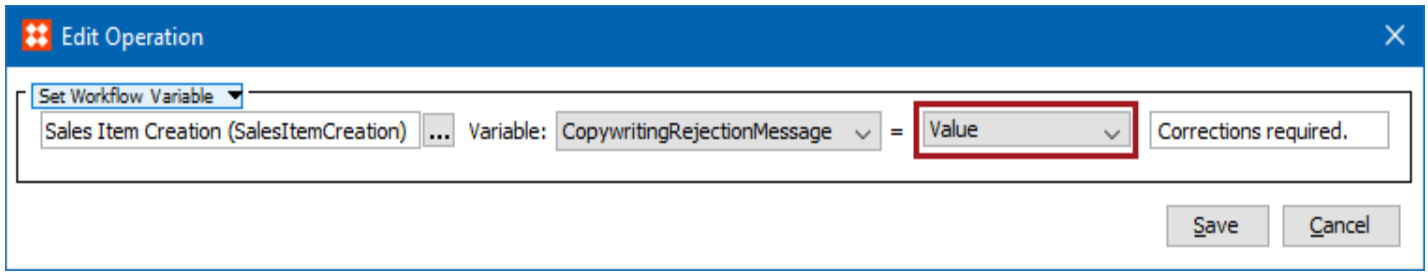
     Select the source workflow by typing into the field on the right to use the auto-complete search functionality, or click the ellipsis button **(…)** to display the Select Workflow dialog. Use search or browse to select the workflow, then click the **Select** button.

     Use the Variable dropdown to select the variable that holds the value used to update the target attribute value.



- When **Value** is selected, a source value text box is displayed. Use this option to store values which can only be used inside the workflow.

  Set the source value by typing into the field on the right. This text will be used to update the target value.



5. Click the **Save** button to add the operation to the business rule editor.

Business Action: Standardize Address

This action allows for different standardization actions to be taken on addresses by using the Loqate solution. If applicable, CASS (Coding Accuracy Support System) operations can also be performed to apply a stricter level of address validation to US-based addresses.

Standardize Address business action operations are valid for execution on entities with flat attributes, e.g. Address object types and address Data Containers (Data Containers with address attributes).

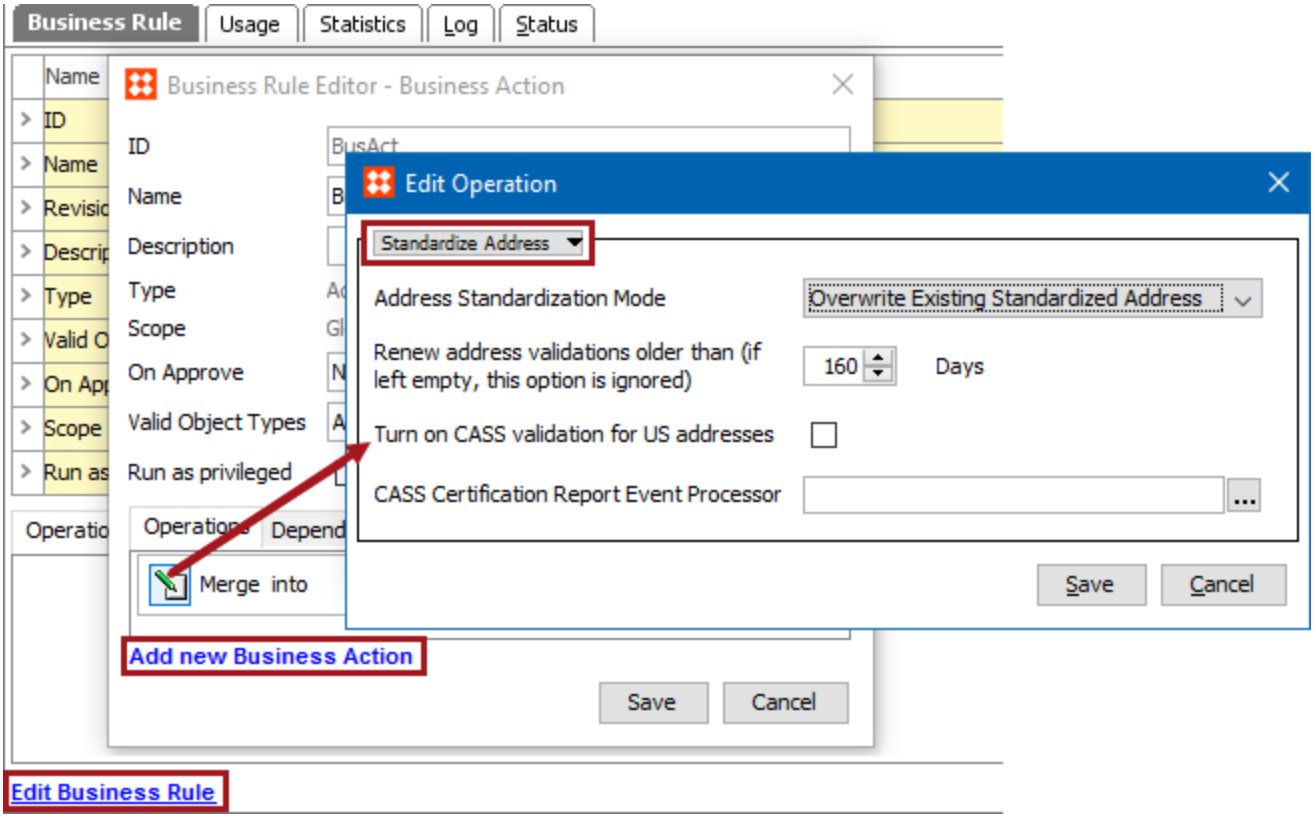
An example scenario where a standardize address business action would be used is in an event processor that is configured to listen for changes in address input attributes on specified entities and/or data containers. The processor will be configured to use the 'Business Action Event Processor' processor. When the event processor is invoked, the Standardize Address business action will execute and perform an address standardization operation on the objects where the changes were detected.

Prerequisites

Before a Standardize Address business action can be used, the following conditions must be met:

- Users must be connected to a Loqate server, either through a cloud API or local API
- The Address Component Model must already be configured. It is strongly recommended to configure this component model using the 'Easy setup of Address Component Model' wizard. For more information, see the **Address Component Model** section of the **Loqate** documentation.
- To view and use the CASS features, users must:
    - Purchase a CASS license
    - Be connected to Loqate through a *local* API (CASS components will not work with a Loqate Cloud API)
    - Be based in the US, as CASS is not valid outside of the US
    - Configure the CASS Address Component Model. For more information, see the **CASS Address Component Model** topic in the **Loqate** documentation.
- Create a business rule as defined in the **Creating a Business Rule or Library** topic.

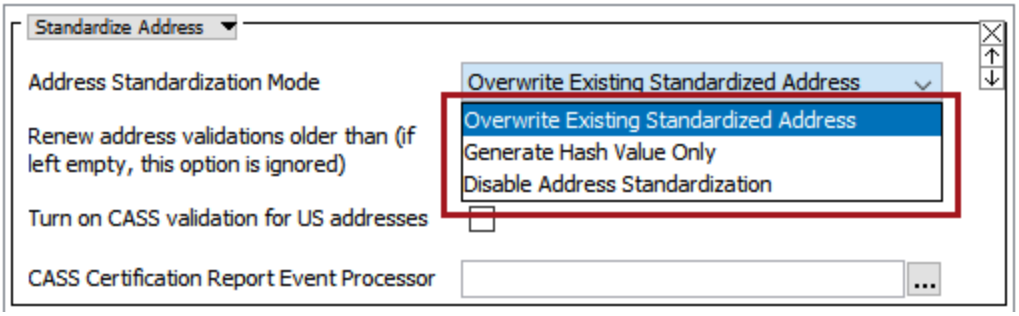
Configuration



This documentation topic describes the options available when configuring a Data Quality > Standardize Address business action. For more detailed information on the overall Loqate and CASS solutions, see the **Loqate Integration** section of the **Data Integration** documentation.

Address Standardization Modes

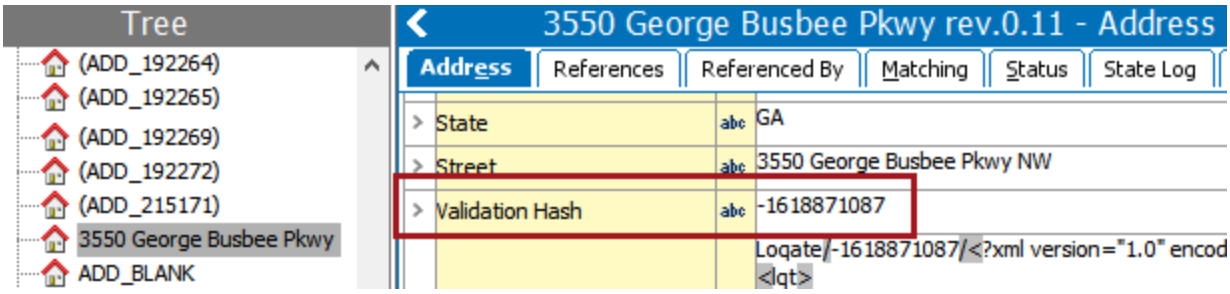
Three options are available from the Address Standardization Mode dropdown list.



- **Overwrite Existing Standardized Address:** Overwrites an existing standardized address by making a call to Loqate, which returns the latest standardized address and overwrites the existing one.

- **Generate Hash Value Only:** When this option is chosen, the business action will recalculate the 'Validation Hash' value without making a call to Loqate. The hash value is based on the address input fields only and is calculated and maintained by STEP.

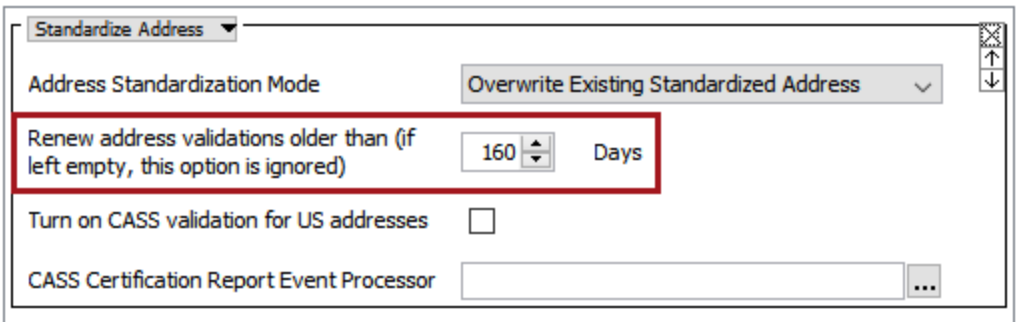
  This option is used in cases when pre-standardized addresses, which are known to be correct, are imported. Because the addresses are new, no hash value is present. When the 'Generate Hash Value only' action is run, STEP generates a hash value and places it into the 'Validation Hash' field. The hash value lets STEP know that the addresses are new, ensuring that they are not picked up and restandardized by STEP shortly after they are imported. For users of the Loqate Cloud API solution, this saves money by avoiding unnecessary calls to Loqate, since users are charged a fee for every call made to Loqate.



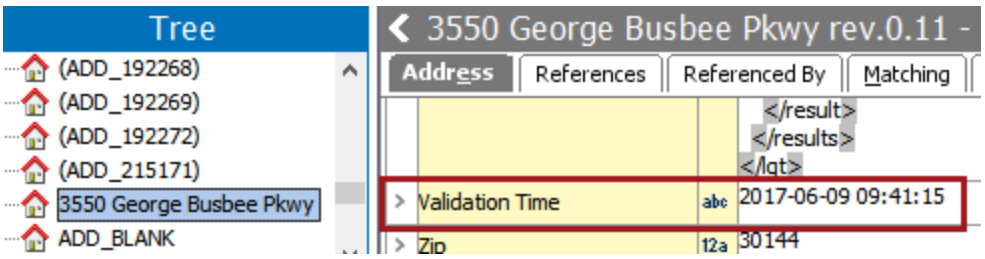
- **Disable Address Standardization:** This option is primarily useful for users who import addresses in bulk, either manually or by using an inbound integration. If a business rule is in place that runs a Standardize Address bulk update on import, then selecting 'Disable Address Standardization' is a simple way to temporarily turn off automatic address validation on import. This allows users to validate the addresses later.

Renew Address Validations

The checkbox option **Renew address validations older than** allows for addresses older than a designated number of days to be revalidated. This is useful in case an address was validated an extended period of time ago and the address may no longer be valid. The default number of days is 160. The field is disabled by entering the value '0.'



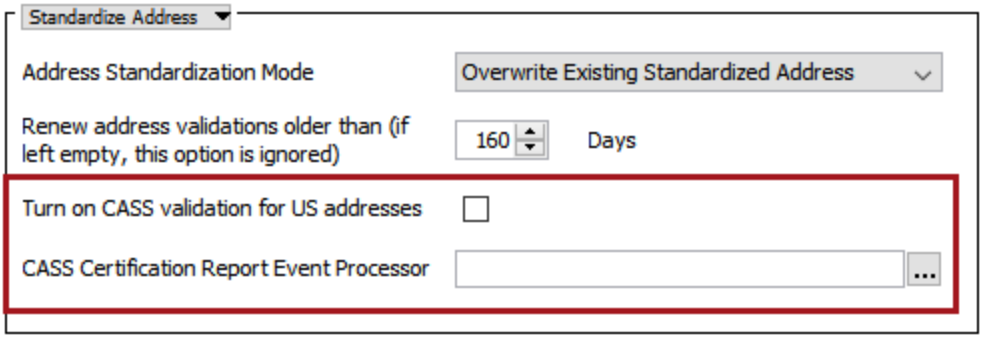
STEP determines the age of an address by the value present in the 'Validation Time' field.



Important: For users of the Loqate Cloud installation, care should be exercised when using this functionality. Since all addresses older than the specified time period will be revalidated, each additional validation will make a chargeable call to Loqate.

CASS Validation

Two CASS address validation options are available on the Standardize Address dialog.



- **Turn on CASS validation for US addresses:** If checked, all addresses are sent for standardization to the Loqate Local server integration and also validated against CASS data.
- **CASS Certification Report Event Processor:** If you would also like a CASS certification report generated after using CASS validation on an address, click the ellipsis button **(...)** and select the relevant CASS certification report event processor from the 'Select Event Processor' dialog. For more information, see the **CASS Certification Report Processing Plugin Parameters and Triggers** section of the **Event Processors** documentation.

Business Action: Trigger STEP Workflow Event

This action can trigger the creation of a specified workflow event on a node when the business action is executed.

For example, when a user completes a task and needs to move the object from one state to another outside of the workflow, running a business rule with this action can generate the event.

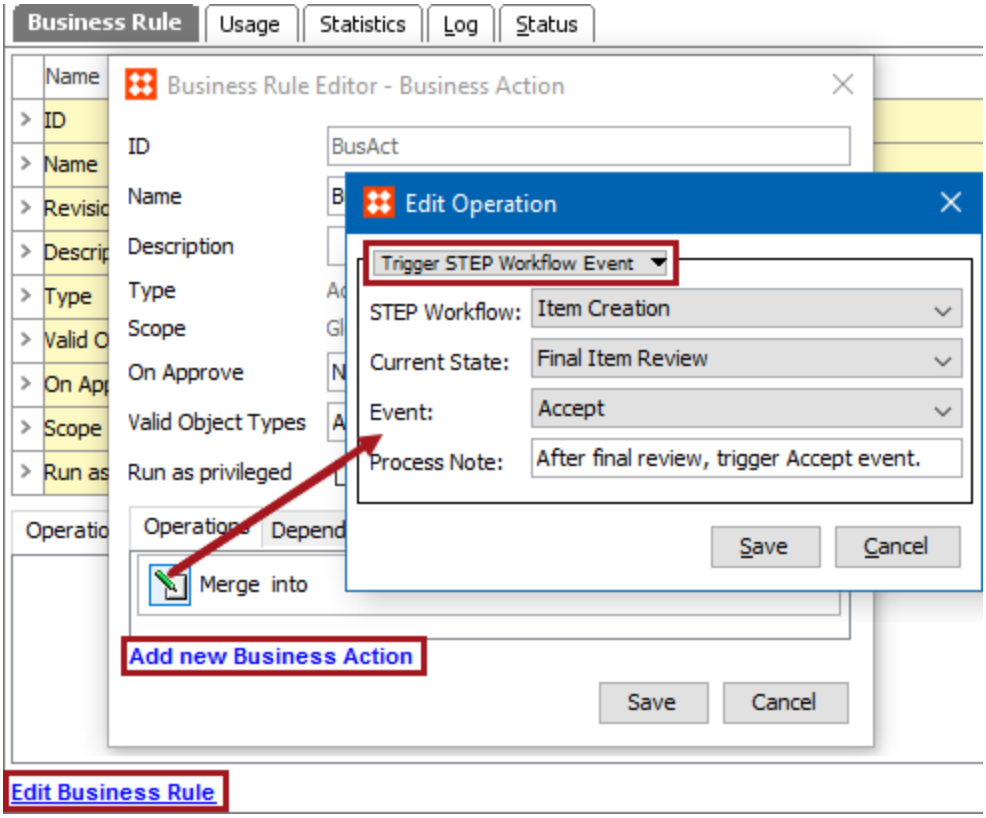
For more information, see the **Business Rules and Workflows** topic in the **Workflows** documentation.

Prerequisites

Before using this action:

1. Ensure the workflow exists including states and events.
2. Ensure the node is present in the state defined in the action.
3. Create a business rule as defined in the **Creating a Business Rule or Library** topic.

Configuration



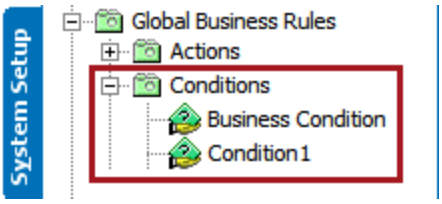
1. On the Edit Operation dialog, select the Workflow > **Trigger STEP Workflow Event** option from the dropdown.

2. In the **STEP Workflow** parameter, from the dropdown select the relevant workflow.
3. In the **Current State** parameter, from the dropdown select the state that the workflow object must be in when the action is applied.
4. In the **Event** parameter, from the dropdown select the event that the action triggers.
5. In the **Process Note** parameter, enter a comment to be written in the state log when the event is triggered.

Business Conditions

While complex business conditions can be created using either STEP Functions or JavaScript, simple conditions can be implemented using the options in the table below. Each is described in further detail in individual sections.

To access the available business conditions, under the System Setup tab, open the Global Business Rules node, select or create a business condition and open the Business Condition flipper. Your system can also include a 'Setup Group' root node that includes all Business Rules.



Before deciding to use a business condition, review the **Alternate Options** section below to determine if the rule could be enforced using standard configuration.

Important: Business conditions are optimized for determining the true / false result of read-only scenarios. Although it is technically possible to write business conditions to change data, doing so can result in unpredictable errors and should therefore not be attempted. To change data effectively, use a business action. For more information on business actions, see **Business Actions**. To generate an output on not changing input parameters, use business functions. For more information on business functions, see **Business Functions**.

Condition	Description
Attribute Value Comparison	Compares the value of one attribute on the object being tested by the business rule with either a constant or the value of another attribute on the same object. For more information, see **Attribute Value Comparison**.
Evaluate JavaScript	Runs defined JavaScript code added on the business condition using the same bindings that are available for the business action 'Execute JavaScript'. **Important:** Although the same binds are available, changing STEP data via an Evaluate JavaScript business condition is not supported. For more information, see **Execute JavaScript**.
Function	Makes it possible to use functions to define business conditions. For more information, see **Business Condition: Function**

Condition	Description
LOV Cross-Validation	Specifies that the value of one List of Value (LOV) attribute limits which values are valid for another LOV attribute. For more information, see **Business Condition: LOV Cross-Validation**.
Reference other Business Condition	Allows the current business condition to reference other business conditions. For more information, see **Business Condition: Reference other Business Condition**.
Valid Hierarchies	Specifies sub-hierarchies where the condition is true. For more information, see **Business Condition: Valid Hierarchies**.
Validate Product Variant	Validates product variants for duplicates. This condition uses the configuration created when setting up Product Variant Families, so there is no additional configuration. For more information, see **Business Condition: Validate Product Variant**.

To add a business condition, see **Specifying a Business Condition**.

Alternate Options

When considering the use of a business condition, it is good practice to evaluate if the same result can be reached with a more efficient function. The following examples illustrate some of these scenarios.

Required value on approval: To determine if an attribute has a value before approval, use the 'mandatory' parameter on the attribute itself, as described in the **Mandatory Attributes** topic of the **System Setup / Super User Guide** documentation.

Restrict value input: The Attribute Validation Base types can be used to globally enforce value lengths, minimum and maximum values, data masks, as described in the **Validation Rules** topic of the **System Setup / Super User Guide** documentation.

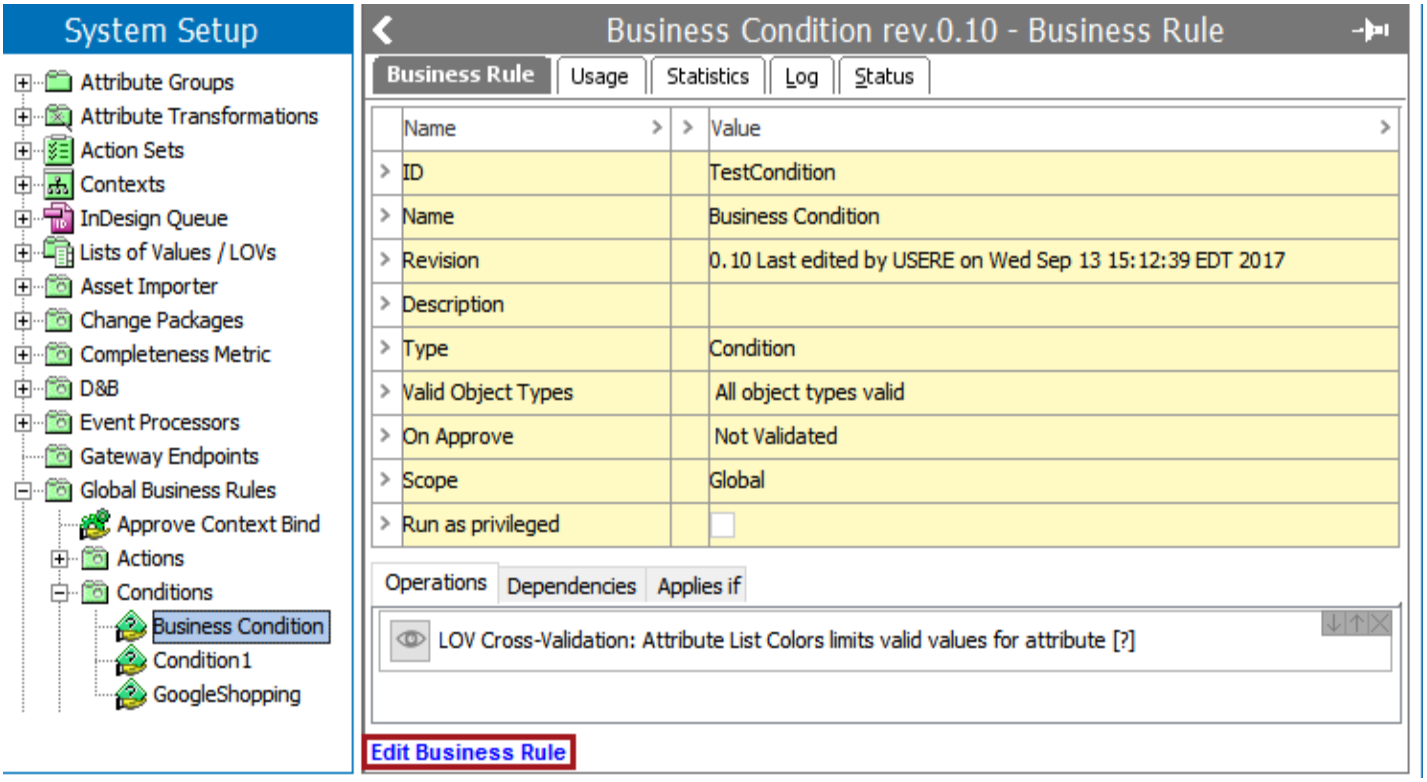
Restrict object type within a hierarchy: The Object Types & Structures configuration provides a way to apply restrictions to the hierarchy so only objects of specific types can exist below other objects of specific types, as described in the **Object Type Hierarchy** topic of the **System Setup / Super User Guide** documentation.

Restrict references and links based on object type: References and Links can be made valid only from objects of specific types to other objects of specific types, as described in the **Reference and Link Types** topic of the **System Setup / Super User Guide** documentation..

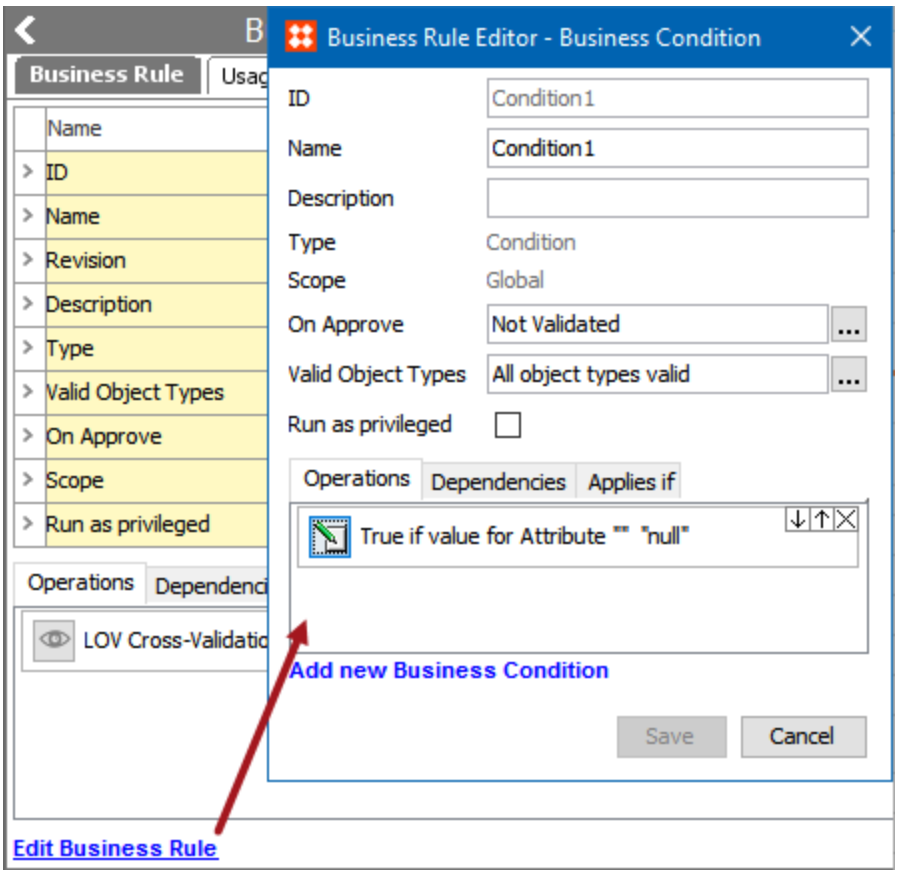
Specifying a Business Condition

Once a business rule exists, use the following steps to add a business condition. For information on creating a new business rule, see the **Creating a Business Rule or Library** topic.

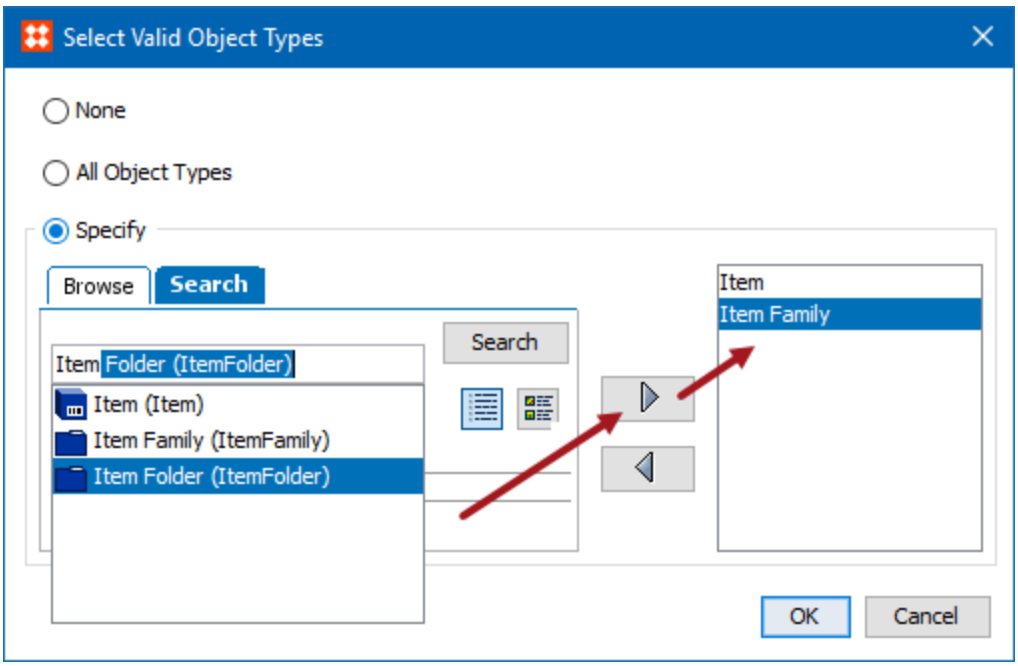
1. In System Setup, expand the **Business Rules** setup group and select an existing business condition to display the business rule editor.



2. On the Business Rule tab, click the **Edit Business Rule** link to open the business rule editor for changes.

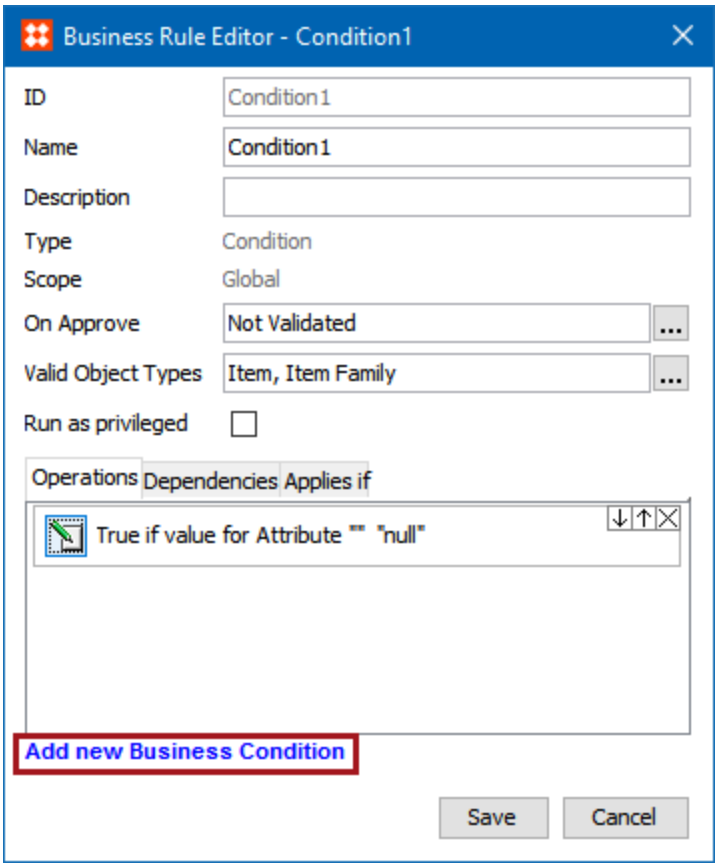


3. In the **Valid Object Type** field, click the ellipsis button **(…)** to display the Select Valid Object Types dialog.
4. Choose a radio button:
   - **None** prevents the business condition from running. This can be used to temporarily disable a condition.
   - **All Object Types** means the business condition will run regardless of the object type. It is not recommended to use this option.
   - **Specify** allows you to limit the object types that will cause the business condition to run. Selecting Specify displays the Browse and Search tabs. Identify the desired object types and use the right arrow button to move the item to the right panel. Use the left arrow button to remove an object type from selection.

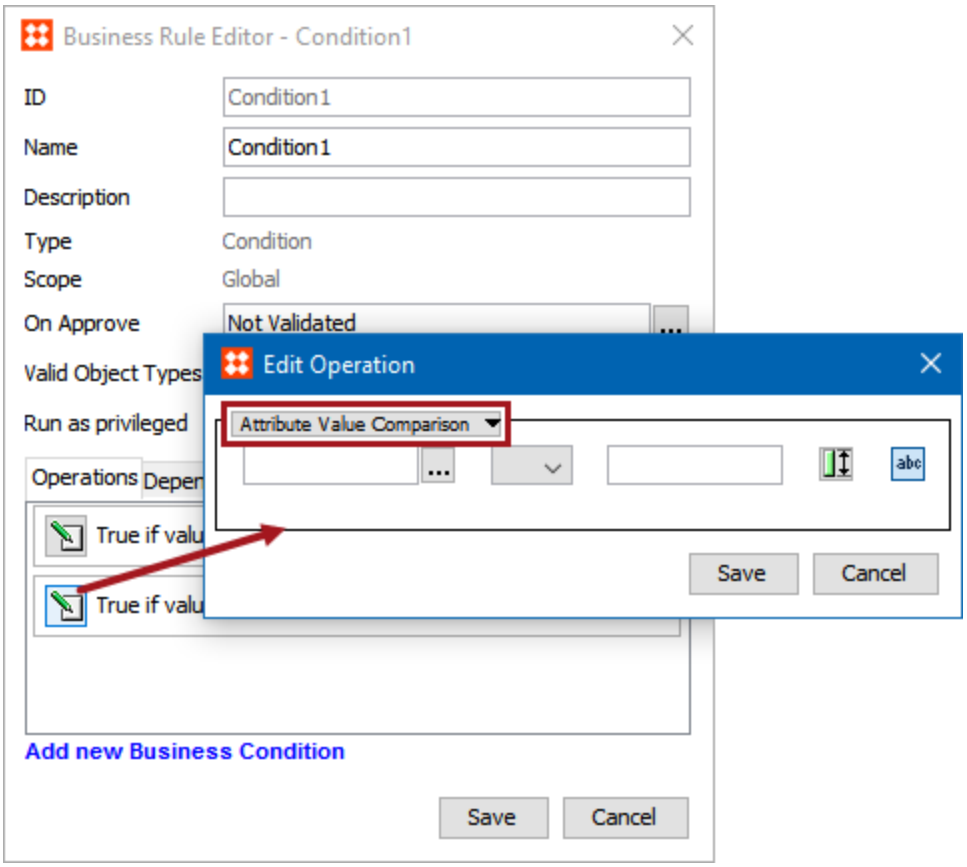


After selecting object types, click the **OK** button to save the changes.

5. In the lower left corner, click the **Add New Business Condition** link to add an additional operation.



6. Click the **Edit Operation** icon for the newly added business condition.



7. In the Edit Operation dialog, use the dropdown list to select the preferred condition. The conditions are described in the **Business Conditions** topic.
8. Click **Save** to save the changes.
9. Add additional business conditions as needed.

Important: When more than one condition is specified, all conditions are carried out when the overall business condition is executed.

Business Condition: Attribute Value Comparison

The 'Attribute Value Comparison' operation allows users to compare an attribute value on a selected object in relation to whatever the business condition is testing, be it a constant value or another attribute value on the same object. Alphanumeric comparison is used in most cases, except for dates (date, ISO date, and ISO date and time) and numbers.

For example, consider an attribute named ‘Active’ on products which has value ‘Yes’ and ‘No.' If the value is ‘Yes,’ then they are active products, and they should be approved. If value is ‘No,’ then consider the objects as non-active products and do not approve them.

Another example would be classifying products based on the product attribute ‘Color.' If the value of the ‘Color’ attribute is ‘Red,’ then classify the products into ‘Red’ classifications. However, if the value of the ‘Color’ attribute is ‘Blue,’ then classify the products into ‘Blue’ classification.

Special Comparisons

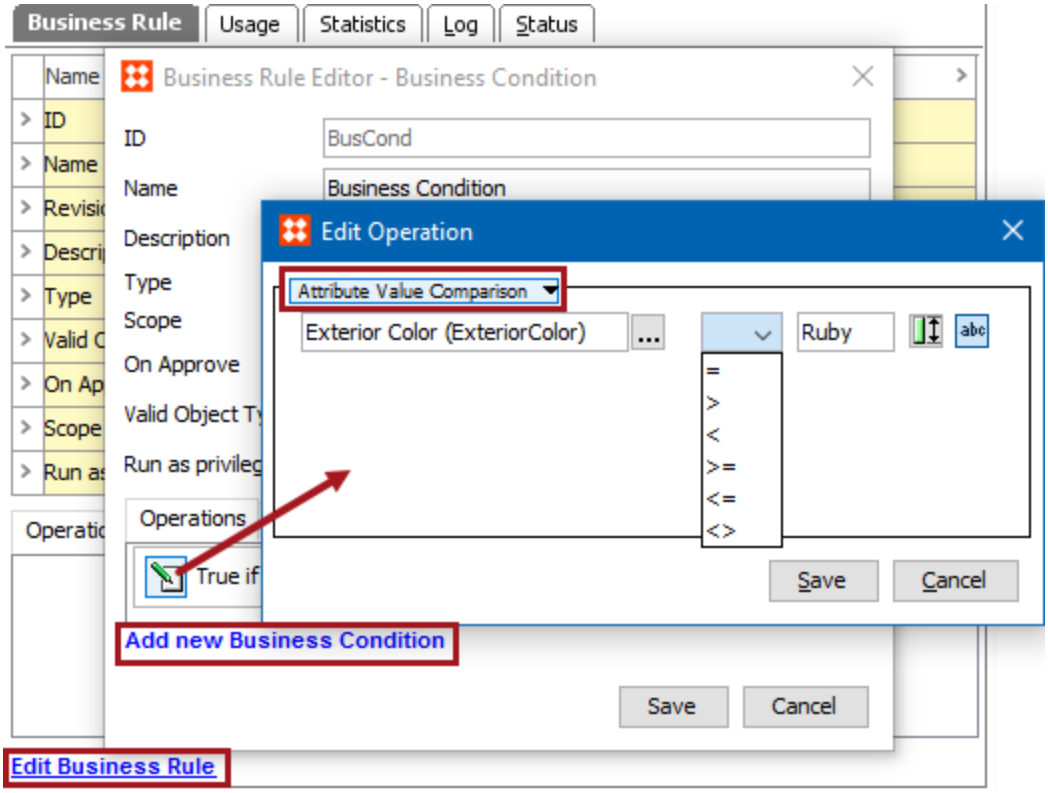
- Numeric comparison is used when both values are number based (fractions, integers, or numbers).
- When numerical values are compared with units, the system attempts to convert the values to base units.
- Multivalued attributes and attributes of the validation base type **Embedded Number** are not supported.

Prerequisites

Before using this business condition:

1. Create a business rule as defined in the **Creating a Business Rule or Library** topic.
2. Edit the business rule to configure the operation as defined in **Editing a Business Rule** topic.

Configuration



1. On the Edit Operation dialog, select **Attribute Value Comparison** from the dropdown.
2. Click the ellipsis button **(…)** next to the first parameter to display the Attribute 1 dialog. Use Browse or Search and select the attribute to compared. Click the **Select** button to close the Attribute 1 dialog.
3. In the dropdown parameter, select the condition / operator to be used to compare attributes value. Values can be equal to, greater than, less than, or any of the combinations.
4. Select the type of compare to be performed and the value:
   - Click the constant button (abc) to compare the attribute value with a constant value. Enter a constant value in the text box. 'Ruby' is shown in the image above.
   - Click the attriute button ( ) to compare the attribute value with a different attribute value. Click the ellipsis button **(…)** next to the text box to display the Attribute 2 dialog. Use Browse or Search and select the attribute to be compared against. Click the **Select** button to close the Attribute 2 dialog.
5. Click the **Save** button to add the operation to the business rule editor.

Business Condition: Evaluate JavaScript

In addition to the standard business rule operations, more complex functions can be carried out using JavaScript and the Scripting API. This condition enables you to use the same STEP Public Java API methods that are available in the Execute JavaScript business action operation.

Conditions contain tests, and the JavaScript should only evaluate to true or false.

- The Boolean value 'true' must be returned for the condition to evaluate to **true**.
- Any returned value other than the Boolean value 'true' makes the condition evaluate to **false**. A false condition can display a user-facing message by either returning a string or a message object (localized).

Important: Business conditions are optimized for determining the true / false result of read-only scenarios. Although it is technically possible to write business conditions to change data, doing so can result in unpredictable errors and should therefore not be attempted. To change data effectively, use a business action and the Execute JavaScript operation. For more information, see the **Execute JavaScript** section of the **Business Rules** documentation.

Using JavaScript in business rules can include:

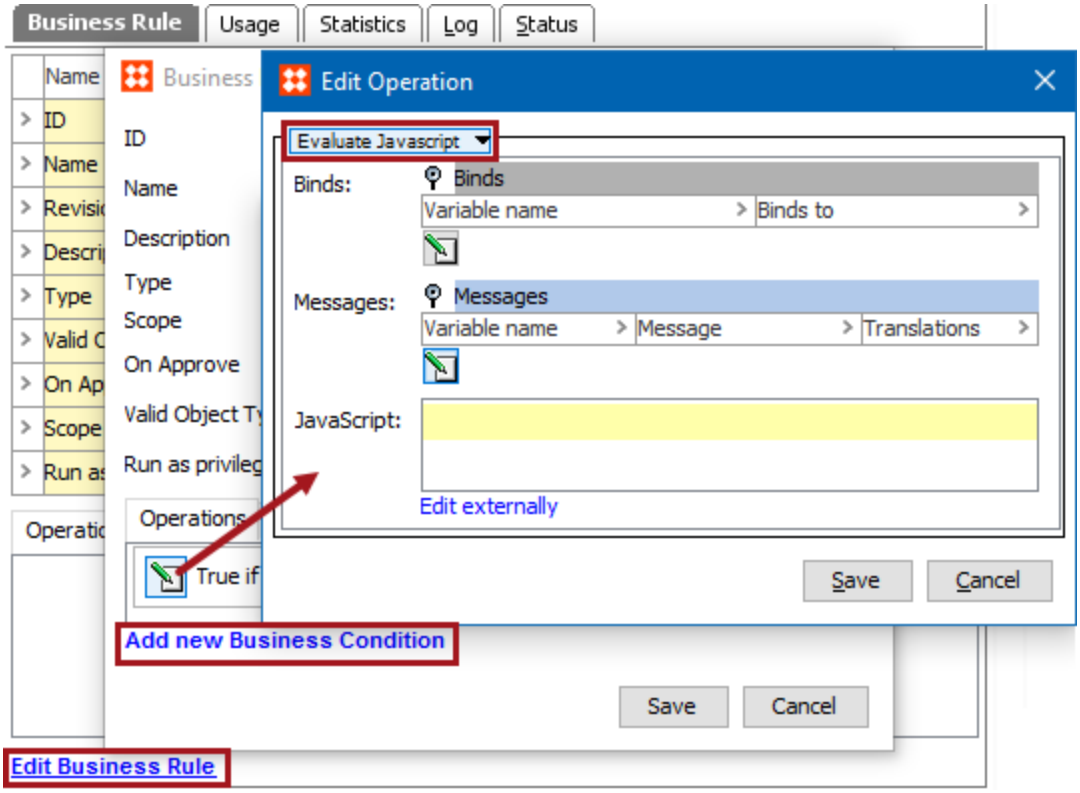
- **Evaluate JavaScript** operation is used for conditions. For more information about business rule conditions, see **Business Conditions**.
- **JavaScript Binds** for both action. condition. and function operations. Binds provide access to the STEP data being worked on. For more information, see **JavaScript Binds**.
- **Business Libraries** is a set of JavaScript functions that can be reused in multiple business rules. For more information, see **Business Libraries**.
- **Scripting API** documentation is available by clicking the STEP API Documentation button on the STEP WebStart page.

Prerequisites

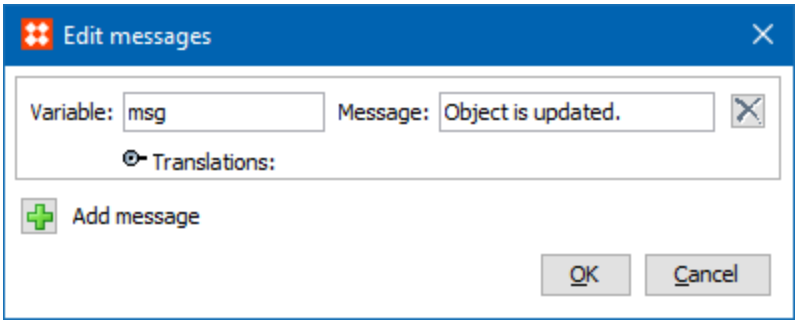
Before using this business condition:

1. Understand the implications of using JavaScript as defined in the **Java vs. JavaScript** topic.
2. Understand the considerations that should be taken as defined in the **JavaScript Considerations** topic.
3. Create a business rule as defined in the **Creating a Business Rule or Library** topic.
4. Edit the business rule to configure the operation as defined in **Editing a Business Rule** topic.

Configuration



1. On the Edit Operation dialog, select **Evaluate JavaScript** from the dropdown.
2. For the **Binds** parameter, create binds needed for the JavaScript code as described in the **Adding a Bind** topic.
3. For the **Messages** parameter, create messages needed for the JavaScript code as follows:
   - Click the **Edit** button ( ) to display the Edit Messages dialog.



   - Click the **Add message** button ( ) to create a new message entry.
   - Click the **Remove** button ( ) to delete a message entry.
   - For the **Variables** text box, type a label for the variable.

- For the **Message** text box, type the message.
- When necessary, open the Translations flipper and provide the data required. A localized message option is available in both JavaScript business actions and conditions. For more information, see the **Localized Messages for JavaScript Business Rules** documentation.
- Click **OK** to save the messages to the operation.

4. For the **JavaScript** parameter, add the JavaScript code.
5. Click the **Save** button to add the operation to the business rule editor.

Business Condition: Function

Conditions can be formulated using STEP Functions, the functional language also used for Calculated Attributes and elsewhere in STEP. You can add function templates, and you can also insert an attribute ID by navigating to the relevant attribute.

The basic functionality is that a condition is true if the function evaluates to 1, and false if it evaluates to 0. Explicitly returning 1 or 0 is not required since a range of functions do this automatically.

For example, the following STEP Functions are available for use in business rules:

- Binary logical functions <, >, <=, >=, =, != used for numerical comparisons
- exact(string1, string2)
- and(condition1, condition2, …)
- or(condition1, condition2, …)
- not(condition)
- listcontains(list, text)

Conditions based on STEP Functions do not allow for dynamic user facing messages. For this type of condition, it is only possible to display an error message that is written in the error message text field, which cannot include any variables.

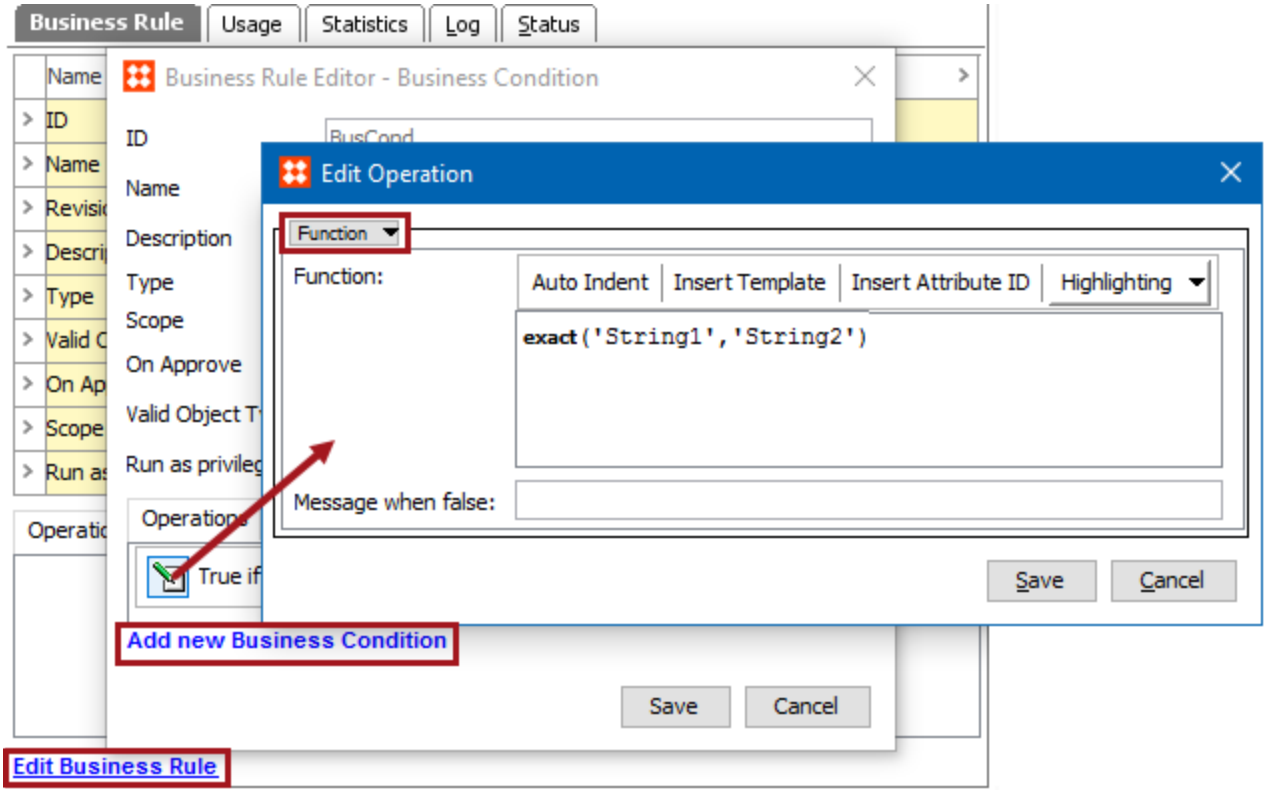
For more information, see the **STEP Functions** documentation.

Prerequisites

Before using this business condition:

1. Create a business rule as defined in the **Creating a Business Rule or Library** topic.
2. Edit the business rule to configure the operation as defined in **Editing a Business Rule** topic.

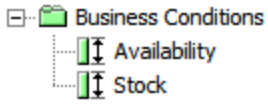
Configuration



1. On the Edit Operation dialog, select **Function** from the dropdown.
2. For the **Function** parameter, add the function in the same way as you would in the Function Editor. For more information, see the **Function Editor** topic or **Using Function Editor** in the **System Setup / Super User Guide** documentation.
3. For the **Message when false** parameter, specify the message to display when appropriate.
   - If a function returns `1`, the condition is **true**.
   - If a function returns `0` or anything else but `1`, the condition is **false**.
3. Click **Save** to add this operation to the business condition.

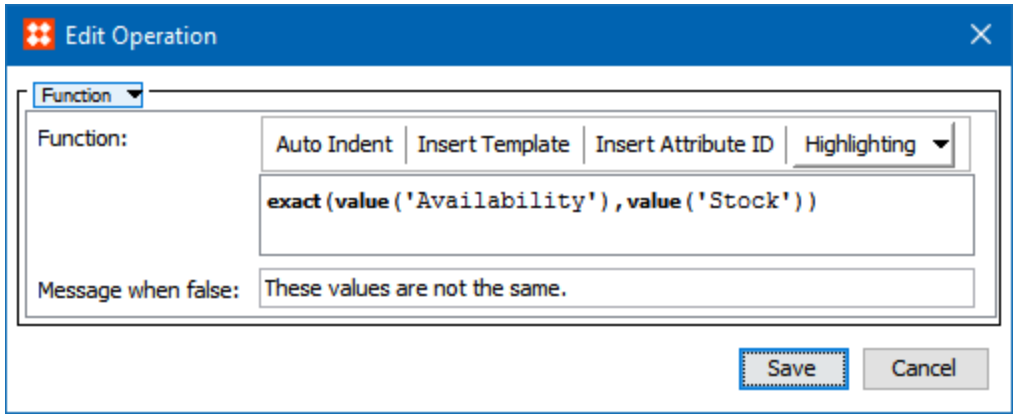
Function Example

In this case, the user would like to compare both the values of two attributes: 'Availability' and 'Stock.' These attributes will be located in a 'Business Condition' attribute group for the sake of clarity.

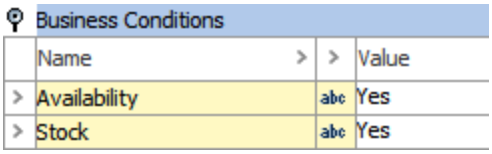


Create the business condition using the steps earlier in this topic. To compare two attribute value strings, use the exact and value functions. Click the 'Insert Template' tab and select the 'Simple String Comparison.' This adds the

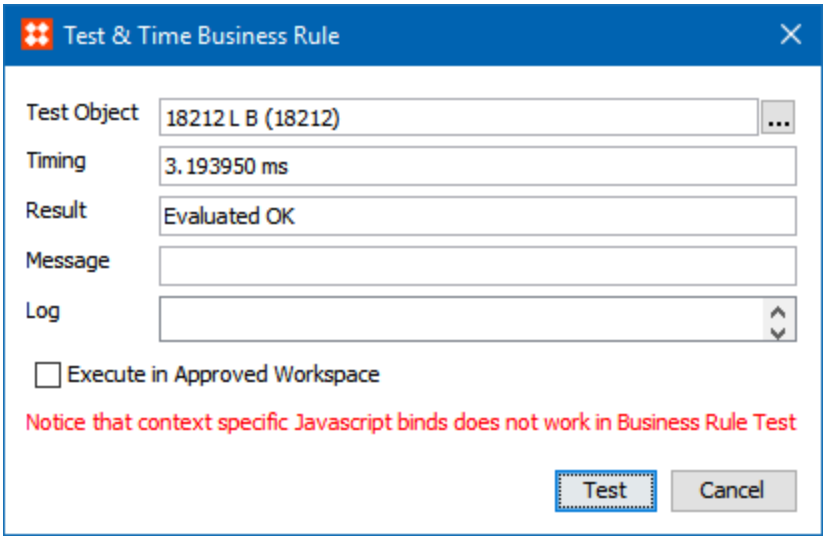
exact() template to the function. Next, use the Insert Attribute ID option to add the two attributes using value (Availability) and value(Stock). Add a message for when the condition is false to provide context for other users. When completed, the function will appear like this:



Now that the business condition is set up, navigate to an object with these attributes and set both values to the same.



Return to the 'Global Business Rules' in the **System Setup** tab. Right-click on the business condition, and select 'Test Business Rule.' On the 'Test & Time Business Rule' dialog, select the test object where the Availability and Stock values are the same. Select 'Test.' The results will show that these attributes are the same.

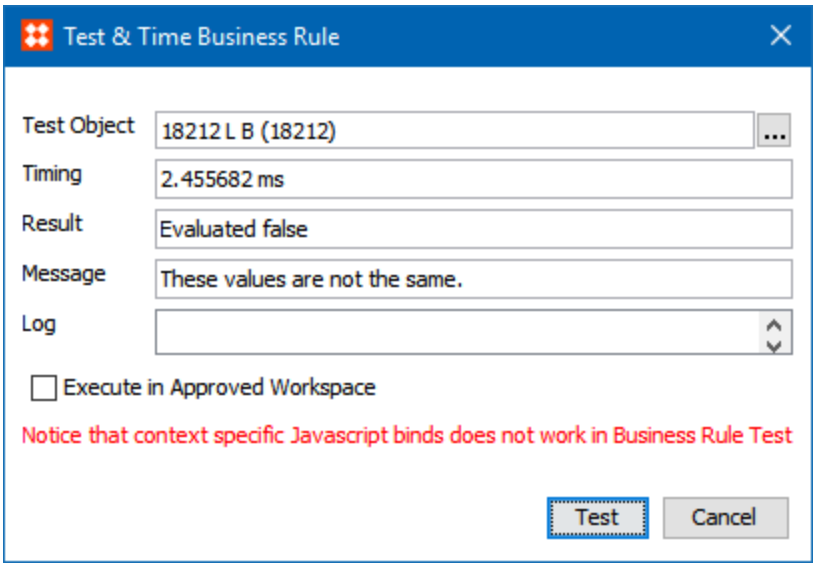


Now, return back to the object and change one of the values. In this example, Stock is set to 'No.'

Business Conditions

Name	Value
Availability	Yes
Stock	No

Return back to the business condition and test it again. The results will show that the two attributes are not the same.



Business Condition: LOV Cross-Validation

LOV Cross-Validation is a condition which specifies legal relationships between the values in two Lists of Values (LOV) attributes. Setting up the 'LOV Cross-Validation' condition requires setting a defining attribute to which the dependent attribute will be compared.

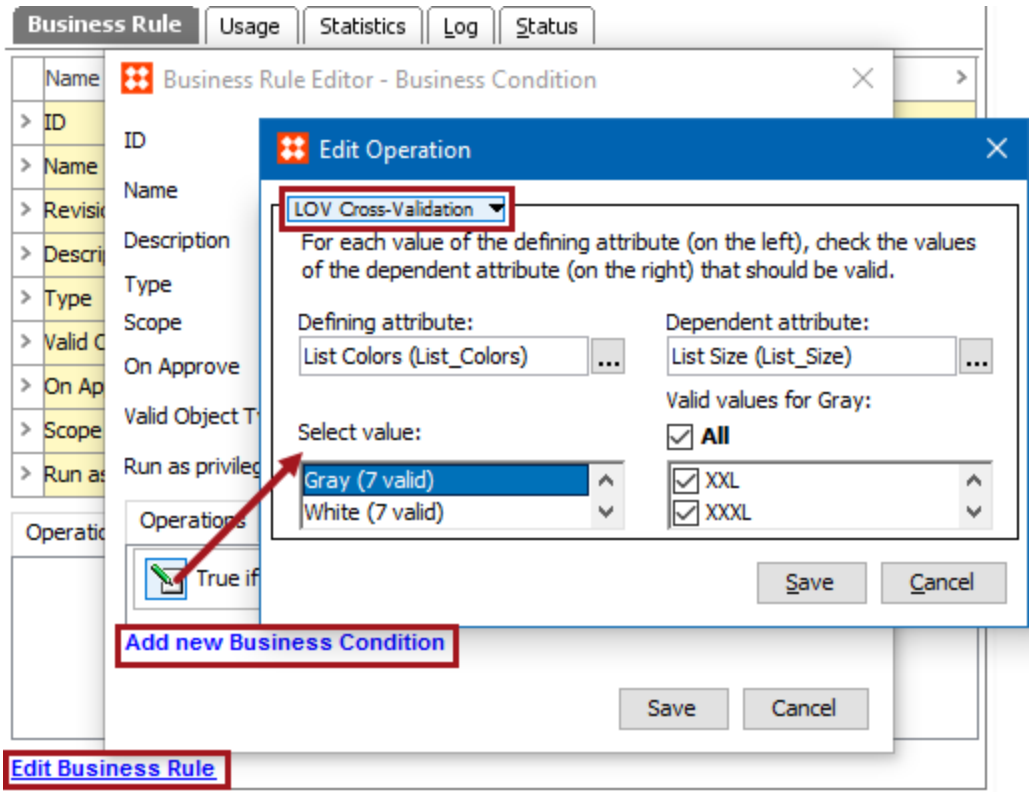
For example, an LOV attribute named ‘Shirt Color’ with values ‘Blue’, ‘Black’ and ‘Red,’ and another LOV attribute named ‘Shirt Size’ with values 'M,' ‘L’, ‘XL’, or ‘XXL.' Based on the availability of different sizes of shirt for a particular color, you can make use of the ‘LOV Cross-Validation’ business condition to specify the relationships between the two LOV attributes like ‘Blue’ > L, XL, ‘Black’ > XXL, ‘M,’ and so on.

Prerequisites

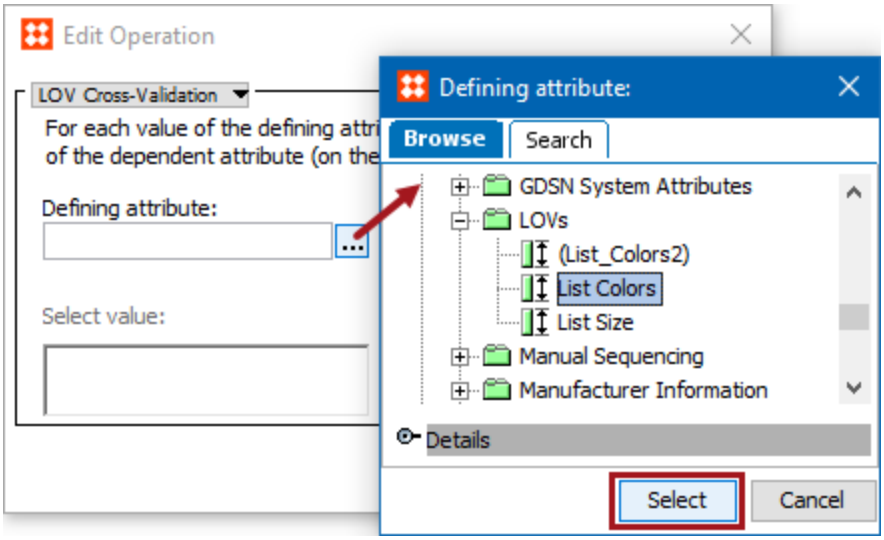
Before using this business condition:

1. Verify the Defining and the Dependent attribute LOVs use IDs as values.
2. Verify the Defining and the Dependent attribute LOVs are configured so that users cannot add new values. This enables the user to specify that the value of one LOV attribute limits which values are valid for another LOV attribute.
3. Verify the Defining and the Dependent attribute LOVs 'Use IDs on values' option is set to **Yes**.
4. Create a business rule as defined in the **Creating a Business Rule or Library** topic.
5. Edit the business rule to configure the operation as defined in **Editing a Business Rule** topic.

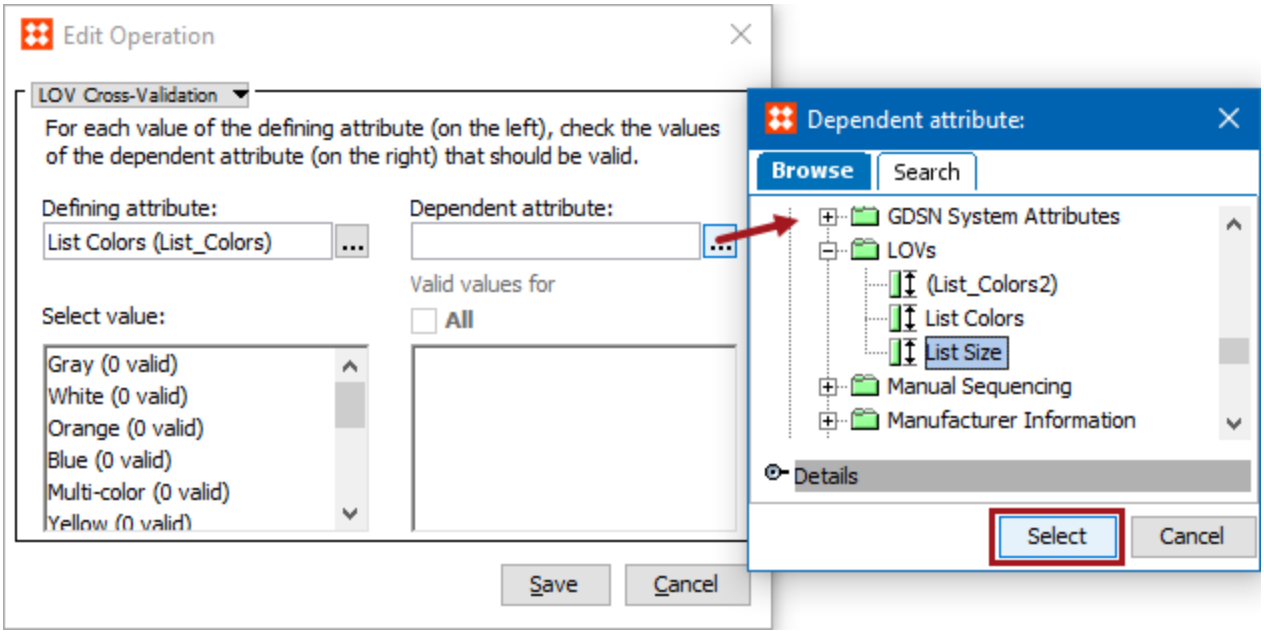
Configuration



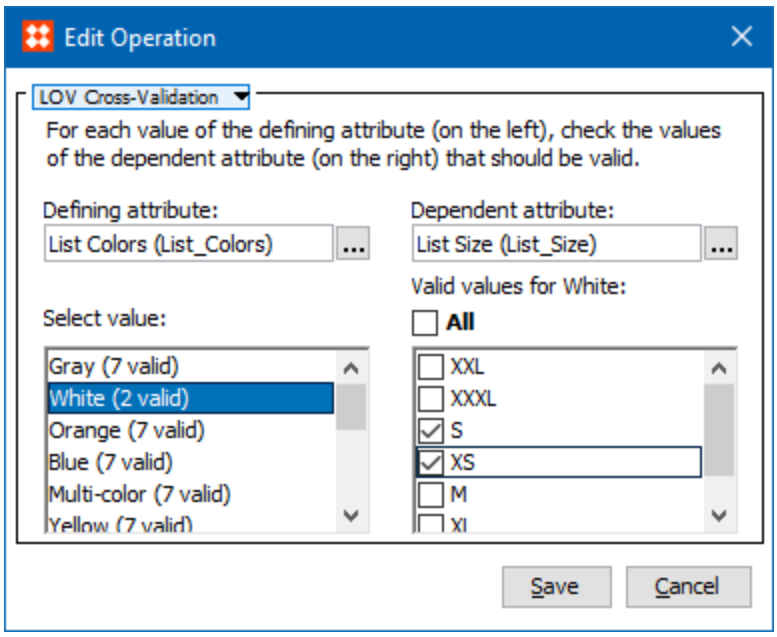
1. On the Edit Operation dialog, select **LOV Cross-Validation** from the dropdown.
2. For the 'Defining attribute' field, click the ellipsis button **(…)** to display the Defining Attributes dialog. Use the Browse or Search tab to identify the defining list of values, select the desired single-valued LOV attribute. Click **Select** to display this attribute as the Defining attribute.



3. For the 'Dependent attribute' field, click the ellipsis button **(…)** to display the Dependent Attributes dialog. Use the Browse or Search tab to identify the dependent list of values, select the desired single or multi-valued LOV attribute. Click **Select** to display this attribute as the Dependent attribute.



4. Select an attribute in the 'Defining Attribute' column list on the left, and set the valid dependent attribute values in the 'Dependent attribute' column list on the right.



Note: The number in the parentheses next to each attribute name specifies how many values of the dependent attribute are configured to be valid. This provides an overview of the configuration without clicking through every defining value.

5. Select the **Save** button to save this business condition.

To setup business conditions in the Web UI, see the **Business Conditions in Web UI** topic in the **Business Rules** documentation.

Business Condition: Reference other Business Condition

This operation allows the current business condition to reference another business condition.

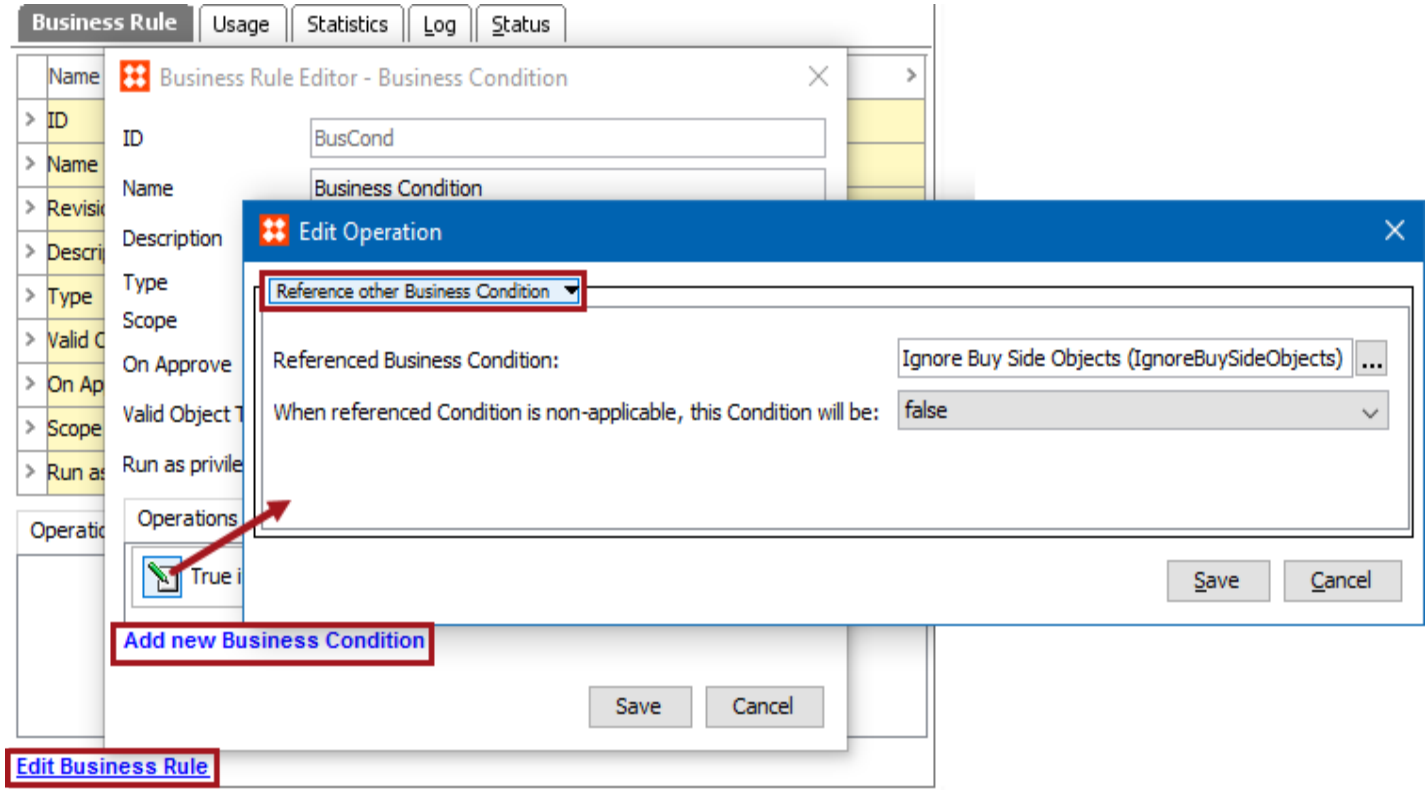
For example, this allows you to use the same business condition in multiple places.

Prerequisites

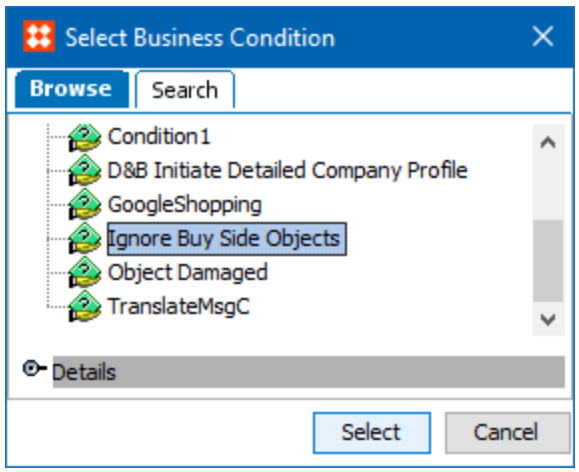
Before using this business condition:

1. Create a business rule as defined in the **Creating a Business Rule or Library** topic.
2. Edit the business rule to configure the operation as defined in **Editing a Business Rule** topic.

Configuration



1. On the Edit Operation dialog, select **Reference Other Business Condition** from the dropdown.
2. For the **Referenced Business Condition** parameter, click the ellipsis button **(...)** to display the Select Business Condition dialog. Use the Browse or Search tab to identify the business condition, select the desired condition and click **Select** to display this attribute as the referenced business condition.



3. For **When referenced Condition is non-applicable, this Condition will be** parameter, select **True** or **False** in the dropdown. This selection determines how invalid conditions respond.
4. Click the **Save** button to add the operation to the business rule editor.

Business Condition: Validate Product Variant

A product variant family is a group of products where the members are considered to be the same product except for variations in the predefined attributes. Variations are, for example, different sizes or colors. This condition works on the Product Variant level, and returns False if the product variant is a duplicate. For more information, see the **Product Variants** topic in the **System Setup / Super User Guide** documentation.

For example, a product folder named 'Shoes' contains children products and has one variant attribute: Size. This condition can be used to identify the duplicate sizes (variants) within the product variant family.

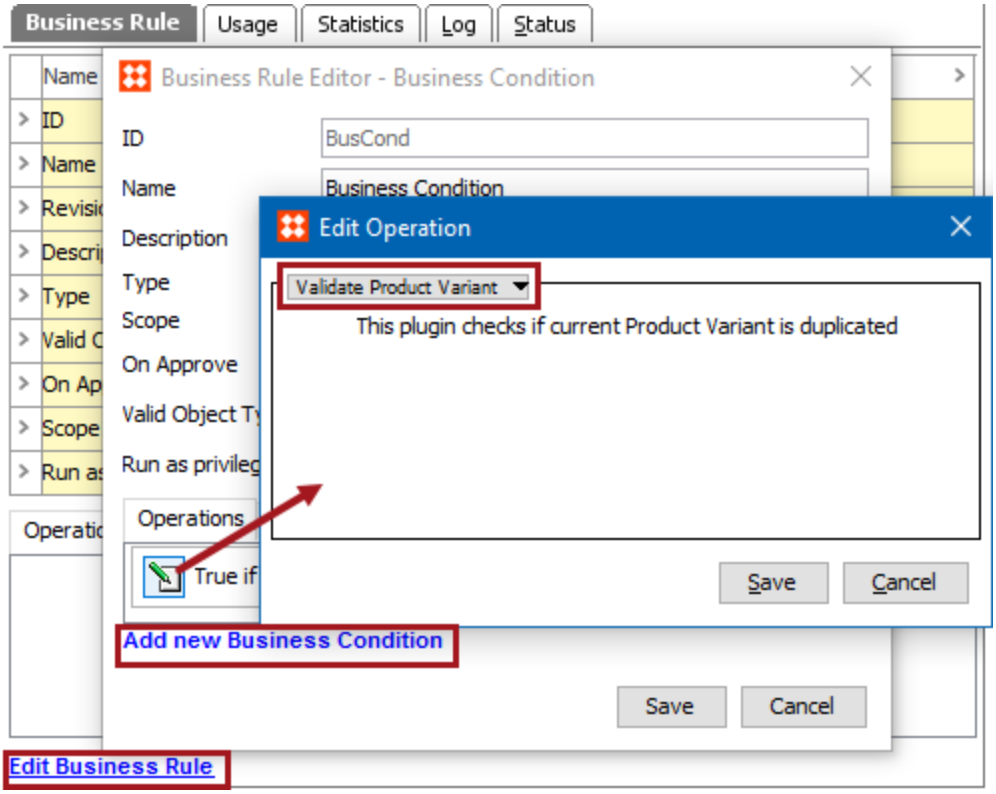
Note: This condition can affect performance based on the number of siblings and variant attributes to be checked.

Prerequisites

Before using this business condition:

1. Verify the configuration on the Product Variant Families is correct since it is used by this condition.
2. Create a business rule as defined in the **Creating a Business Rule or Library** topic.
3. Edit the business rule to configure the operation as defined in **Editing a Business Rule** topic.

Configuration



1. On the Edit Operation dialog, select **Validate Product Variant** from the dropdown.
2. Click the **Save** button to add the operation to the business rule editor.

Business Condition: Valid Hierarchies

The 'Valid Hierarchies' operation will evaluate to **True** if the object the condition is tested on is present below one of the specified hierarchy nodes. If not, the condition will return as **False**.

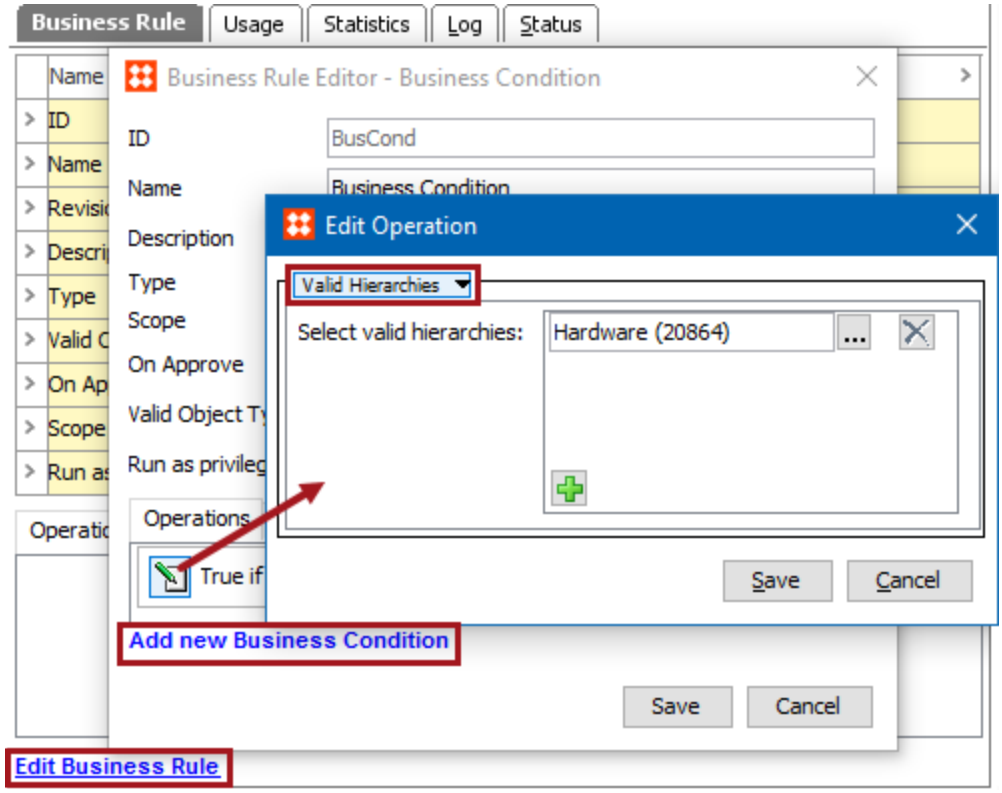
Commonly, this condition is used on the 'Applies If' tab and serves as a precondition for testing the main condition or applying an action.

Prerequisites

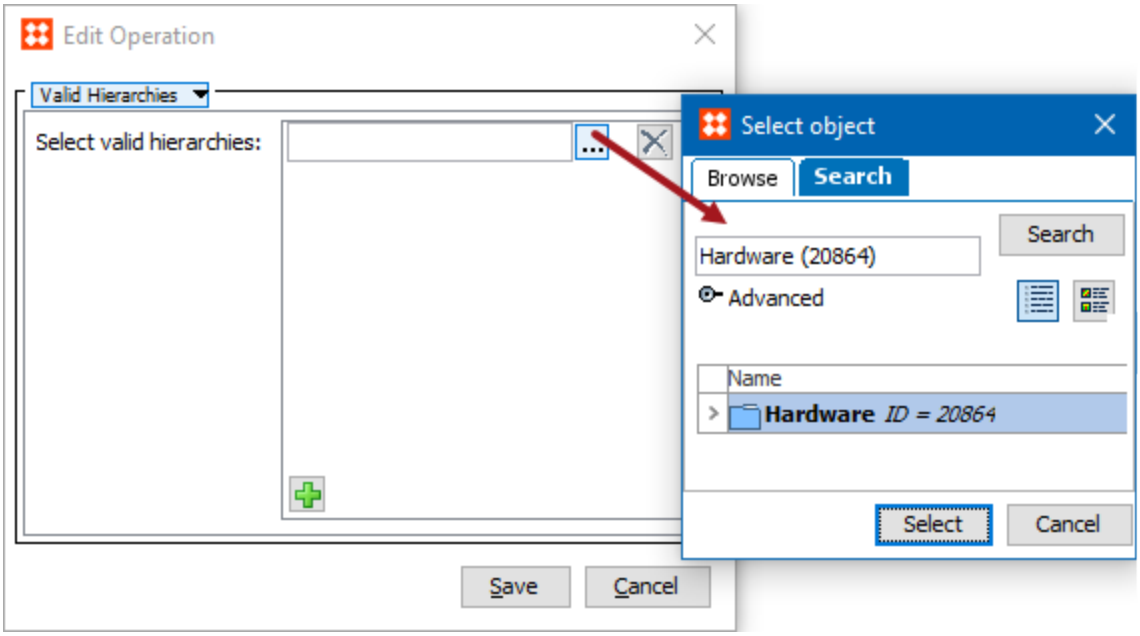
Before using this business condition:

1. Create a business rule as defined in the **Creating a Business Rule or Library** topic.
2. Edit the business rule to configure the operation as defined in **Editing a Business Rule** topic.

Configuration



1. On the Edit Operation dialog, select **Valid Hierarchies** from the dropdown.
2. In **Select valid hierarchies**, click the plus button ( ) to add a hierarchy row. Multiple rows can be added when additional hierarchies should be validated.
3. Click the ellipsis button **(...)** to display the Select Object dialog. Use the Browse or Search tab to identify the relevant hierarchy. Click **Select** to display the hierarchy in the Select valid hierarchies list.



4. Click the **Save** button to add the operation to the business rule editor.

Business Functions

Business functions are basic units of logic that produce an output from an input without affecting the state of data in STEP. Business functions will typically serve as helpers allowing other functionality to delegate a part of their logic to reusable business functions. Business functions are valid on all object types.

As with other business rules like business actions and business conditions, business functions can be configured to appear in a setup group in System Setup.

Note: For information on business conditions and business actions, see the **Business Conditions** topic or the **Business Actions** topic. To create a business rule, see the **Creating a Business Rule or Library** topic.

Creating Business Functions

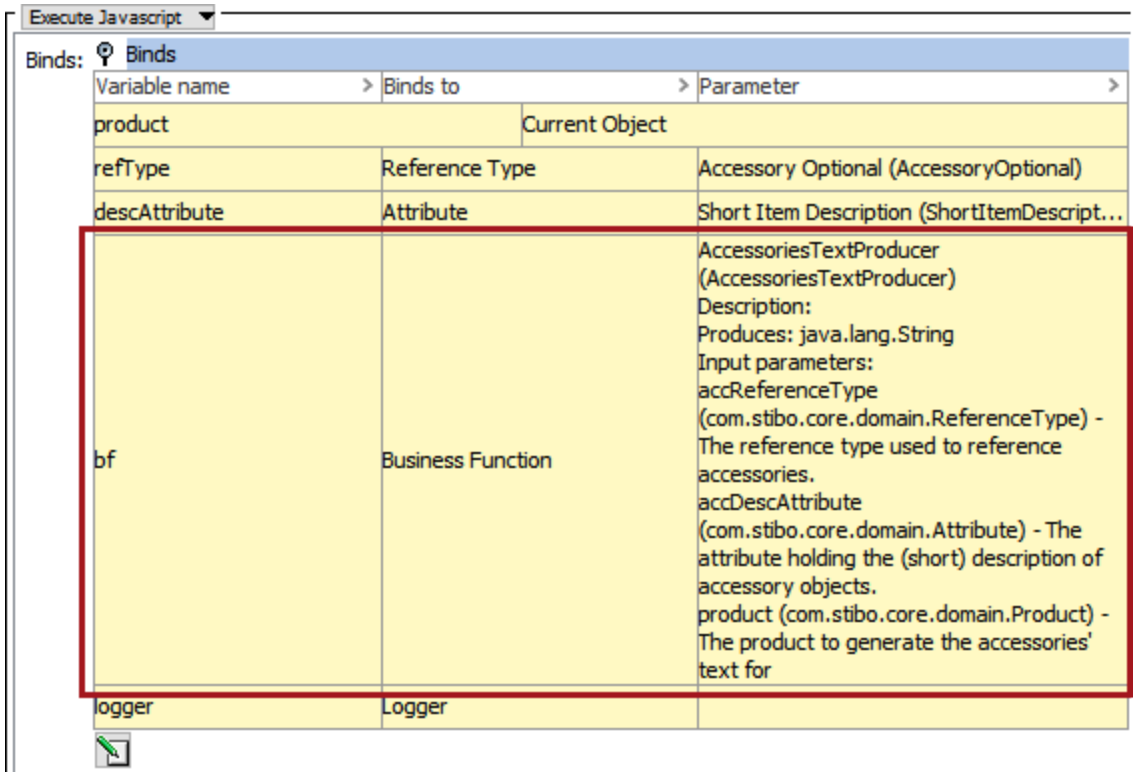
Select a method for creating a business function:

- Use the JavaScript Function operation to write in JavaScript, as described in the **JavaScript Function** section.
- Use the Extension API to create a Java business plugin, as described in the **User-Defined Functions** section.

Calling a Business Function from a JavaScript Business Rule

Below, you will find an example of a business action that will send objects to a previously configured business function to generate the desired output. To set up a business action that calls business functions, see the **Business Action: Execute JavaScript** topic in the **Business Action** documentation.

Note: To evaluate a business function from another JavaScript-based business rule, the business function must be bound to a JavaScript variable.



The bind provides an object implementing the BusinessFunctionScriptingProxy interface for which a user may us the following methods to evaluate the bound business function. These methods are:

- evaluate()
- evaluate(Manager manager, java.util.Map<java.lang.String,java.lang.Object> inputParameters)
- evaluate(java.util.Map<java.lang.String,java.lang.Object> inputParameters)

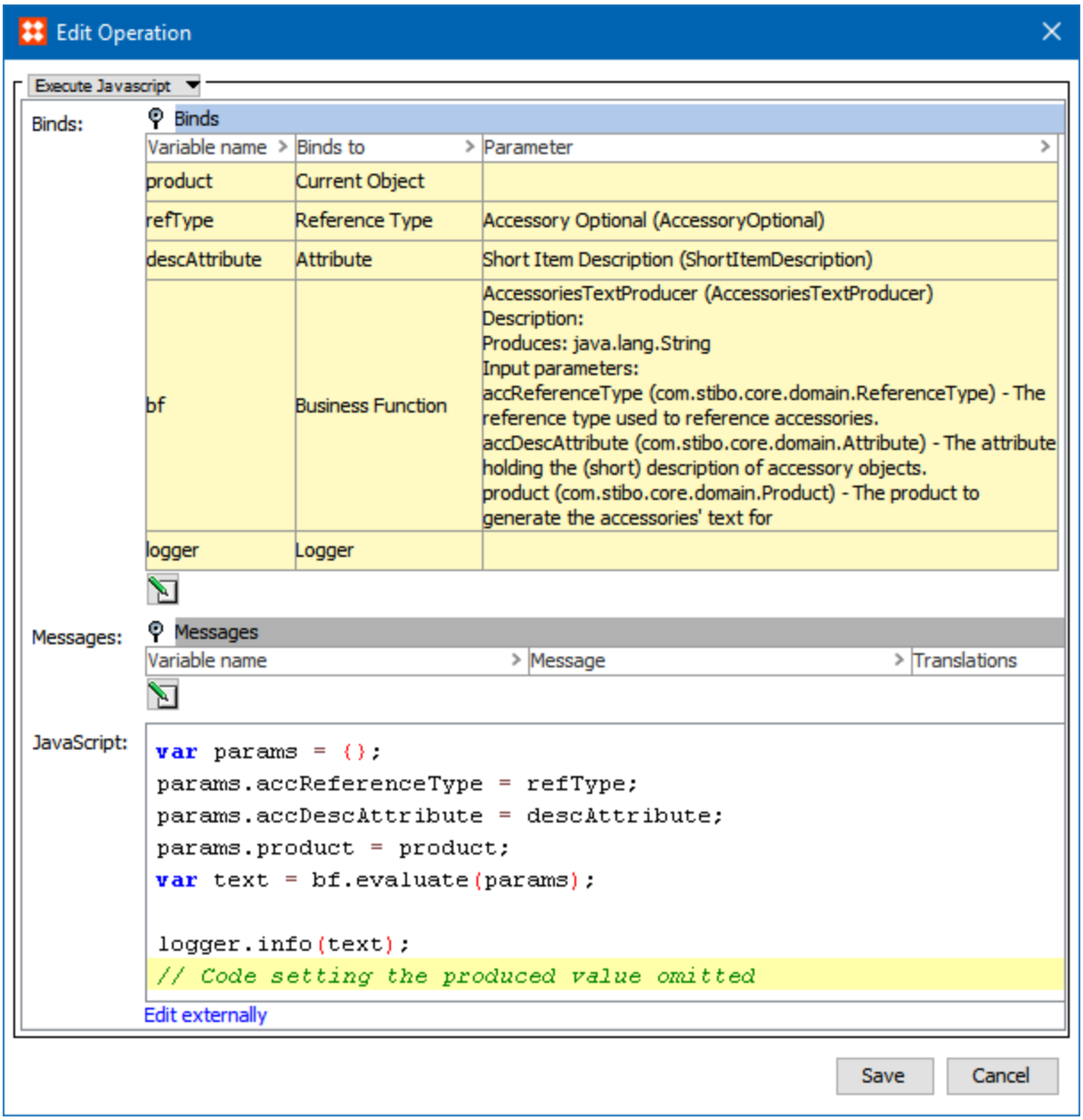
See the scripting API Javadoc in the STEP API Documentation for more information.

While it is possible to instantiate the Java Map in JavaScript, an easier option is to create a JavaScript object for the input parameters as this will be converted to a Map upon execution. Information about the expected input parameters and the output that the function will produce can be found in the bind section.

Note: When calling business functions with Double or Integer input parameters, it is recommended to instantiate the appropriate Java objects instead of relying on the JavaScript-number-to-Java-type conversion. For example:

```
var params = {};
params.aDoubleParameter = new java.lang.Double(2.4);
params.anIntegerParameter = new java.lang.Integer(7);

var result = bf.evaluate(params);
```

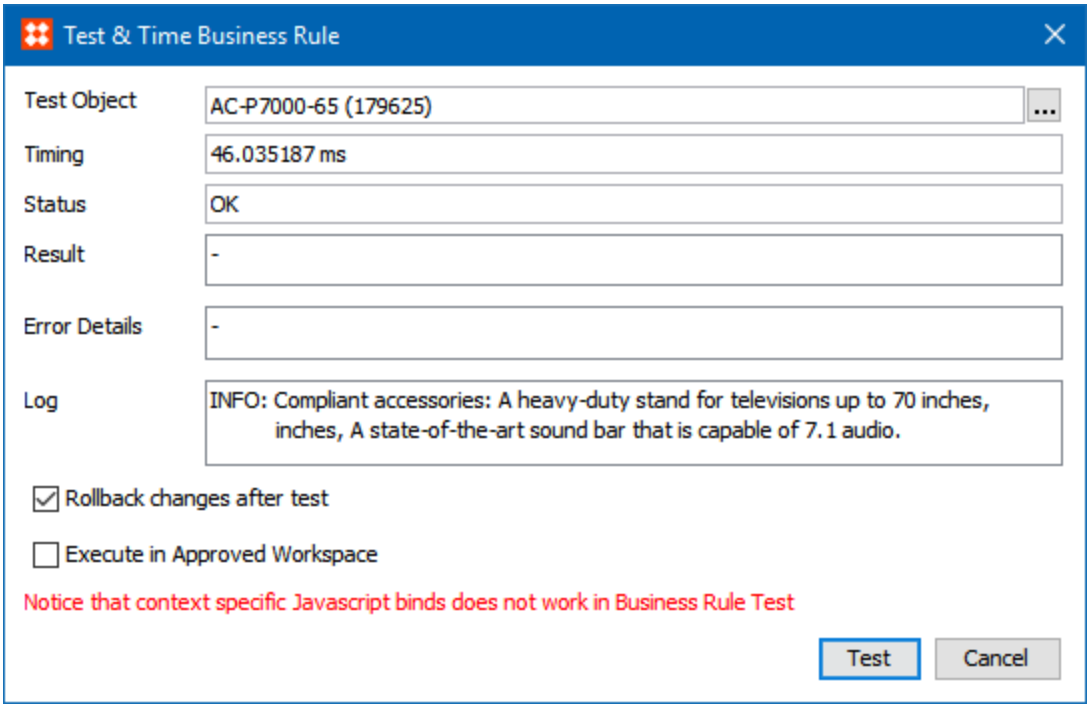
When properly configured, the business action can look like the following:



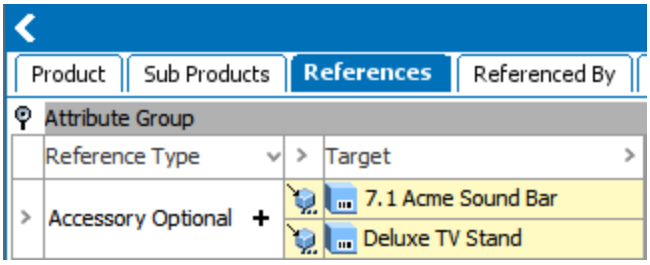
Notice in the Parameter field in the Binds section for the called Business Function, the business function details what type of output can be expected, what variables to provide values for, and what type they must be.

```
var params = {};
params.accReferenceType = refType;
params.accDescAttribute = descAttribute;
params.product = product;
var text = bf.evaluate(params);

logger.info(text);
// Code setting the produced value omitted
```

If we test this business action, we will see the proper results:



In the 'Log' field, the value of the called attribute is shown. In the above example, the value returned from the business function and associated with text is logged. On the AC-P7000-65 product, there are two optional accessories, a 7.1 sound bar and a deluxe TV stand:



For the 7.1 Acme Sound Bar product, the short item description is:

>	Short Item Description	abc A state-of-the-art sound bar that is capable of 7.1 audio.

For the Deluxe TV Stand, the short item description is:

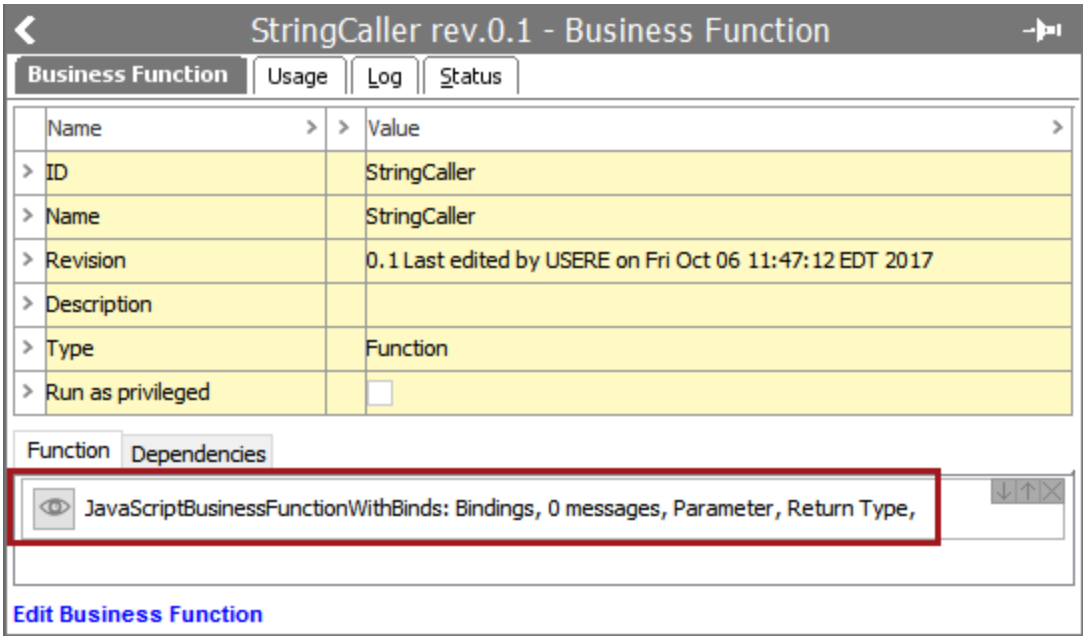
>	Short Item Description	abc A heavy-duty stand for televisions up to 70 inches

In the test, both of these attributes are returned per the JavaScript function.

Log	INFO: Compliant accessories: A heavy-duty stand for televisions up to 70 inches, inches, A state-of-the-art sound bar that is capable of 7.1 audio.

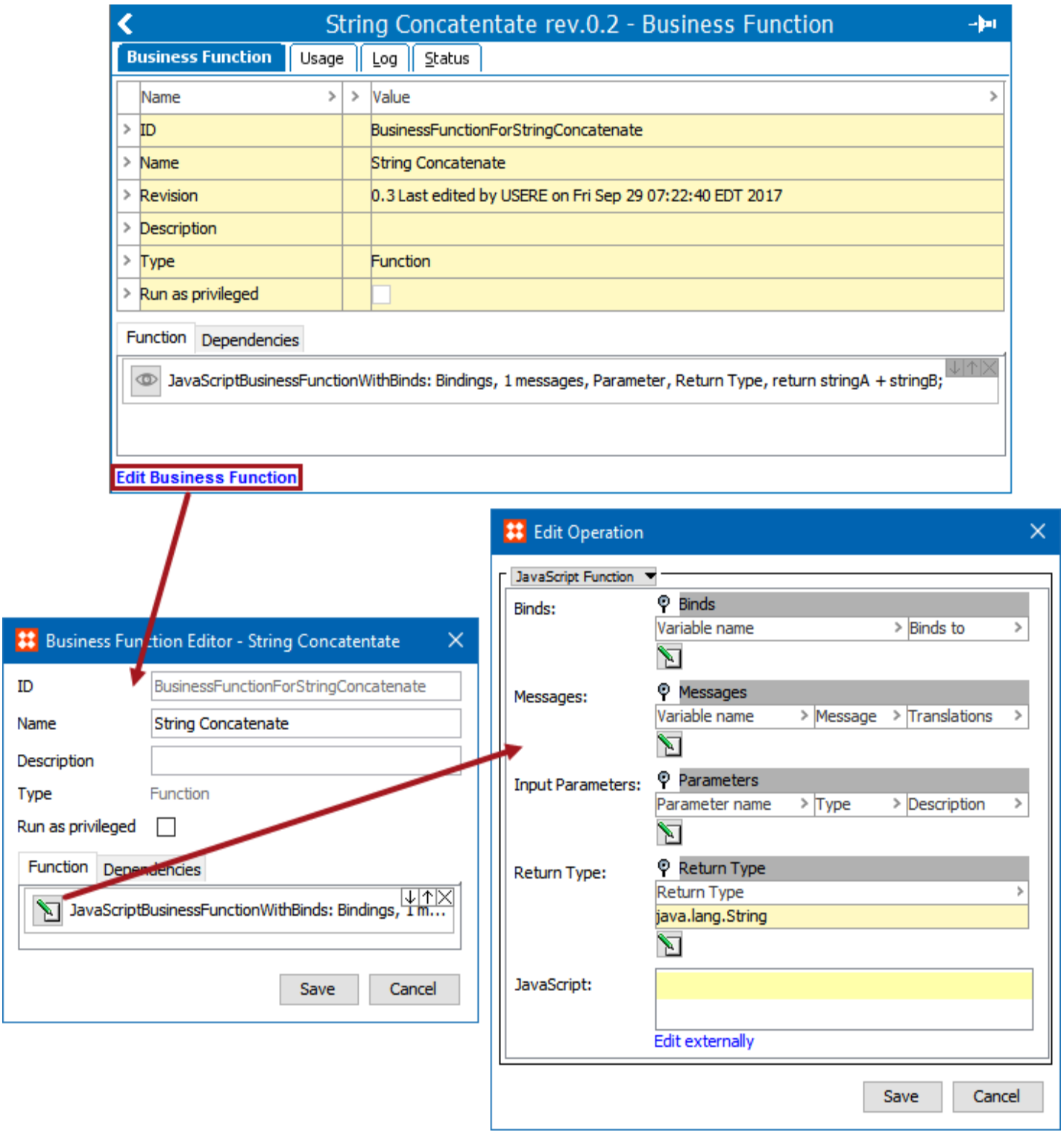
JavaScript Function

The default Business Function operation is the 'JavaScript Function' and displays as 'JavaScriptBusinessFunctionWithBinds' on the Function tab. This operation allows users to define their own functions using JavaScript while binding various STEP data for use.



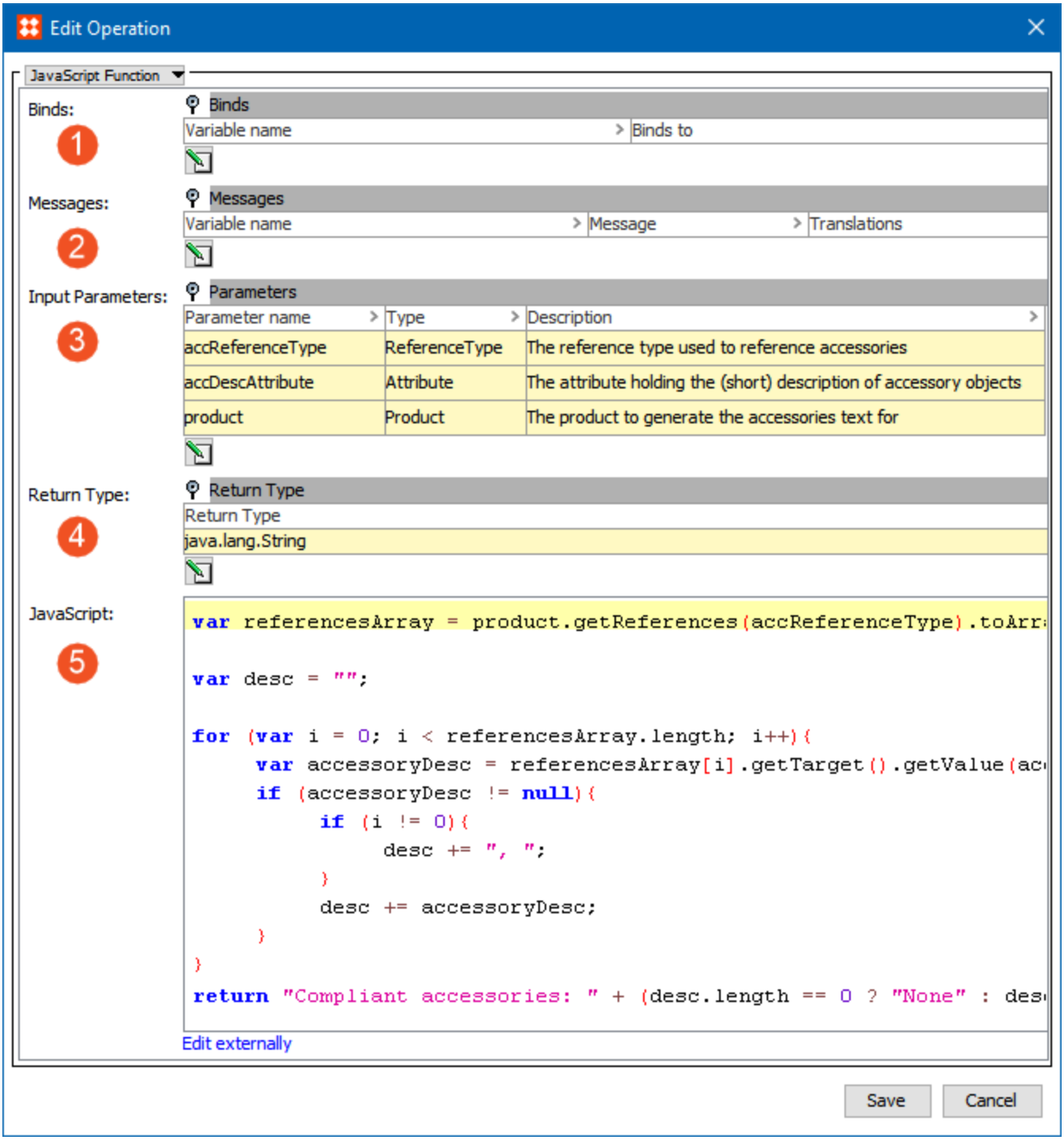
To add the JavaScript to this operation, click the **Edit Business Function** link.

For more information on editing a business function, see the **Editing a Business Rule** topic in the **Business Rules** documentation.



Configuration

In this section, you will find an example of how to configure a Business Function that displays the text of a referenced object's attribute value description.



1. **JavaScript Binds** - This section allows users to bind various STEP objects to JavaScript.

   Note: The dynamic binds available for business functions are 'STEP Manager' and 'Logger.'

   For more information on STEP JavaScript Binds, see the **JavaScript Binds** topic as well as the **Adding a Bind** topic in the **JavaScript** section.

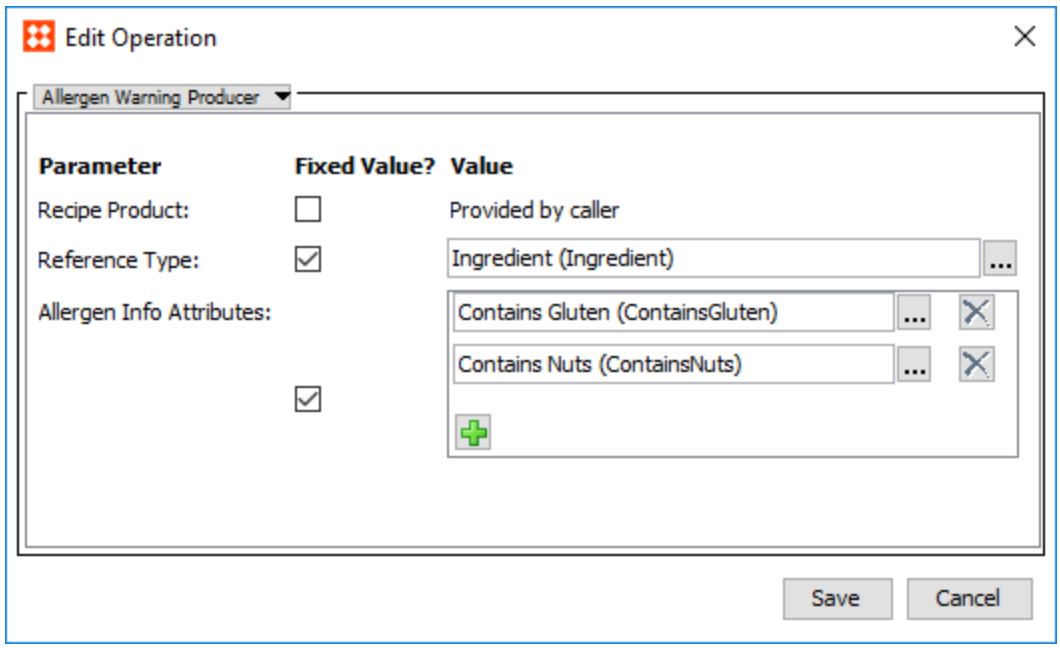
2. Via the **Messages** section, it is possible to define localized error messages that can be returned when an error condition is encountered with this business function. For more information on translatable JavaScript messaging, see the **Localized Messages for JavaScript Business Rules** topic in the **JavaScript** section.
3. The **Input Parameter** section is used for defining the parameters that can be passed to the function to produce an output. A number of STEP and Java types are available.

   Using the provided example, when a user calls this Business Function, they must provide an attribute, references, or a product, which are passed as objects.
4. Via the **Return Type** section, it is possible to define what type of data the business function returns. The only data type that is acceptable as returns from business functions are Java Strings.
5. The **JavaScript** section is where the function logic must be entered. In this example, the values of attributes on referenced objects is returned. The full code used for this functionality can be viewed in the online version of this topic.

See the **Calling a Business Function from a JavaScript Business Rule** topic for information on executing and testing business functions.

User-Defined Functions

While the 'JavaScript Function' is the only out-of-the-box business function plugin, it is possible to develop configurable plugins via the Extension API that will appear in the Edit Operation dropdown. The following screenshot shows the UI of a user-defined plugin named 'Allergen Warning Producer', created with the Extension API:



As seen in this example, with Java business plugins, it is possible to assign a fixed value for some input parameters while leaving others to be provided by the caller. This means that the same plugin can be configured differently in different business functions, thereby exposing different caller 'interfaces.'

For more information on creating business function plugins with the API, see the **STEP Extension API Guide** available on the **STEP API Documentation** link on the WebStart page.

Business Libraries

A Business Library is a set of JavaScript functions that can be reused in multiple business rules. The following should be reviewed when considering the use of a business library:

- When creating multiple JavaScript based business rules, a business library should be used to hold common functions. This makes maintaining the related business rules easier since the JavaScript code exists in a single location. For details, see **Creating a Business Rule or Library**.
- Libraries cannot be called independently, but must be referenced from other actions or conditions.
- Any number of JavaScript business rules can be used to define library functions.
- It is not possible to bind STEP objects when working with libraries, but they can be passed in as arguments.
- For business rules where library functionality is needed, you must declare a dependency to the libraries on the Dependencies tab, as described in the **Editing a Business Library** documentation.

For a JavaScript example of Business Libraries, see the online version of this topic.