STIBO SYSTEMS
MASTER DATA MANAGEMENT

USER GUIDE

Business Rules

Release 2023.3 (11.3) – September 2023

Table of Contents

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, refer to 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 conditions, refer to 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, refer to 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, refer to **Business Libraries** topic documentation.

For more information on differentiating between scenarios where one business rule is more useful than another, refer to the **Business Rule 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, Function, or Library**.
3. Edit the business rule as described in the **Editing a Business Rule or Function**, 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, refer to **Calling a Business Function from a JavaScript Business Rule**.
5. Delete a business rule as described in **Deleting a Business Rule, Function, or Library**.
6. Maintain or modify local business rules as described in **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**.
9. JavaScript-based business rules can be created, maintained, and (unit) tested outside STEP as described in the **VCSI: Editable Business Rules Format** topic in the **Version Control System Integration** documentation.

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, refer to the **Business Rules and Workflows** topic in the **Workflows** documentation.

Using Business Rules in STEP

Wherever business rules are used, 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, refer to **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, refer to **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, refer to **Run Business Rule Operation** documentation.

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

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

- **Event Processors -** Actions can determine when and how to act upon the events from the event processor. This is useful when changes occur on objects and custom actions need to occur.

  For more information, refer to **Execute Business Action Processing Plugin Parameters and Triggers** documentation or **Execute Business Action for Event Batch Processing Plugin Parameters and Triggers** 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, refer to the **Gateway Integration Endpoint Bind** documentation in the **Resource Materials** online help. *Gateway Integration Endpoints (GIEPs) do not always use the REST plugin; the Gateway Integration Endpoint Bind topic only applies for GIEPs that use the REST plugin.*

- **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, refer to **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, refer to Matching section of the **JavaScript Binds** topic in the online help **Resource Materials** documentation or the **Golden Records Survivorship Rules** topic in the **Matching, Linking, and Merging** 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, refer to **OIEP - Event-Based - Event Triggering Definitions Tab** documentation or **OIEP - Pre-Processor - Business Action** documentation.

- **Web UI -** Actions can be executed to update the object. Conditions can be evaluated to improve data validity.

  For more information, refer to **Business Rules 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, refer to **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.

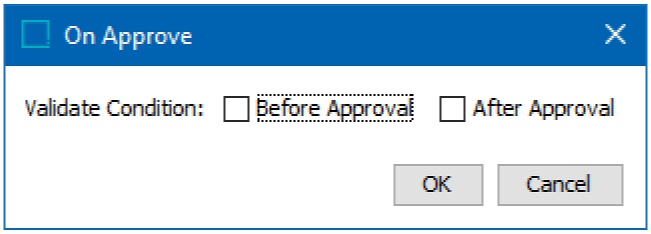
Global business rules are set to run on approval in the Business Rule editor. For more information, refer to the **On Approve** section of the **Editing a Business Rule or Function** topic.

For information on running business rules during the import process, refer to 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 in the 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.



Validate Condition: □ Before Approval □ After Approval

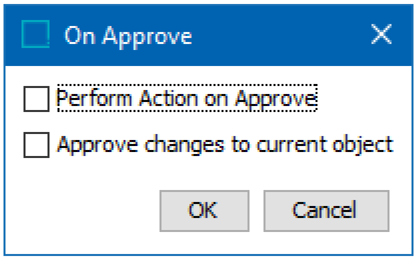
Note: The condition is only guaranteed to be met for a given object at the moment it is tested. If the condition depends on data owned by other objects, those objects can change and be approved without the condition being reevaluated. Therefore, it is common setup to test data on the objects that own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

For details on adding a condition, refer to the **Specifying a Business Condition Operation** 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.


On Approve
☐ Perform Action on Approve
☐ Approve changes to current object
OK
Cancel

For details on adding a condition, refer to the **Specifying a Business Action Operation** 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:

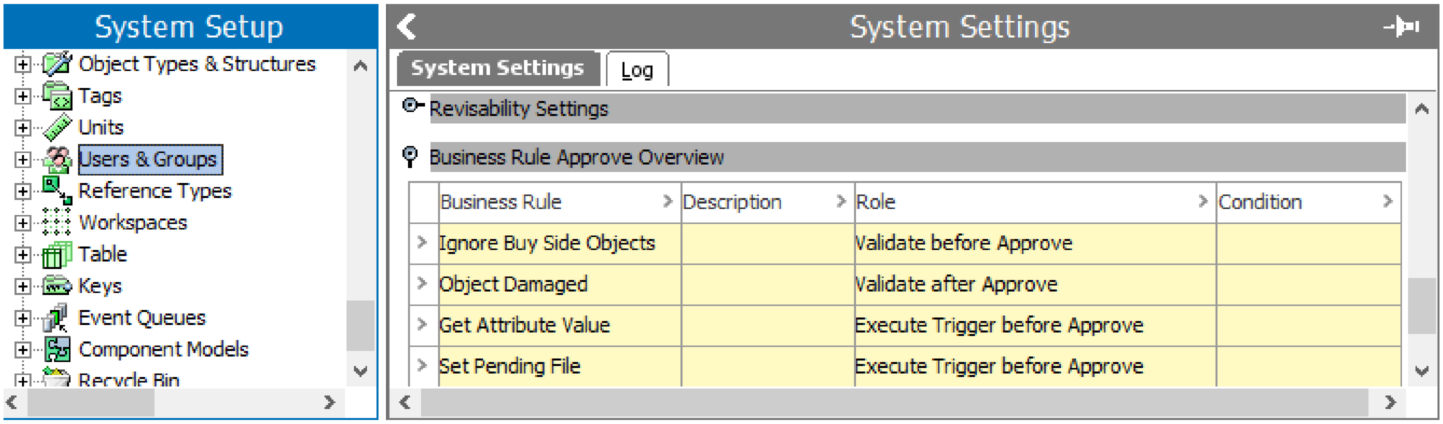
- Changes made by actions are not kept when conditions before or after approval are false.
- If the 'Approve changes to current object' option is not checked and an action makes a change to the object being approved, a complete approval cannot be achieved.
- Actions can make changes that cause a condition 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.



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, refer to the **Conditional Attribute Display** documentation in the **System Setup**.

For information on adding a bind, refer to the **Adding a Bind** topic in the **Resource Materials** online help.

JavaScript business rule bind for Conditional Validity

The Conditionally Invalid Values bind can be used in a JavaScript business action. This resolves to a (possibly empty) set of values, and the set contains all values which have value conditions that evaluate to 'false' for the current node.

> **Note:** 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();
   //write 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

The workflow state bindings can be used for conditions and actions that, upon import, can resolve and be used for general evaluation and handling of the product being imported.

This is an example business condition applied on import of a maintenance Smartsheet and 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 returned 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) {
   //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
}
//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 workflow " + state.getWorkflow().getTitle();
   logger.info(msg);
   return msg;
}
logger.info('Accepted: a task exists, 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 that are missing a selling price from a specific vendor. A business condition can filter for the vendor during profiling, which then can check the selling price value.

For more information, refer to the **Data Profiling** documentation.

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, refer to 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 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.

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. This means that the test is not limited to data in the import file.

Actions are only executed during import if all conditions evaluate to true or a warning is reported. This means, contrary to the approval process described in the **Business Rules on Approval** section, actions cannot make changes that cause the conditions become true.

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. You cannot delete a business rule if it is used by an import configuration.

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

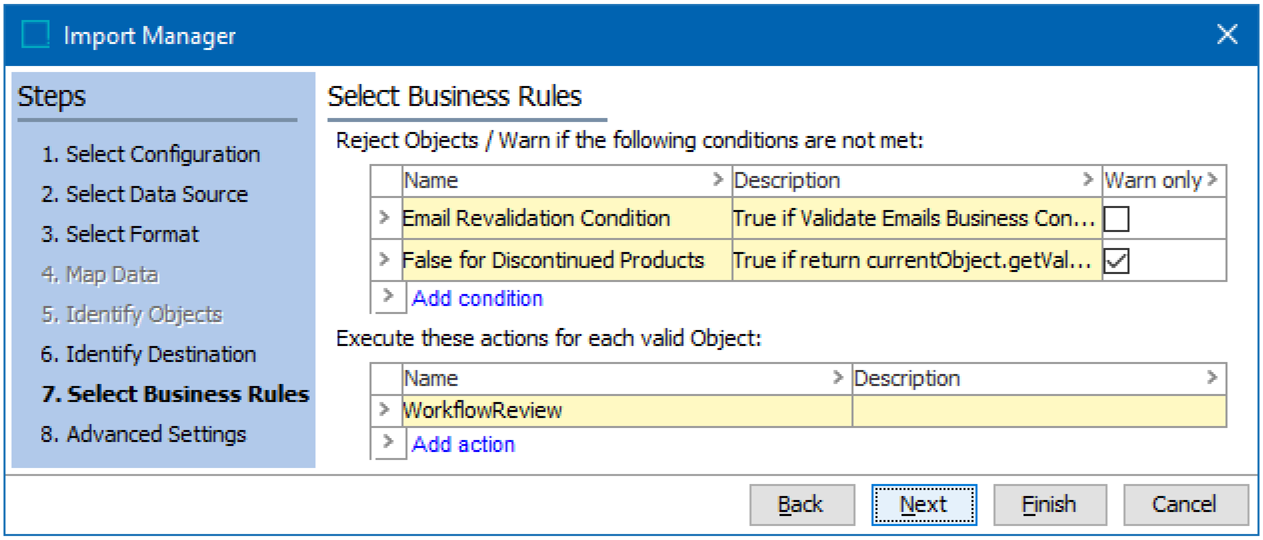
1. In System Setup, expand the Business Rules node.
2. Select the relevant business rule, and click the Usage tab.

Add a Business Rule to an Import

Business rules can be added with the inbound data tools: Import Manager and IIEPs.

Import Manager

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



Select Business Rules

Reject Objects / Warn if the following conditions are not met:

Name	Description	Warn only
Email Revalidation Condition	True if Validate Emails Business Con...	□
False for Discontinued Products	True if return currentObject.getVal...	☑
Add condition		

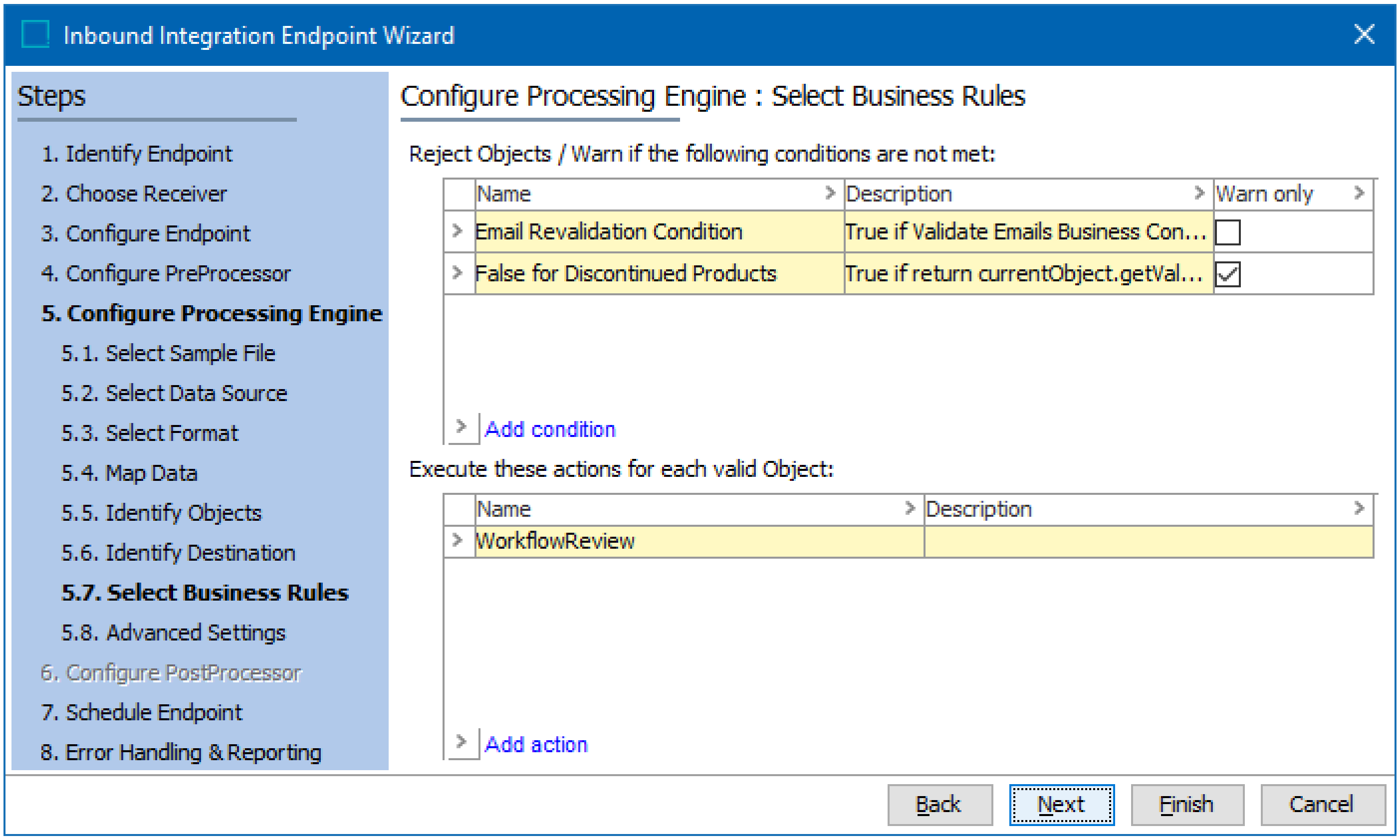
Execute these actions for each valid Object:

Name	Description
WorkflowReview	
Add action	

The saved import configuration includes any selected business rules. For more information, refer to **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, refer to **IIEP - Configure Processing Engine** topic in the **Data Exchange** documentation.



Configure Processing Engine : Select Business Rules

Reject Objects / Warn if the following conditions are not met:

Name	Description	Warn only
Email Revalidation Condition	True if Validate Emails Business Con…	□
False for Discontinued Products	True if return currentObject.getVal…	☑

Execute these actions for each valid Object:

Name	Description
WorkflowReview	

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 first conditions without 'Warn only' enabled, then conditions with 'Warn only' enabled, and finally actions.

- **Conditions** - During the import, both changed and unchanged objects are validated against the business conditions. If the validation fails, the change is rejected unless the 'Warn only' parameter is enabled for the condition. When 'Warn only' is not enabled, the changes are not imported, and an error message is logged in the Import Execution Report. When 'Warn only' is enabled, the changes are imported, and a warning message is logged in the Import Execution Report.

Note: To prevent Actions from running, objects are always validated, even if they are unchanged. 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. If a business action results in an error, the import skips the object. Errors are logged in the Execution Report. Too many errors may stop the import process. It is therefore important that business rules are created to handle expected exceptions.

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

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.
- Based on the Condition Handling Option selected for each business condition, business conditions will run in the following sequence: those with 'Reject and Include Error' selected, those with 'Import and Include Warning' selected, and those with 'Reject and Include Information' selected.
- 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 by business rules. (The references are deferred until after the business rules have run.) 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 Rules in Web UI

Both business conditions and business actions can be applied in the Web UI.

Business Conditions

The Web UI enables admin users to apply business conditions that validate user-entered data in real time, notifying the user immediately if an applied business condition has not been fulfilled.

Business conditions can also be used to filter Lists of Values (LOVs). By applying a condition that validates if an acceptable value has been selected from an LOV, the user is functionally disallowed from selecting a value that violates the condition.

For details on configuring business conditions in the Web UI, refer to the **Data Validation in Web UI** topic in the **Web User Interfaces** documentation.

Business Actions

Via the Web UI designer, business actions can be triggered by action buttons and can be assigned to individual screens using the Business Action parameter.

For details on assigning business actions, refer to the following topics in the **Web User Interfaces** documentation:

- **Action Buttons**
- **Configuring a Mass Creation Screen**
- **Parameterized Business Actions in Web UI**
- **Run Business Action Component**

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, refer to 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, WorkflowFunctionHome, Task, and Workflow interfaces. Using this functionality, however, requires in-depth knowledge about workflows.

Business Rule Use Cases

The following list includes 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.

Translation status of a node

Business functions allow users to access the translation status of any object considering both its context and workspace. It ensures that only translated objects are exported and checks translation status as part of an approval. It also factors translation status into the sufficiency score of an object.

Introduction to Writing Business Rules

Software developers often write business rules in JavaScript since it provides the ability to precisely control the inputs, outputs, and outcomes. Developers may also use the information included in the STEP API Documentation, available at [system]/sdk or accessible from the Start Page. However, users who have a good understanding of their own STEP system but limited technical knowledge may want to modify existing JavaScript business rules or write new JavaScript business rules.

This topic discusses determining the type of business rule needed, understanding the business requirements, implementing the logical parts of the requirements, and testing that the business rule meets the goal.

Prerequisites

The example in this topic uses a business condition, which can be created and configured as follows:

1. Create a new business condition. Refer to the **Creating a Business Rule, Function, or Library** topic.
2. Configure the Evaluate JavaScript operation. Refer to the **Business Condition: Evaluate JavaScript** topic.

Determining the Business Rule Type

This example demonstrates how to write a business rule to read the value on an attribute and return 'true' or 'false' based on the value. This type of business rule is a 'check' or 'validation' rule, which means it is a business condition.

A business condition is used to determine if a subsequent action or decision is needed, for example:

- Allow or reject the transition of a product in a workflow.
- Allow or reject the approval of a product.
- During import, reject object creation or modification.

Understanding the Requirements

Before writing the JavaScript code for a business rule, identify the logical parts of the requirements. In this example, the logical parts are:

1. Get access to the object in context. This is the object that is being imported, getting approved, or transitioning in the workflow.
2. Read the value of an attribute on the object.
3. Compare the attribute value against a fixed value and determine if they match.
4. Return 'true' if the values match or return 'false' if the values do not match.

Implementing the Plan

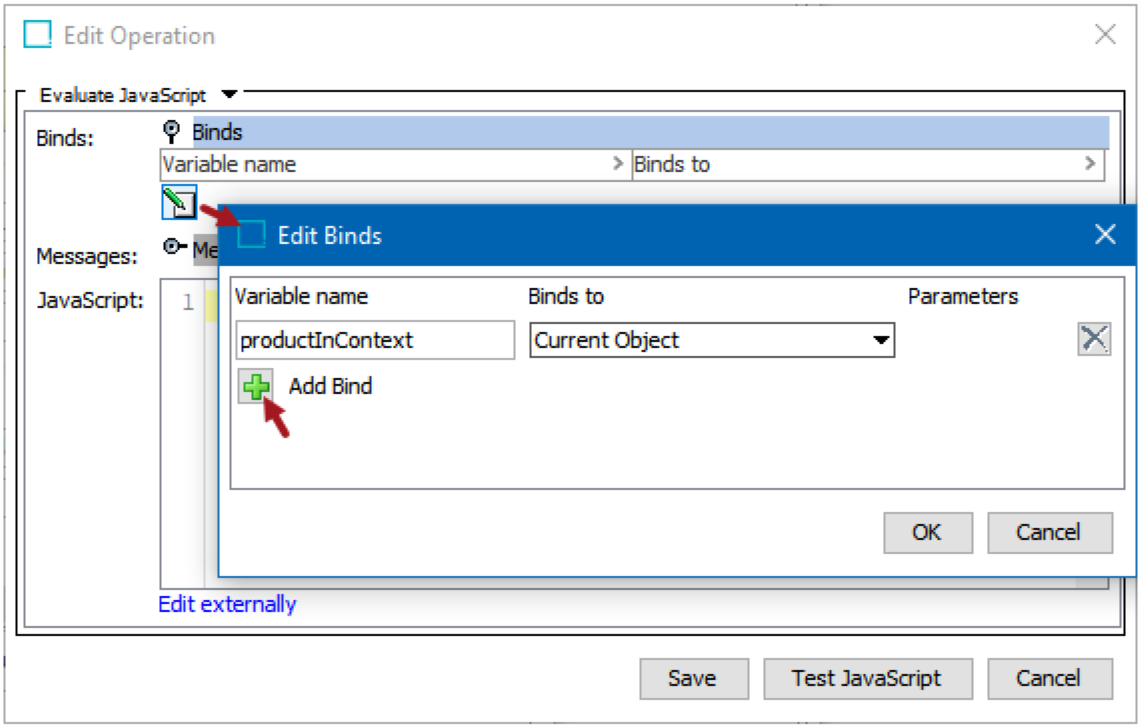
Translating the logical parts of these requirements into a JavaScript business condition includes the steps below.

Get access to the object in context

In a STEP business rule, the Evaluate JavaScript operation uses a 'bind' to access an object in context. 'Binds' allow accessing many types of data and configuration objects and allows the JavaScript code to act on them. A bind requires a variable to represent it within the code. The 'Current Object' bind gives access to values on the object being evaluated.

To implement this part of the business condition, add a bind for the current object:

- Add 'productInContext' for the 'Variable name' parameter.
- Select 'Current Object' from the 'Binds to' dropdown.



Variable name: productInContext
Binds to: Current Object

This binding creates the 'productInContext' variable, which represents the object that this business rule is processing and makes it available to the JavaScript code. Since the 'productInContext' variable is essentially a Java object, the code can read its parameters and invoke methods on the object. The available parameters and methods are documented in the **Scripting API** section of the **STEP API Documentation**, available at [system]/sdk or accessible from the Start Page.

For example, some of the methods are available for the 'Current Object' bind include:

```
productInContext.getName()
productInContext.getParent()
productInContext.getChildren()
productInContext.getValue()
```

Read the value of an attribute on the object

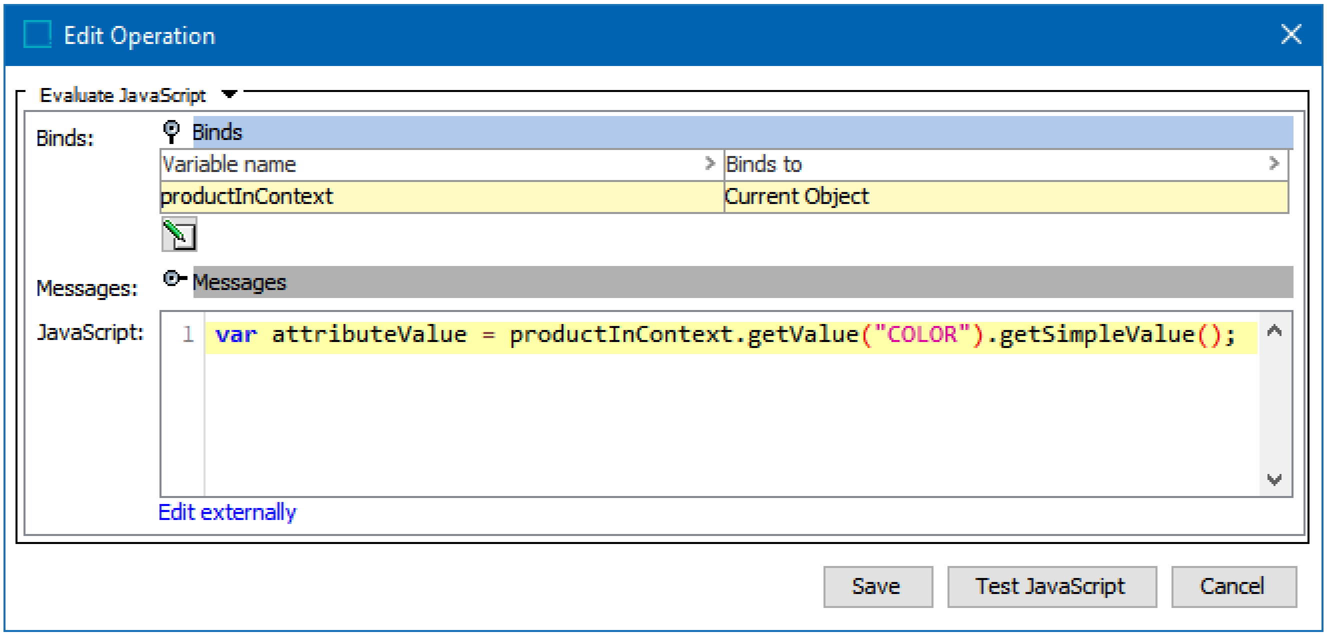
With access to the object in context, now the value of an attribute can be determined. In this example, the attribute ID is 'COLOR' and it is valid for the product being processed by the business condition.

Within the JavaScript code, the variable 'attributeValue' is defined to:

- use the current object, indicated by the 'productInContext' bind variable.
- use the method 'getValue' to get the value of the 'COLOR' attribute.

Add the following code to the JavaScript parameter:

```
var attributeValue = productInContext.getValue("COLOR").getSimpleValue();
```



Variable name	Binds to
productInContext	Current Object

```
var attributeValue = productInContext.getValue("COLOR").getSimpleValue();
```

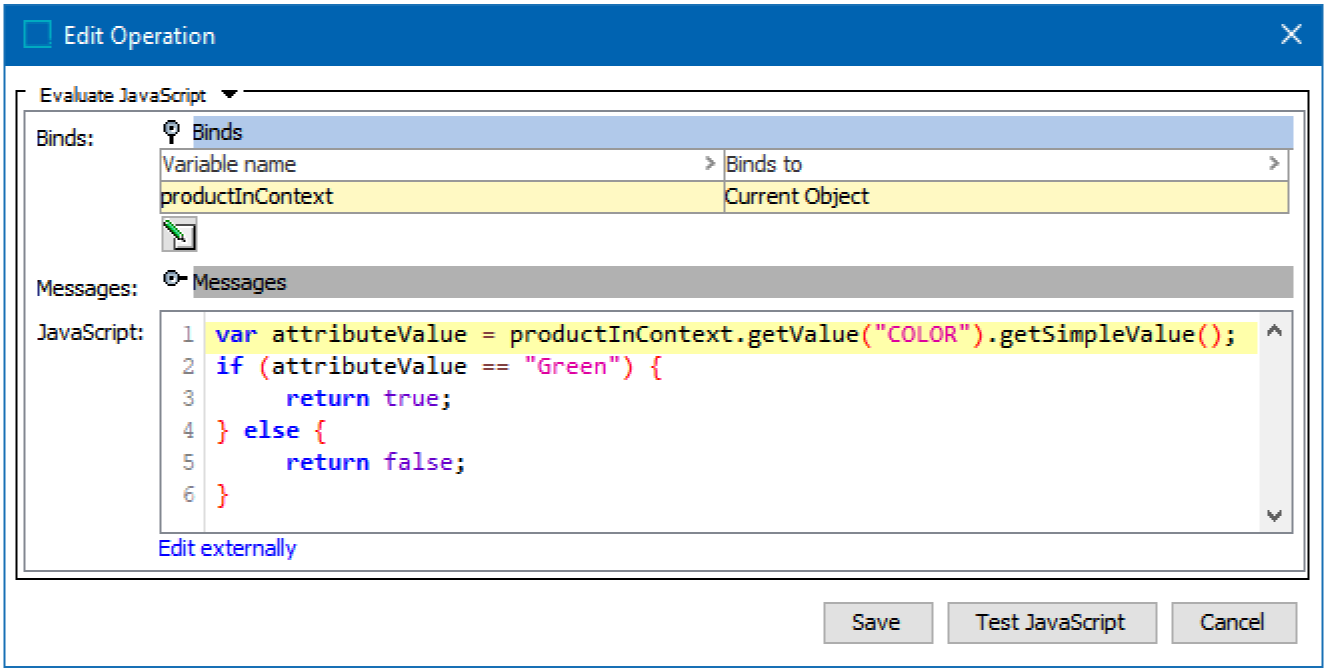
Compare the attribute value against a fixed value

An `if / else` conditional statement compares the value of the current object to a fixed value and determines the outcome.

- The `if` block specifies code to be executed if the condition is true.
- The `else` block specifies code to be executed if the condition is false.

Add the following code to the JavaScript parameter:

```
if (attributeValue == "Green") {
        return true;
} else {
        return false;
}
```



Variable name	Binds to
productInContext	Current Object

```
var attributeValue = productInContext.getValue("COLOR").getSimpleValue();
if (attributeValue == "Green") {
     return true;
} else {
     return false;
}
```

Return 'true' or 'false'

The 'return' method (included above) generates the result of the business condition.

In this example, the result is:

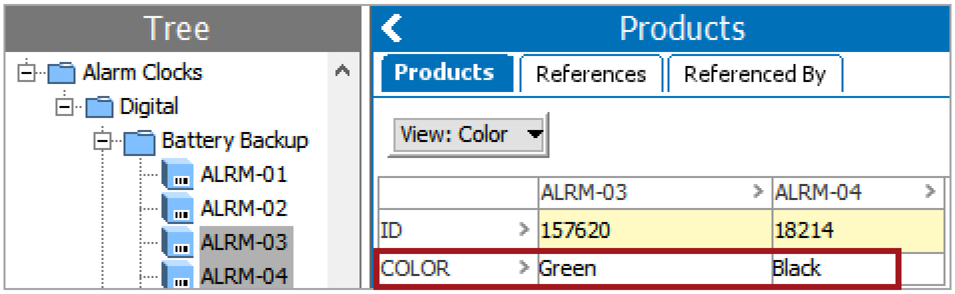
- True for objects with a value of 'Green' for the 'COLOR' attribute.
- False for objects with any value other than 'Green' for the 'COLOR' attribute.

Testing the Business Rule

Before applying a business rule to your data, test your JavaScript against applicable objects to identify coding errors.

> **Note:** Valid object types are defined when creating a business rule. Ensure that the object type of your test objects is set on the business rule. For more information, refer to the **Testing a Business Rule** topic.

To verify the JavaScript code results, test against an object that should return 'true' and against one that should return 'false.'



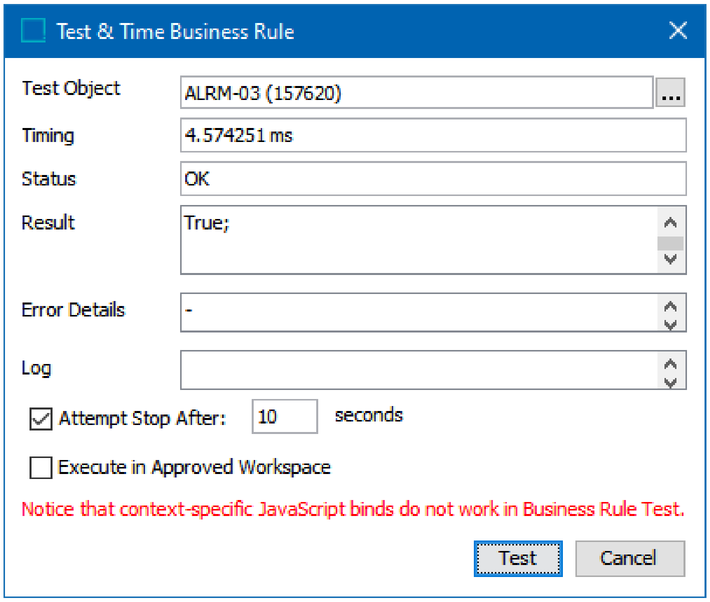
	ALRM-03	ALRM-04
ID	157620	18214
COLOR	Green	Black

On the Edit Operation dialog (shown in the previous section), click the **Test JavaScript** button.

On the Test & Time Business Rule dialog (shown below), select a test object (the object to run the business condition against) and click the **Test** button.

The 'productInContext' bind variable in the JavaScript code recognizes the 'Test Object' as the 'Current Object.'

- The ALRM-03 product has the value of 'Green' and the business rule result is True.
- The ALRM-04 product has the value of 'Black' and the business rule result is False.

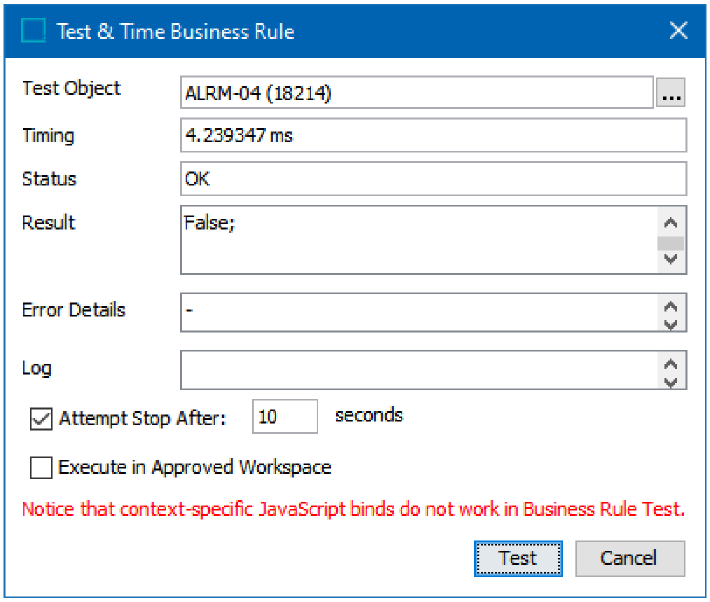


Test Object	ALRM-03 (157620)
Timing	4.574251 ms
Status	OK
Result	True;
Error Details	-
Log	

☑ Attempt Stop After: 10 seconds

□ Execute in Approved Workspace

Notice that context-specific JavaScript binds do not work in Business Rule Test.



Test Object	ALRM-04 (18214)
Timing	4.239347 ms
Status	OK
Result	False;
Error Details	-
Log	

☑ Attempt Stop After: 10 seconds

□ Execute in Approved Workspace

Notice that context-specific JavaScript binds do not work in Business Rule Test.

Initial Set Up for Business Rules

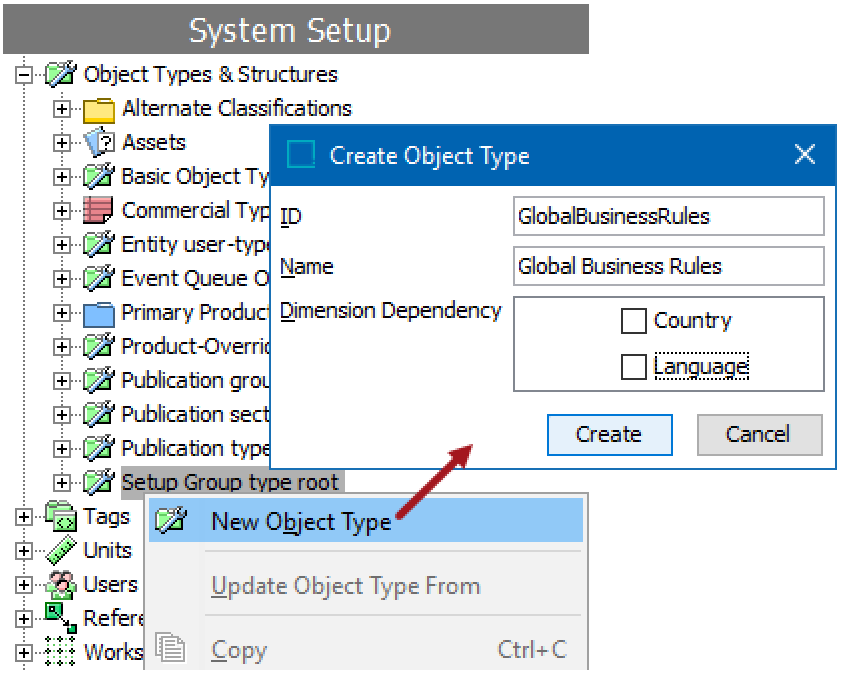
Before business rules, functions, and libraries can be created, a one-time configuration must be completed that includes:

- Create System Setup object types to hold the 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 set up is covered in the steps below. These steps can be carried out multiple times to separate different type of objects for organizational purposes.

Configuration

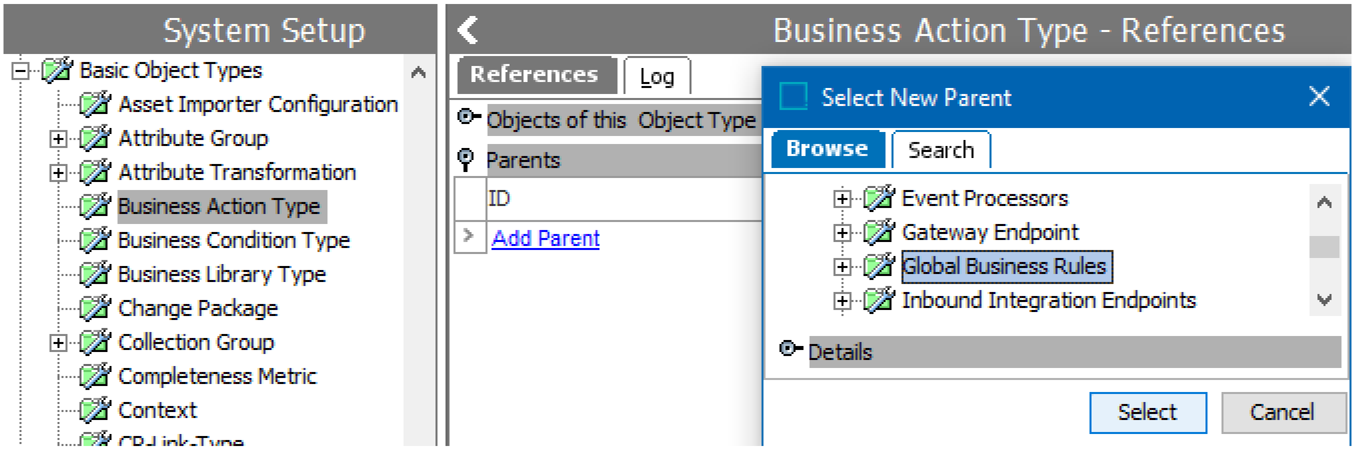
1. In System Setup, expand Object Types & Structures.
2. Right-click Setup Group type root and click **New Object Type** to display the Create Object Type dialog.



ID: GlobalBusinessRules
Name: Global Business Rules
Dimension Dependency: □ Country □ Language

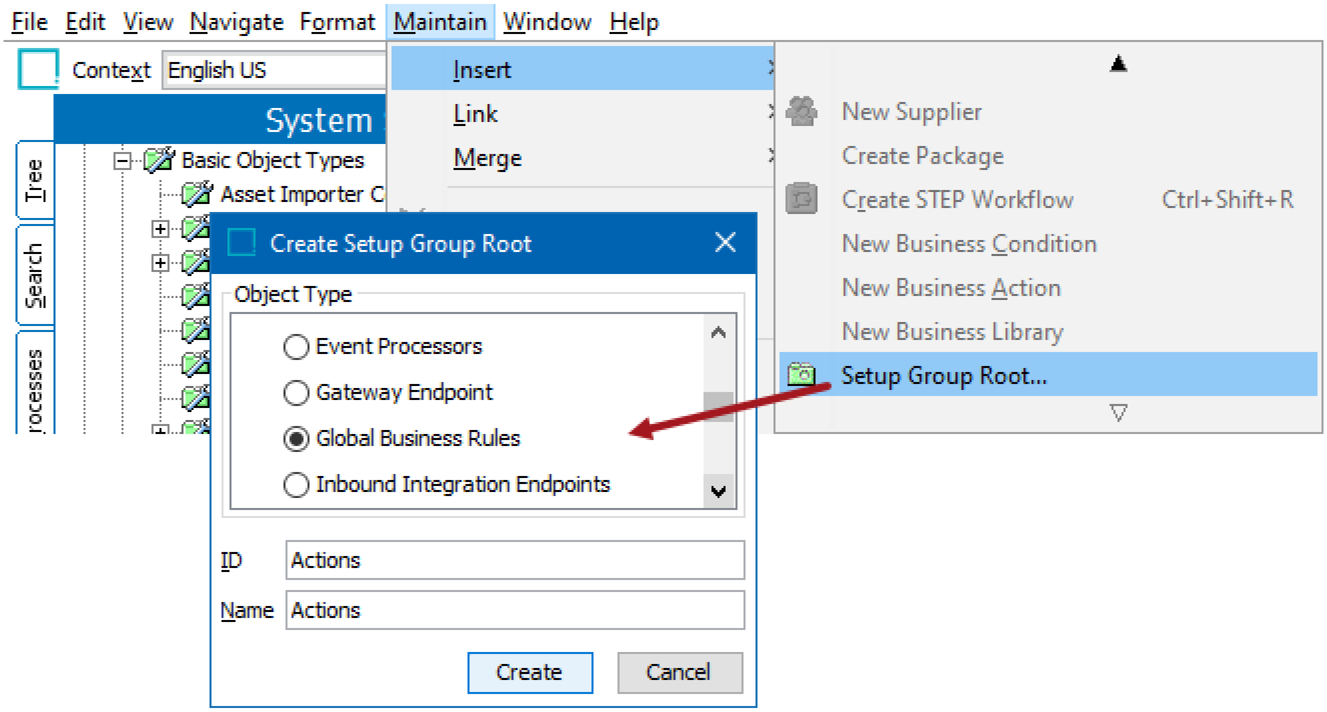
3. Enter an **ID** and a **Name** (such as Global Business Rules) and then click **Create**.
4. Optionally, if additional levels of organization are required to separate business conditions, business actions, and business libraries, right-click the object type you just created, and add the necessary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



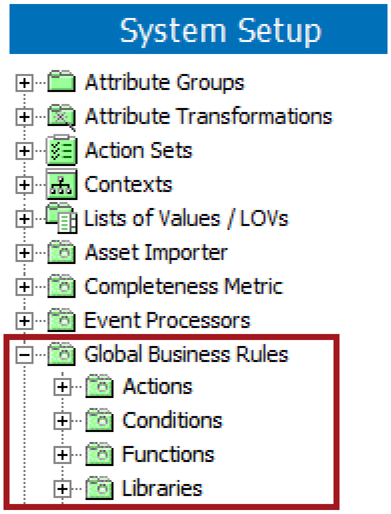
○ Event Processors
○ Gateway Endpoint
● Global Business Rules
○ Inbound Integration Endpoints

ID: Actions

Name: Actions

- 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, functions, and libraries.



System Setup

- Attribute Groups
- Attribute Transformations
- Action Sets
- Contexts
- Lists of Values / LOVs
- Asset Importer
- Completeness Metric
- Event Processors
- Global Business Rules
  - Actions
  - Conditions
  - Functions
  - Libraries

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

Creating a Business Rule, Function, or Library

After completing the one-time configuration, business actions, business conditions, business functions, and business libraries can be created in System Setup.

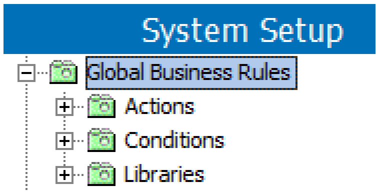
Prerequisites

Perform the required initial setup, refer to the **Initial Set Up for Business Rules** documentation.

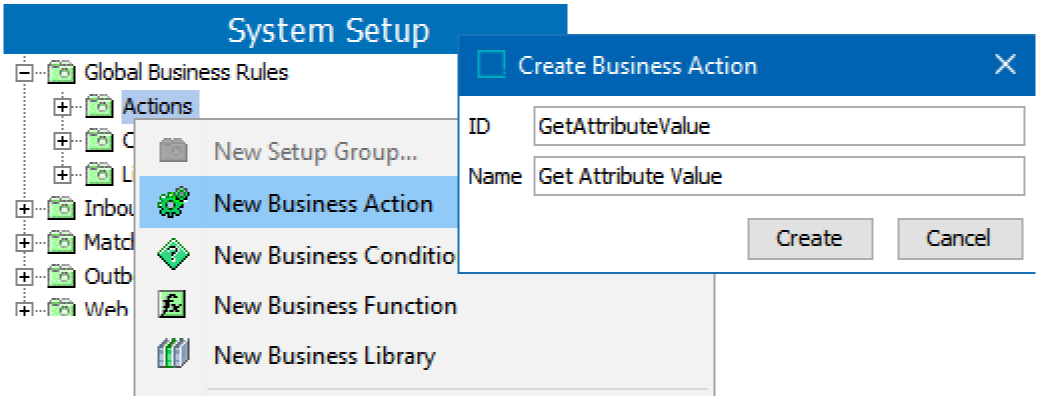
Configuration

Follow these steps to create a business action, business condition, business function, or business library:

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.



ID: GetAttributeValue
Name: Get Attribute Value

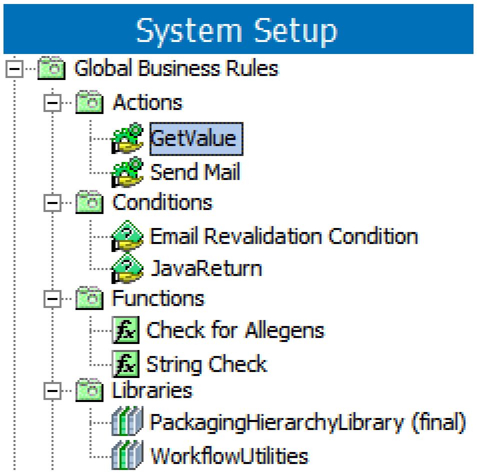
3. Enter an **ID** and a **Name** (such as Get Attribute Value) and then click the **Create** button. The selected business rule is created.

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

4. Use the appropriate step:

- To configure a new action, condition, or function, refer to the **Editing a Business Rule or Function** topic.
- To configure a new library, refer to the **Editing a Business Library** topic.

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), functions (for example, Check for Allergens), 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 or Function

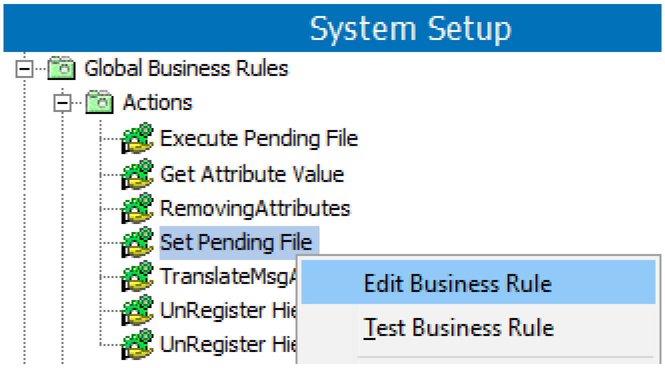
The Business Rule editor allows you to edit settings for a business rule and handles Global and Local business rules differently. For more information, refer to the **Global and Local Business Rules** section of the **Business Rules** documentation.

> **Note:** Global business rules can also be edited via workflows and uses the same global Business Rule Editor. For more information, refer to the **Business Rules and Workflows** topic in the **Workflows** documentation.

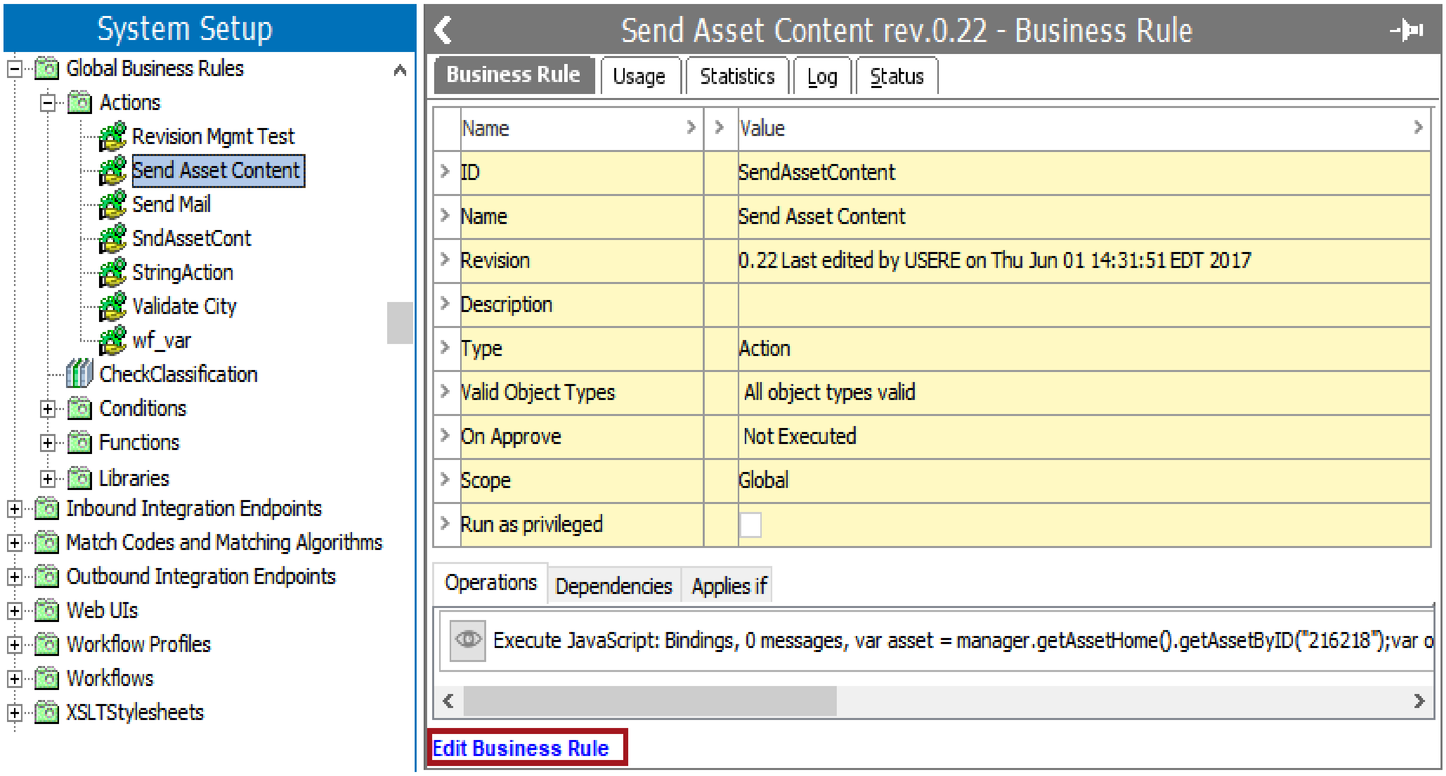
Open the Business Rule Editor

To open the Business Rule editor in System Setup:

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



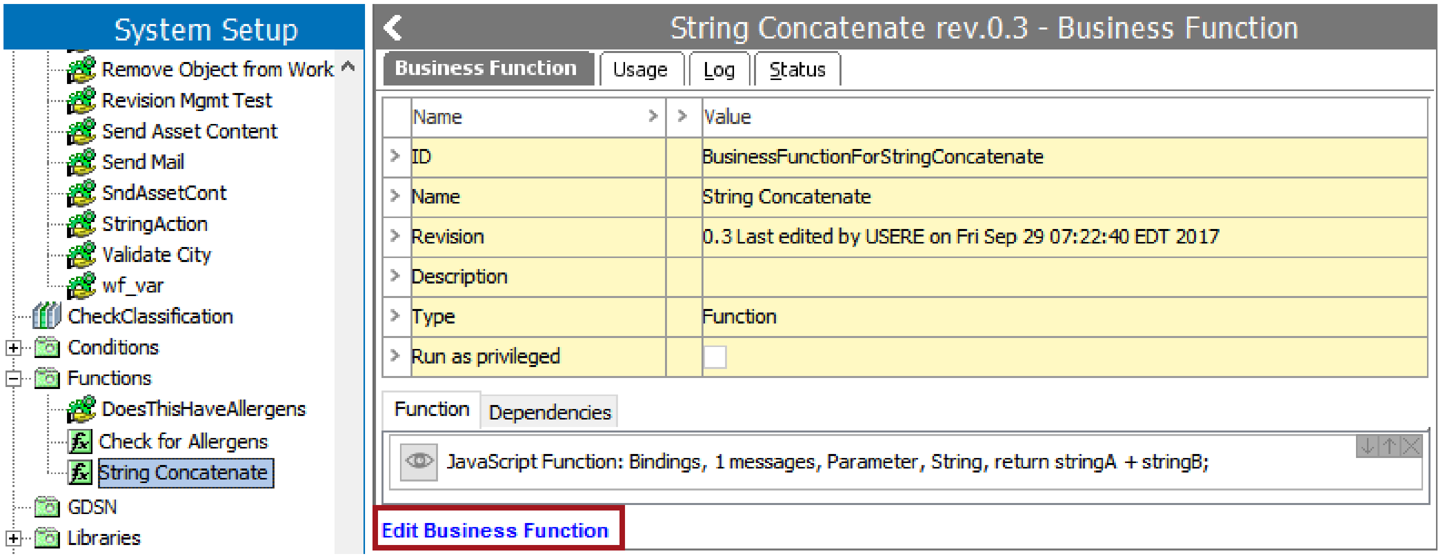
- For a business action or condition, click the **Edit Business Rule** link on the Business Rule tab.



Name	Value
ID	SendAssetContent
Name	Send Asset Content
Revision	0.22 Last edited by USERE on Thu Jun 01 14:31:51 EDT 2017
Description	
Type	Action
Valid Object Types	All object types valid
On Approve	Not Executed
Scope	Global
Run as privileged	□

Execute JavaScript: Bindings, 0 messages, var asset = manager.getAssetHome().getAssetByID("216218");var o

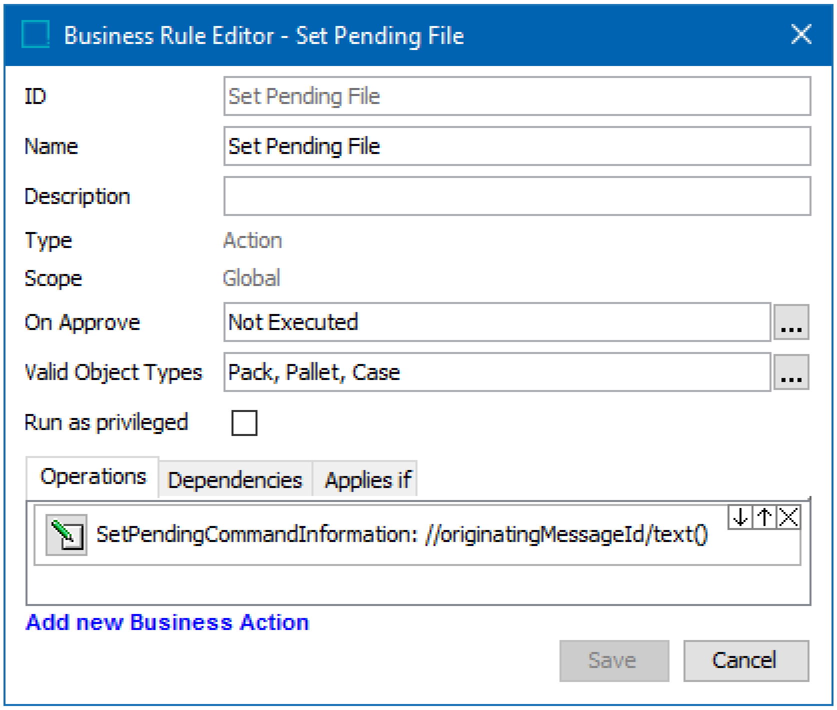
- For a business function, click the **Edit Business Function** link on the Business Rule tab.



Name	Value
ID	BusinessFunctionForStringConcatenate
Name	String Concatenate
Revision	0.3 Last edited by USERE on Fri Sep 29 07:22:40 EDT 2017
Description	
Type	Function
Run as privileged	□

JavaScript Function: Bindings, 1 messages, Parameter, String, return stringA + stringB;

For actions and conditions, the Business Rule Editor allows you to modify most parameters on a business rule. However, the ID, Type, and Scope, cannot be edited and are shown with read-only text. Additional configuration is performed on the Edit Operation dialog (accessed by clicking an edit button on the Operations tab).



Business Rule Editor - Set Pending File

ID: Set Pending File

Name: Set Pending File

Description:

Type: Action

Scope: Global

On Approve: Not Executed

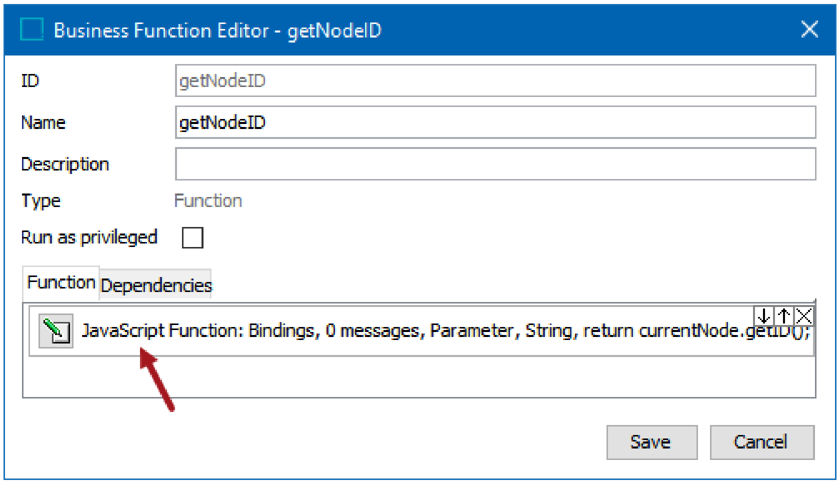
Valid Object Types: Pack, Pallet, Case

Run as privileged □

Operations | Dependencies | Applies if

SetPendingCommandInformation: //originatingMessageId/text()

For functions, the Business Function Editor includes options about the operation provided via the Query Template Builder or the JavaScript included on the Edit Operation dialog (accessed by clicking an edit button on the Function tab). For more information on setting up business functions, refer to the **Calling a Business Function from a JavaScript Business Rule** topic in the **Business Function** documentation.



Business Function Editor - getNodeID

ID: getNodeID

Name: getNodeID

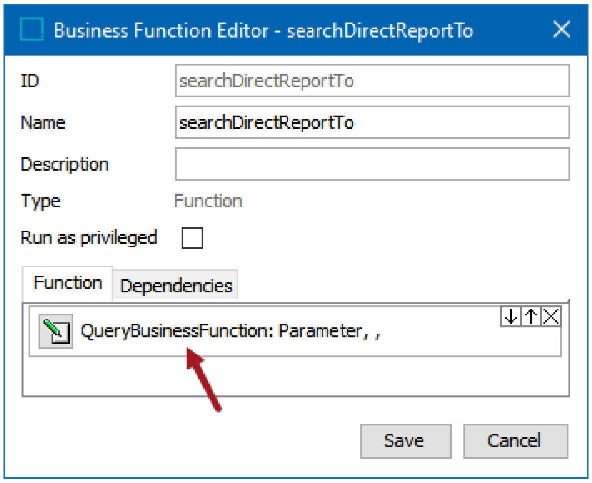
Description:

Type: Function

Run as privileged □

Function | Dependencies

JavaScript Function: Bindings, 0 messages, Parameter, String, return currentNode.getID();



Business Function Editor - searchDirectReportTo

ID: searchDirectReportTo

Name: searchDirectReportTo

Description:

Type: Function

Run as privileged □

Function | Dependencies

QueryBusinessFunction: Parameter, ,

Editor Parameters

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.

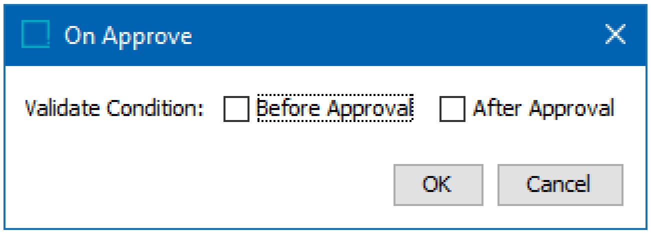
On Approve Parameter

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, refer to **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.

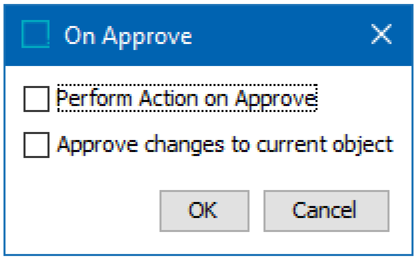
- On conditions, specify to perform the validations before or after approval.



On Approve

Validate Condition: □ Before Approval □ After Approval

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



□ Perform Action on Approve
□ Approve changes to current object

Valid Object Types Parameter

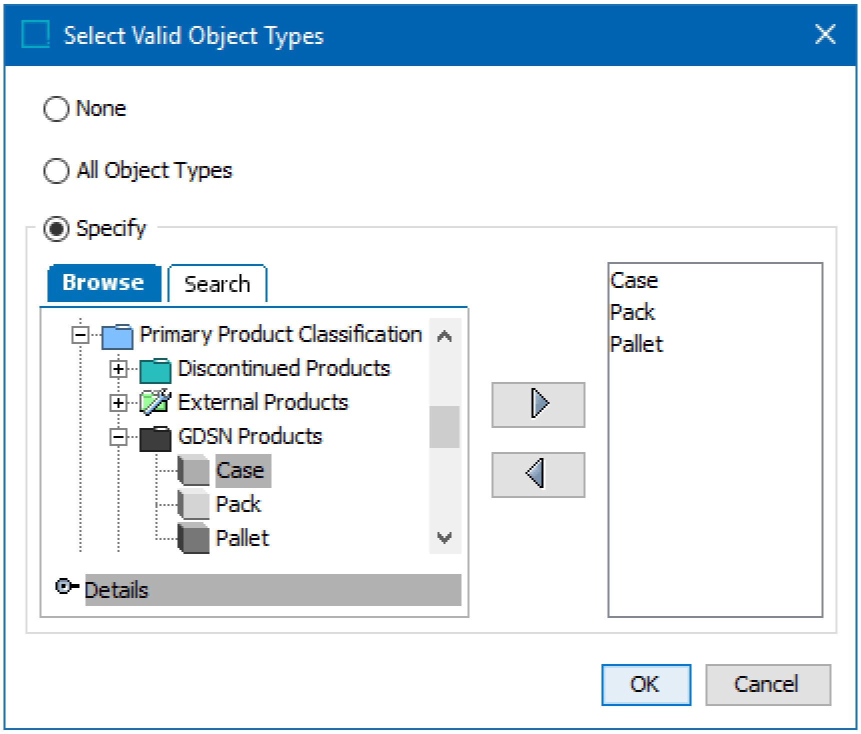
Specify the object type or types for which the business action or condition is valid. This only applies to global business rules. For local business rules, the setting of the workflow determine this parameter.

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.



Select Valid Object Types

○ None

○ All Object Types

● Specify

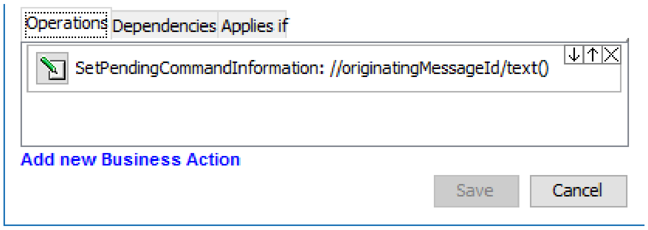
Case
Pack
Pallet

Run as Privileged Parameter

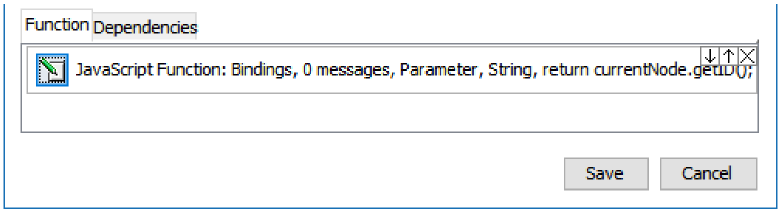
Check the **Run as privileged** checkbox to allow the business rule to execute functions that the user is not authorized to execute.

Operations or Function Tab

The Operations tab allows you to add or modify the actions or conditions that apply to a business rule. The Function tab allows you to modify the Query Template Builder of JavaScript function that applies to the business rule.



SetPendingCommandInformation: //originatingMessageId/text()



JavaScript Function: Bindings, 0 messages, Parameter, String, return currentNode.getID();

- 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, action or function, click the **Edit** () button.

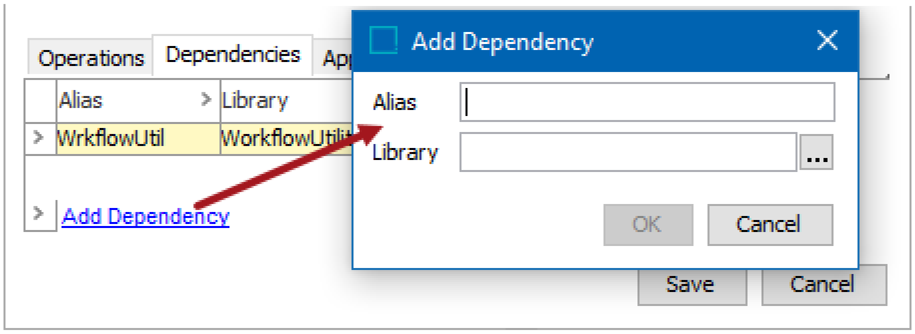
- To change the order of the business rules, use the up or down arrow buttons.
- To delete a business condition, action or function, click the **X** button.

For more information about actions, conditions, and functions, refer to 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**.

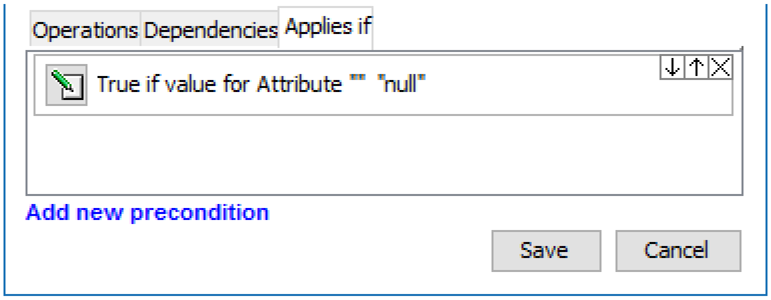


Alias	Library
WrkflowUtil	WorkflowUtil

For more information, refer to **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.



True if value for Attribute "" "null"

- 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.

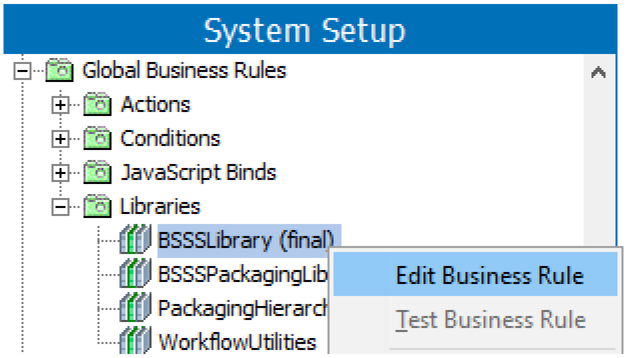
- To change the order of the conditions, click the up or down arrow buttons.
- To delete a condition, click the **X** button.

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

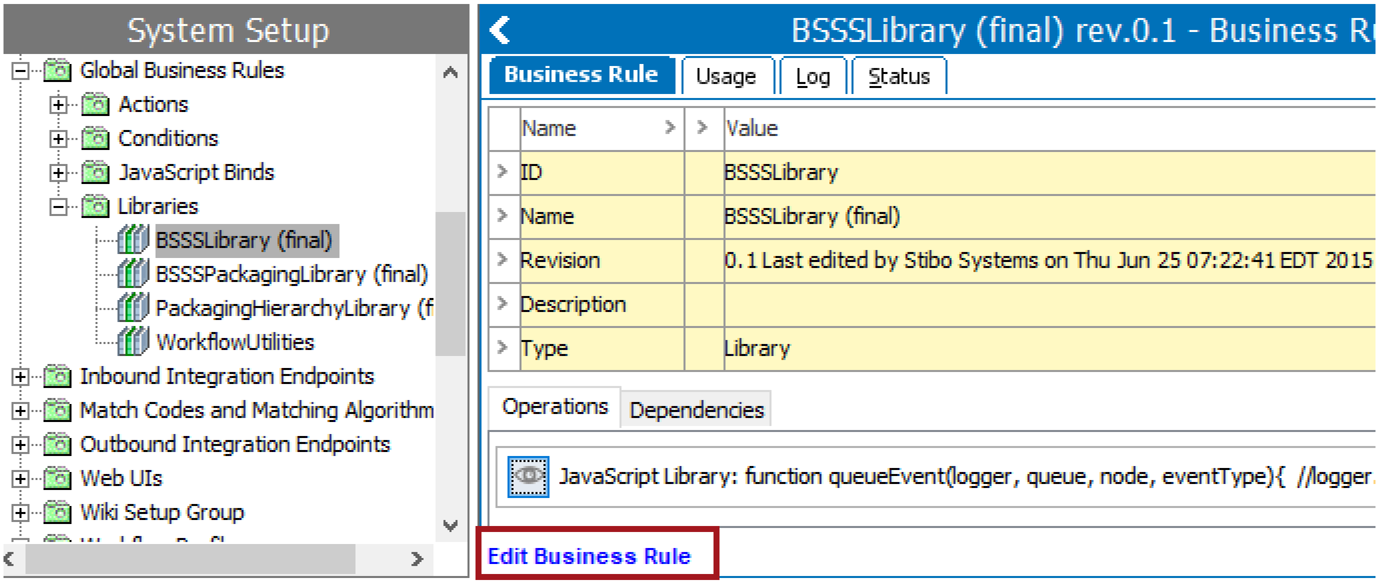
Editing a Business Library

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

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



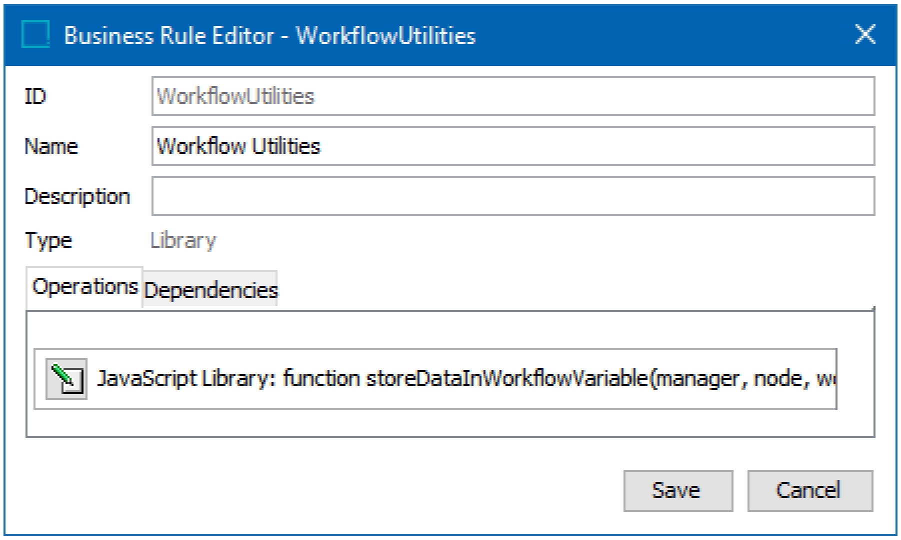
- Select the business rule and select the **Edit Business Rule** link on the Business Rule tab.



Name	Value
ID	BSSSLibrary
Name	BSSSLibrary (final)
Revision	0.1 Last edited by Stibo Systems on Thu Jun 25 07:22:41 EDT 2015
Description	
Type	Library

JavaScript Library: function queueEvent(logger, queue, node, eventType){ //logger.

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.


Business Rule Editor - WorkflowUtilities

ID	WorkflowUtilities
Name	Workflow Utilities
Description	
Type	Library

JavaScript Library: function storeDataInWorkflowVariable(manager, node, w

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

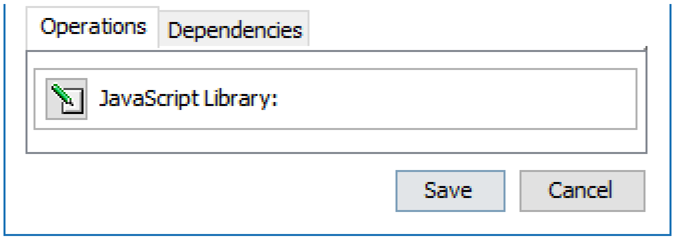
Displays the business rule description assigned while creating the business rule and can be modified.

Type Parameter

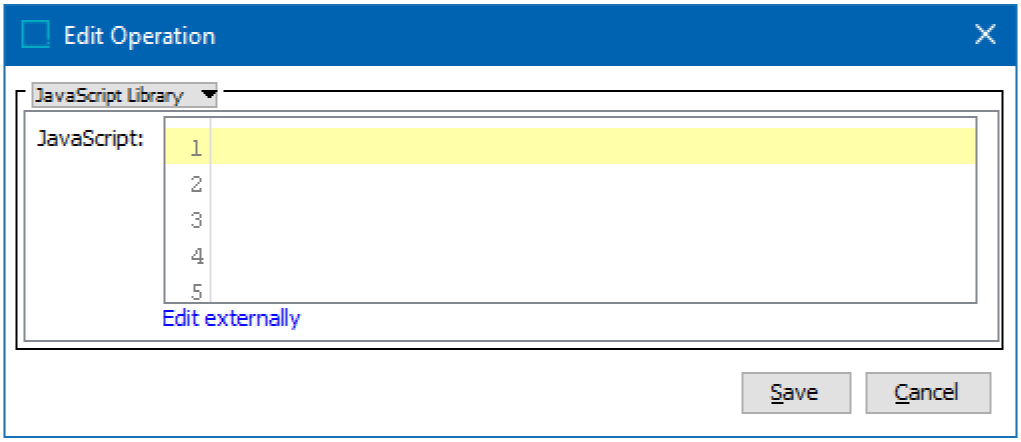
Displays the type of business rule being edited. This can be action, condition, a function, or a library.

Operations Tab

The Operations tab allows you to modify the JavaScript code for the business rule. By default, a JavaScript Library 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.

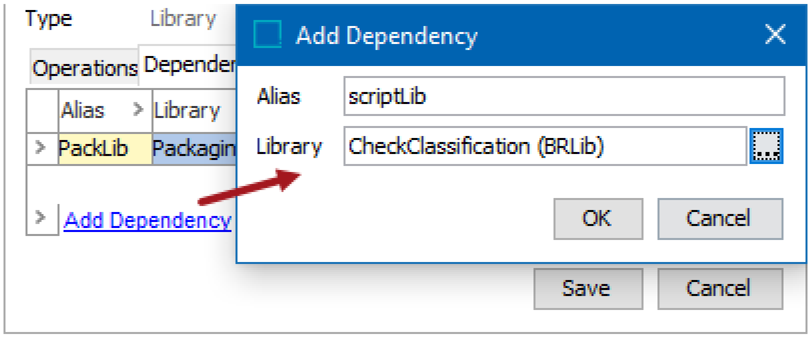


Dependencies Tab

The Dependencies tab allows you to 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 any business rule, it is important to test that it performs as expected.

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

> **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.

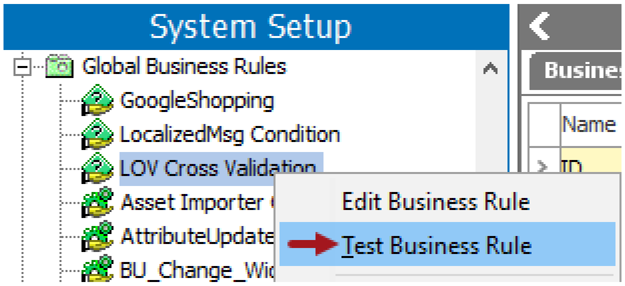
Monitor Business Rules

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 Start Page. For information on the log, refer to the **Logs** topic in the **Administration Portal** documentation.

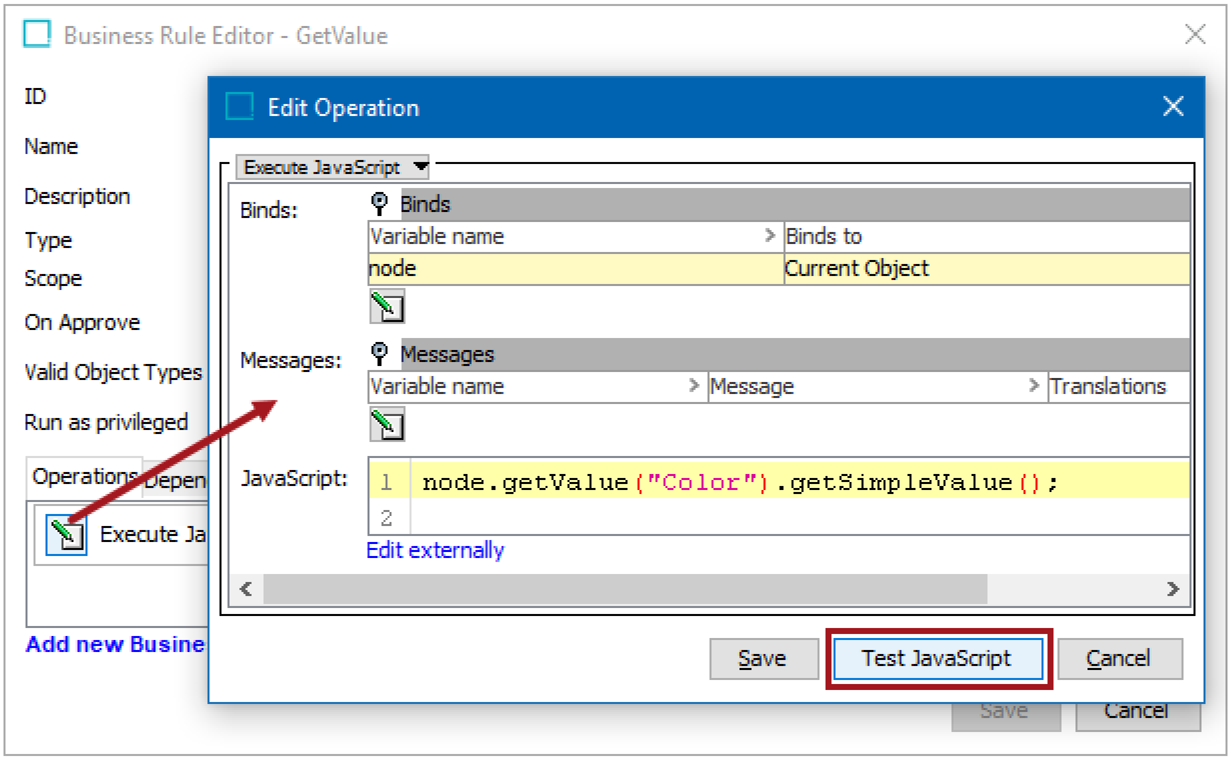
Test & Time Business Rule Dialog

Choose one of the following ways to display the Test & Time Business Rule dialog:

- In System Setup, expand the business rules node, select the business rule to test, display the right-click menu, and click **Test Business Rule**.



- While displaying the Edit Operation dialog for a JavaScript business rule, click the **Test JavaScript** button.



Binds

Variable name	Binds to
node	Current Object

Messages

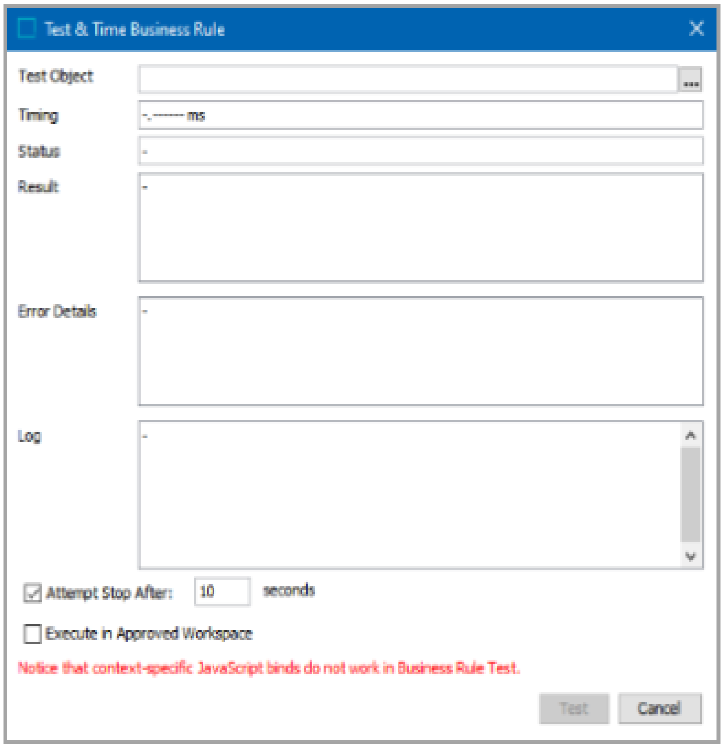
Variable name	Message	Translations

```
node.getValue("Color").getSimpleValue();
```

Click the **Test JavaScript** button to test modifications made to the JavaScript. If the modifications should be retained, click **Save**. If the modifications should be rolled back, click **Cancel**.

Perform Test & Time on a Business Rule

The Test & Time Business Rule dialog includes standard parameters for actions, conditions, and functions. In addition to the standard parameters, testing action business rules includes the 'Rollback changes after test' parameter.



Test & Time Business Rule

Test Object ...

Timing -,------ ms

Status -

Result -

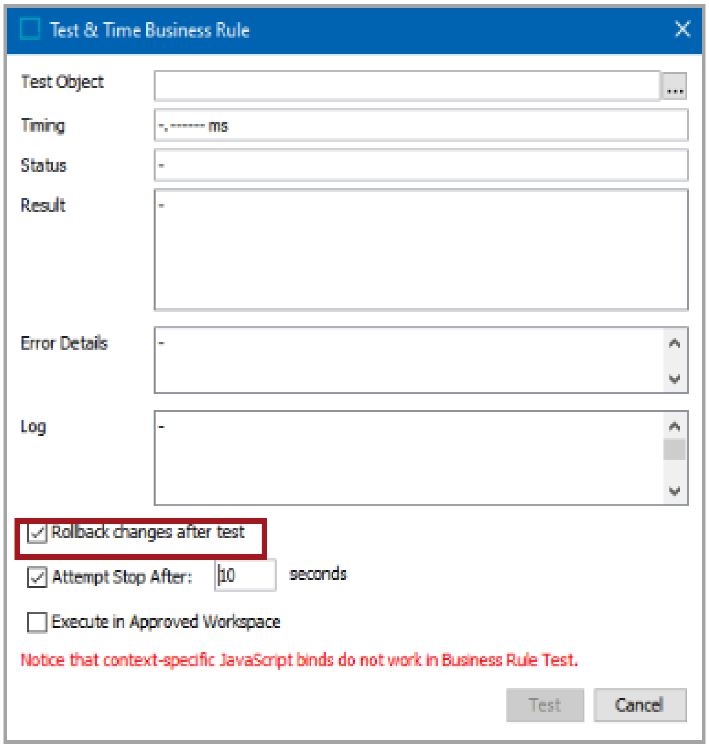
Error Details -

Log -

☑ Attempt Stop After: 10 seconds

☐ Execute in Approved Workspace

Notice that context-specific JavaScript binds do not work in Business Rule Test.



Test & Time Business Rule

Test Object ...

Timing -,------ ms

Status -

Result -

Error Details -

Log -

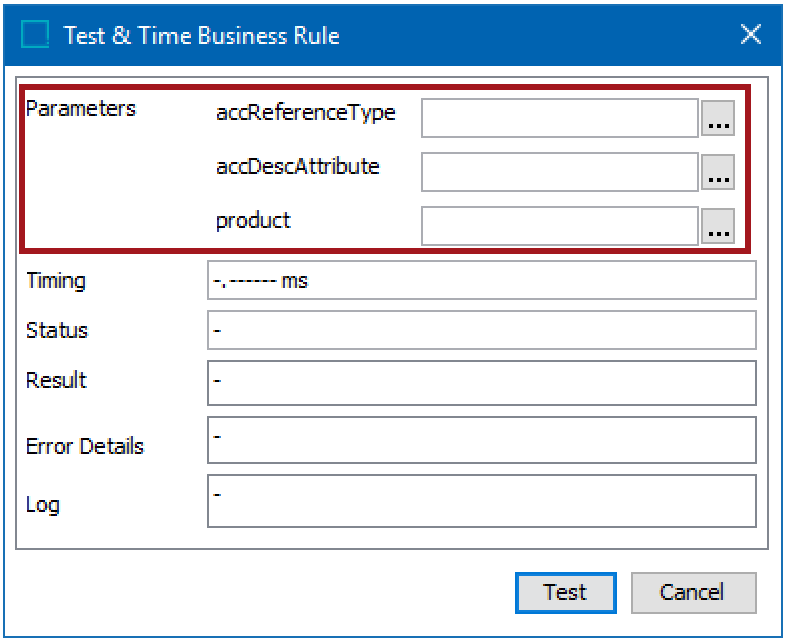
☑ Rollback changes after test

☑ Attempt Stop After: 10 seconds

☐ Execute in Approved Workspace

Notice that context-specific JavaScript binds do not work in Business Rule Test.

In addition to the standard parameters, testing function business rules includes a Parameters section that shows each of the input parameters defined in the business function.



Test & Time Business Rule

Parameters
accReferenceType ...
accDescAttribute ...
product ...

Timing -,------ ms

Status -

Result -

Error Details -

Log -

After displaying the dialog, supply the following information to test a business rule.

1. For actions or conditions, 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.

   > **Note:** The selected object must be of an object type that is valid for the business rule.

2. For functions only, in the **Parameters** section, click the ellipsis button (...) to display the appropriate selection dialog required for the Input Parameters available in the function. Use the Browse or Search tab to identify and select each input object for testing.
3. For actions only, the **Rollback changes after test** checkbox is available. When checked, any changes caused by the action are rolled back after the action has been executed.
4. The **Attempt Stop After: __ Seconds** is checked by default, with the numeric value of ten in the editable field. When checked, if the business rule test does not complete within the number of seconds specified, it is terminated, if possible. This prevents an infinite loop or other unintended long running business rules. The maximum value for the field is 600 seconds. If the business rule test exceeds the limit, an error stating 'Javascript terminated as its execution time exceeded timeout of X seconds,' where the 'X' is the number of seconds in the field.
5. The **Execute in Approved Workspace** is unchecked by default and the test is performed in the Main Workspace. When checked, the action is performed on the selected object in the Approved Workspace.

   Since business functions cannot change data in any workspace, this parameter is not available for business functions. Testing a business function is performed in the current workspace.

6. Click the **Test** button.

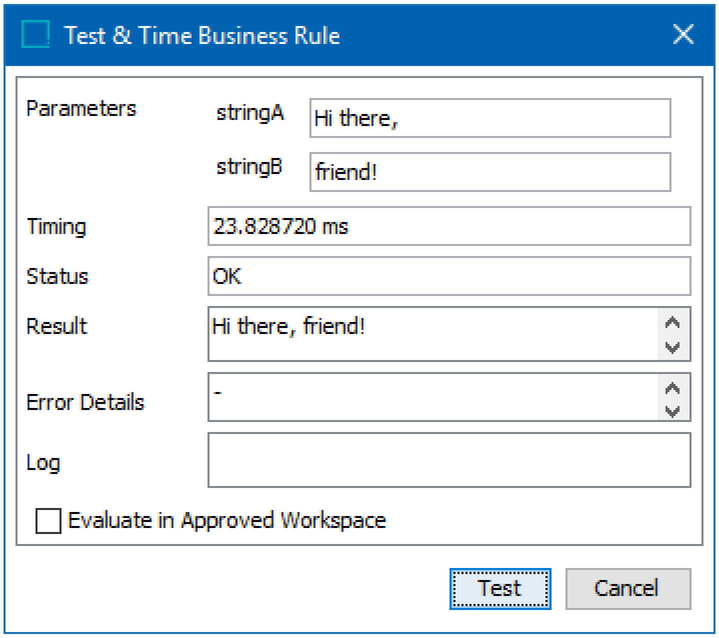
Test & Time Business Rule Results

The following parameter values are supplied when the test is complete.

- **Timing** - Displays the time taken to execute the business rule on the selected node.
- **Status** - Displays **OK** (text succeeded), **Not Applicable** (object type tested is not valid), or **Error** as the outcome of the business rule test.
- **Result** - For conditions only, displays the result of the test (**true** or **false**) and any localized messages. For more information, refer to the **Localized Messages for JavaScript Business Rules** topic.
- **Error Details** - Displays additional information on an error result, including the stacktrace when available.
- **Log** - Displays the log message for debug and/or testing purpose. In JavaScript, use Logger in the code to log a message, as described in the **Logger Bind** topic in the **Resource Materials** online help.

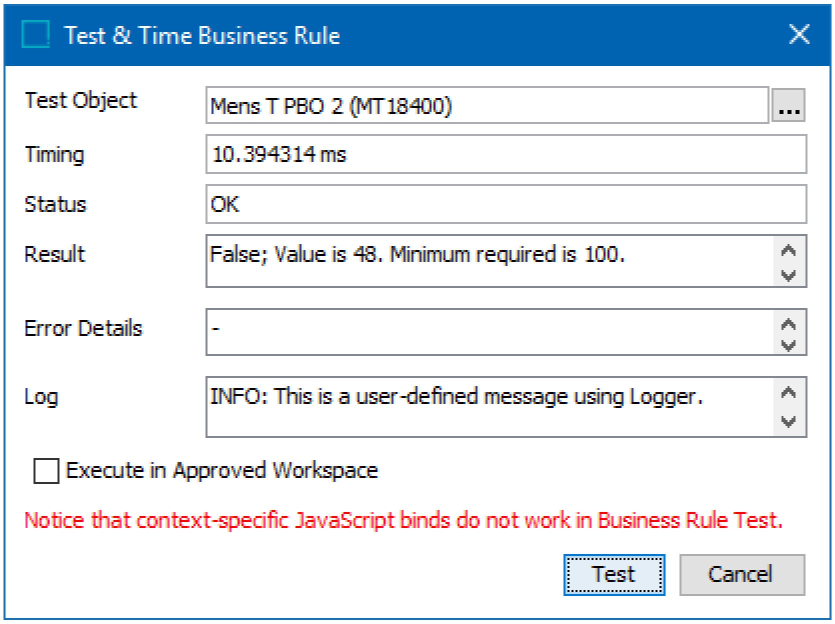
Testing Outcome Examples

- **OK** - This status is displayed when the test is executed successfully. In the following example, a function that concatenates two input values is tested. The outcome of the function is displayed in the Result parameter. For another testing scenario including a function, refer to the **Calling a Business Function from a JavaScript Business Rule** topic.



Parameters stringA: Hi there,
stringB: friend!
Timing: 23.828720 ms
Status: OK
Result: Hi there, friend!
Error Details: -
Log:
□ Evaluate in Approved Workspace

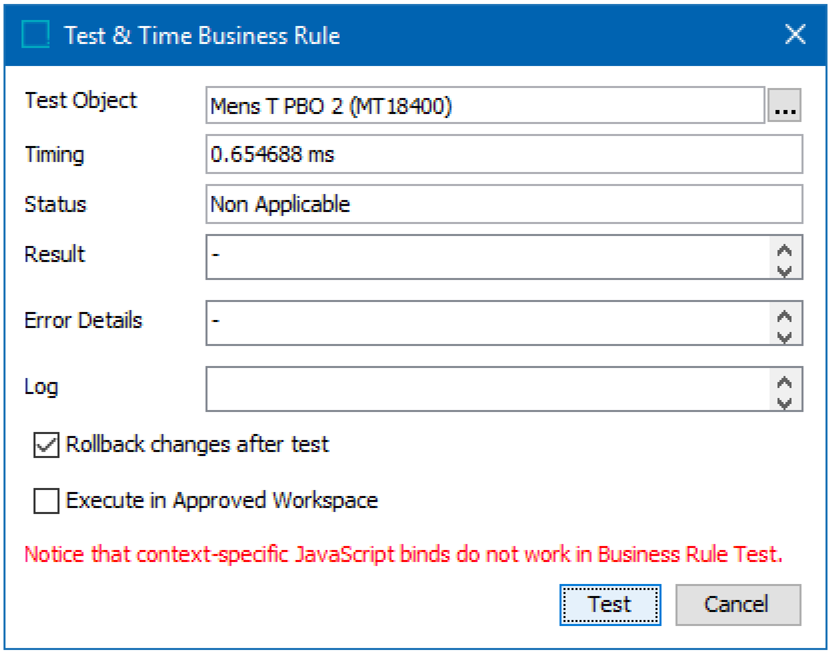
Testing the following business condition includes the outcome of the condition and the localized message in the Result parameter. A Logger message is displayed in the Log parameter.



Test Object: Mens T PBO 2 (MT18400)
Timing: 10.394314 ms
Status: OK
Result: False; Value is 48. Minimum required is 100.
Error Details: -
Log: INFO: This is a user-defined message using Logger.
□ Execute in Approved Workspace

Notice that context-specific JavaScript binds do not work in Business Rule Test.

- **Non Applicable -** This status is displayed when the test object does not match the Valid Object Types on the business rule, as illustrated in the action business rule test image below.



Test & Time Business Rule

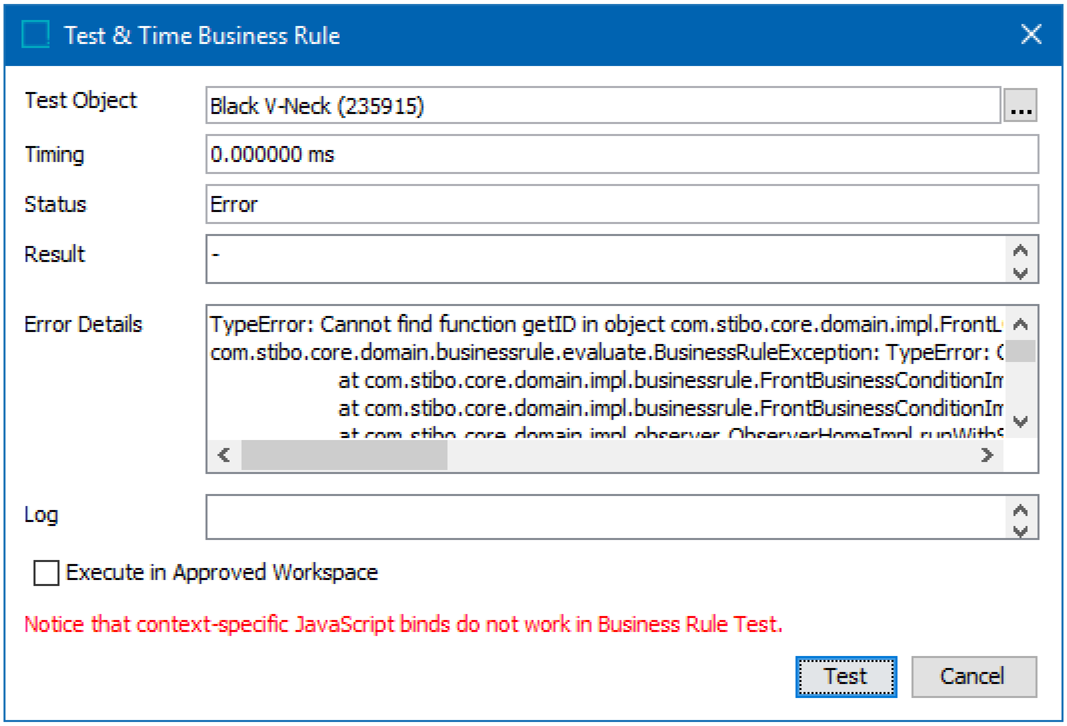
Test Object	Mens T PBO 2 (MT18400)
Timing	0.654688 ms
Status	Non Applicable
Result	-
Error Details	-
Log	

☑ Rollback changes after test

□ Execute in Approved Workspace

Notice that context-specific JavaScript binds do not work in Business Rule Test.

- **Error -** This status is displayed when the business rule has invalid methods, syntax, or function. For the following condition test, the Error Details parameter includes stacktrace and troubleshooting information.



Test & Time Business Rule

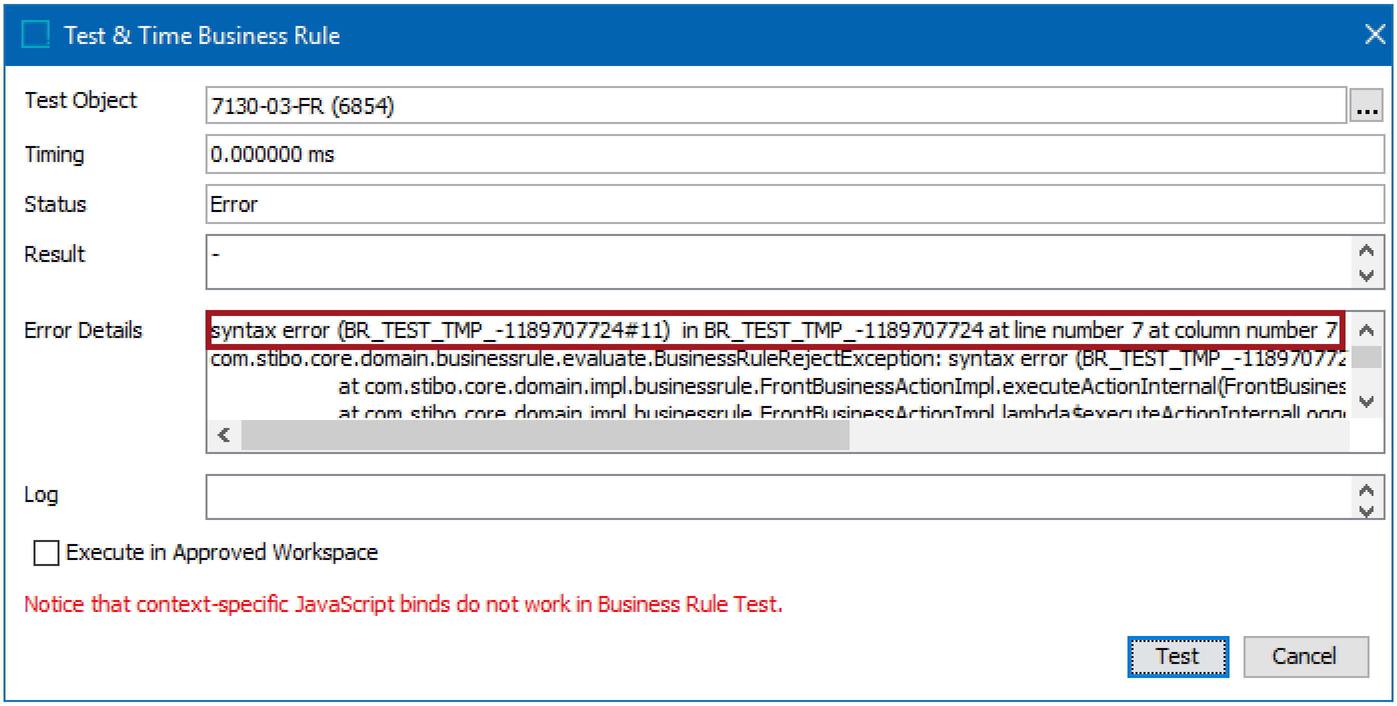
Test Object	Black V-Neck (235915)
Timing	0.000000 ms
Status	Error
Result	-
Error Details	see below
Log	

```
TypeError: Cannot find function getID in object com.stibo.core.domain.impl.FrontL
com.stibo.core.domain.businessrule.evaluate.BusinessRuleException: TypeError: (
                at com.stibo.core.domain.impl.businessrule.FrontBusinessConditionIm
                at com.stibo.core.domain.impl.businessrule.FrontBusinessConditionIm
                at com.stibo.core.domain.impl.observer.ObserverHomeImpl.runWithS
```

□ Execute in Approved Workspace

Notice that context-specific JavaScript binds do not work in Business Rule Test.

The following business rule has a problem in the JavaScript code as indicated by the row number in the error details parameter.



Test Object: 7130-03-FR (6854)

Timing: 0.000000 ms

Status: Error

Result: -

Error Details:

```
syntax error (BR_TEST_TMP_-1189707724#11)  in BR_TEST_TMP_-1189707724 at line number 7 at column number 7
com.stibo.core.domain.businessrule.evaluate.BusinessRuleRejectException: syntax error (BR_TEST_TMP_-11897077
                at com.stibo.core.domain.impl.businessrule.FrontBusinessActionImpl.executeActionInternal(FrontBusines
                at com.stibo.core.domain.impl.businessrule.FrontBusinessActionImpl.lambda$executeActionInternal
```

Log:

□ Execute in Approved Workspace

Notice that context-specific JavaScript binds do not work in Business Rule Test.

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, refer to 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.

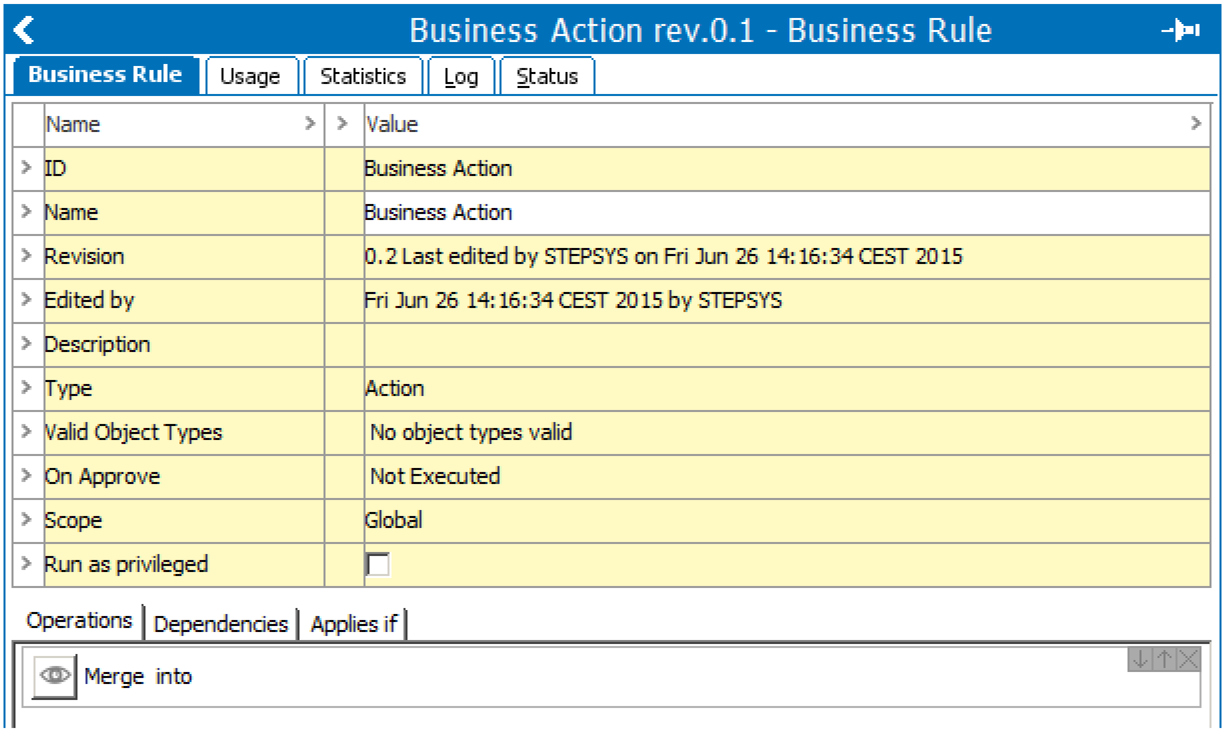
> **Note:** Business rules viewed in the approved workspace are read-only. Business rules can only be edited from the main workspace.

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

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

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

Business Rule Tab



Business Action rev.0.1 - Business Rule

Name	Value
ID	Business Action
Name	Business Action
Revision	0.2 Last edited by STEPSYS on Fri Jun 26 14:16:34 CEST 2015
Edited by	Fri Jun 26 14:16:34 CEST 2015 by STEPSYS
Description	
Type	Action
Valid Object Types	No object types valid
On Approve	Not Executed
Scope	Global
Run as privileged	□

Merge into

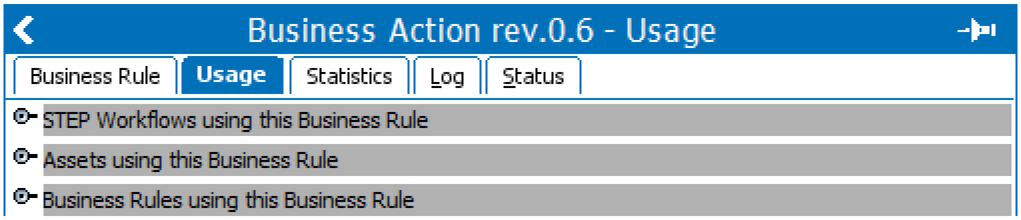
The Business Rule tab displays the following information:

- **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, refer to 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, refer to 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, refer to the **Business Rules** section of the **Setup Actions and Error Descriptions** topic within the **System Setup** 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, refer to **Editing a Business Rule or Function**.

Usage Tab

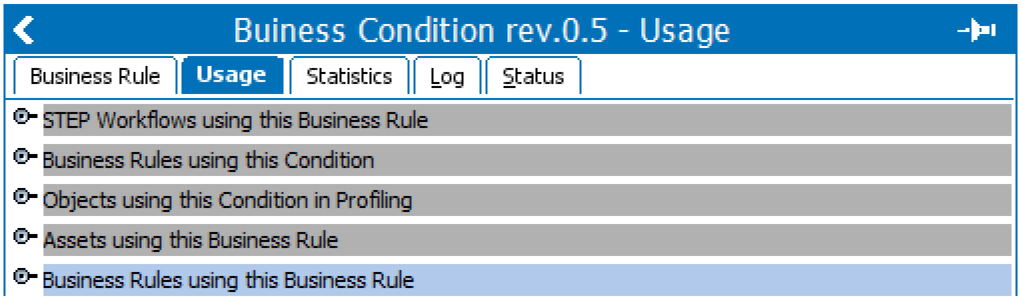
The Usage tab displays information about where the business rule is used. The available flippers are determined by the type of business rule displayed, as shown in the following images.



STEP Workflows using this Business Rule

Assets using this Business Rule

Business Rules using this Business Rule



STEP Workflows using this Business Rule

Business Rules using this Condition

Objects using this Condition in Profiling

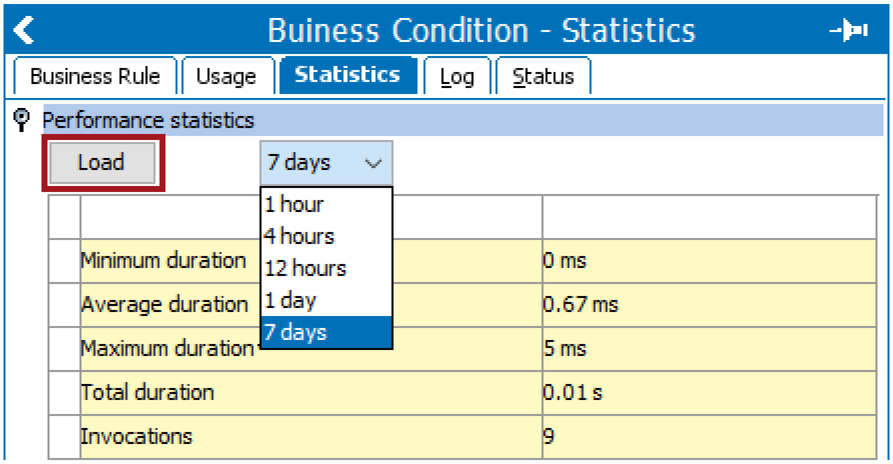
Assets using this Business Rule

Business Rules using this Business Rule

Statistics Tab

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

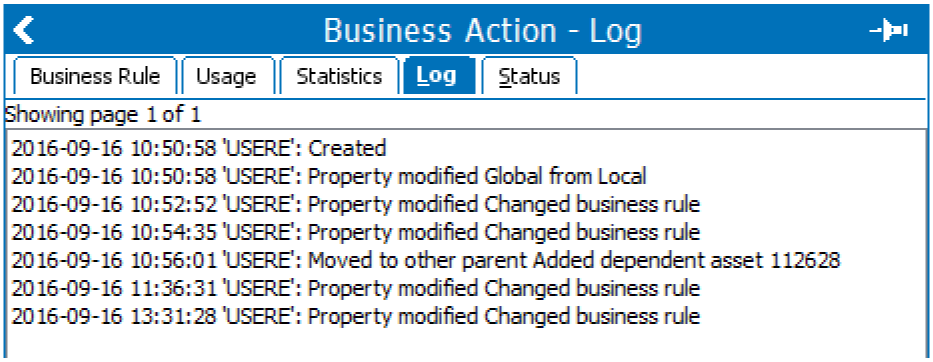
To view the statistics, select a time interval from the dropdown, and then click the **Load** button.



Minimum duration	0 ms
Average duration	0.67 ms
Maximum duration	5 ms
Total duration	0.01 s
Invocations	9

Log Tab

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

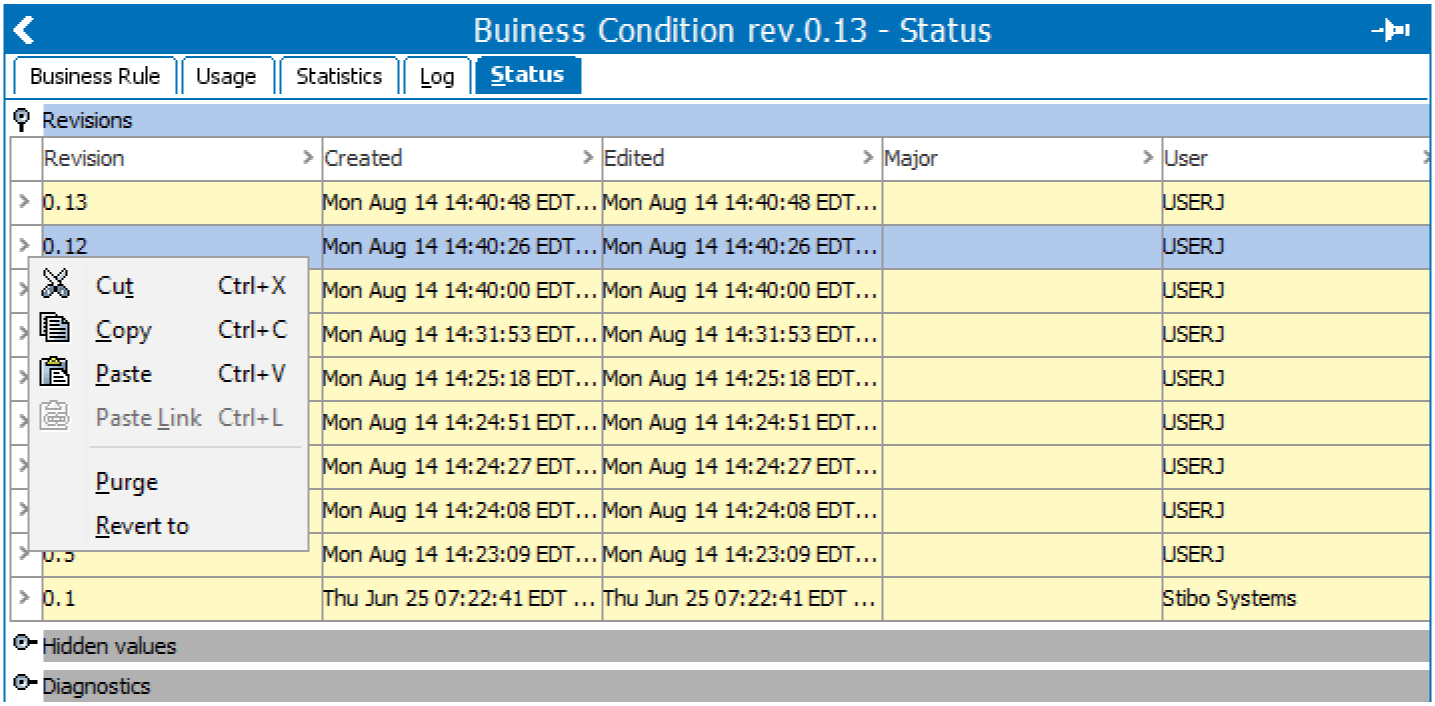


Showing page 1 of 1

```
2016-09-16 10:50:58 'USERE': Created
2016-09-16 10:50:58 'USERE': Property modified Global from Local
2016-09-16 10:52:52 'USERE': Property modified Changed business rule
2016-09-16 10:54:35 'USERE': Property modified Changed business rule
2016-09-16 10:56:01 'USERE': Moved to other parent Added dependent asset 112628
2016-09-16 11:36:31 'USERE': Property modified Changed business rule
2016-09-16 13:31:28 'USERE': Property modified Changed business rule
```

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.



Revision	Created	Edited	Major	User
0.13	Mon Aug 14 14:40:48 EDT...	Mon Aug 14 14:40:48 EDT...		USERJ
0.12	Mon Aug 14 14:40:26 EDT...	Mon Aug 14 14:40:26 EDT...		USERJ
	Mon Aug 14 14:40:00 EDT...	Mon Aug 14 14:40:00 EDT...		USERJ
	Mon Aug 14 14:31:53 EDT...	Mon Aug 14 14:31:53 EDT...		USERJ
	Mon Aug 14 14:25:18 EDT...	Mon Aug 14 14:25:18 EDT...		USERJ
	Mon Aug 14 14:24:51 EDT...	Mon Aug 14 14:24:51 EDT...		USERJ
	Mon Aug 14 14:24:27 EDT...	Mon Aug 14 14:24:27 EDT...		USERJ
	Mon Aug 14 14:24:08 EDT...	Mon Aug 14 14:24:08 EDT...		USERJ
[illegible]	Mon Aug 14 14:23:09 EDT...	Mon Aug 14 14:23:09 EDT...		USERJ
0.1	Thu Jun 25 07:22:41 EDT ...	Thu Jun 25 07:22:41 EDT ...		Stibo Systems

- **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, refer to the **Managing Workflows** section of the **Workflows** documentation.

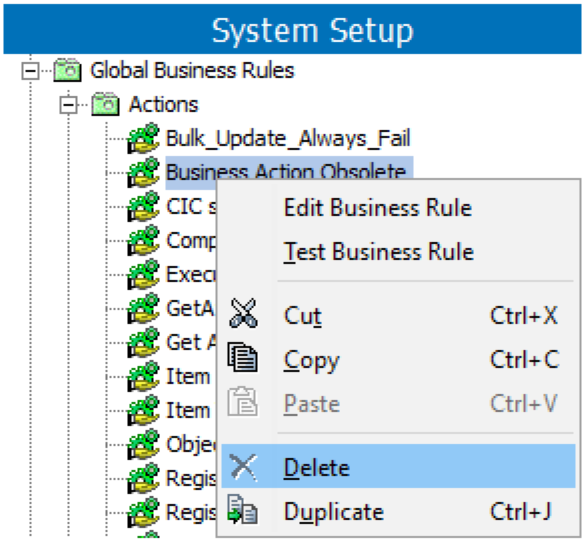
- To delete a revision, select the desired row, right-click the > column, and click the **Purge** option from the menu.

For detail about creating revisions, refer to the **Revisions** topic within **System Setup** documentation.

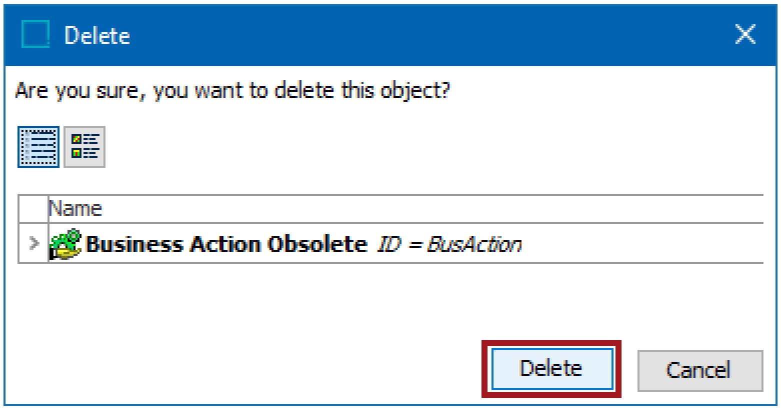
Deleting a Business Rule, Function, or Library

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

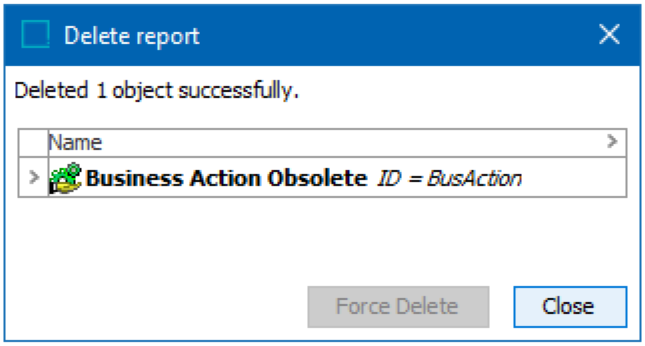
1. On System Setup, select the business rule or library to be deleted.
2. Right-click to display the menu and click the **Delete** option.



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


Delete
Are you sure, you want to delete this object?
Name
Business Action Obsolete ID = BusAction
Delete
Cancel

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

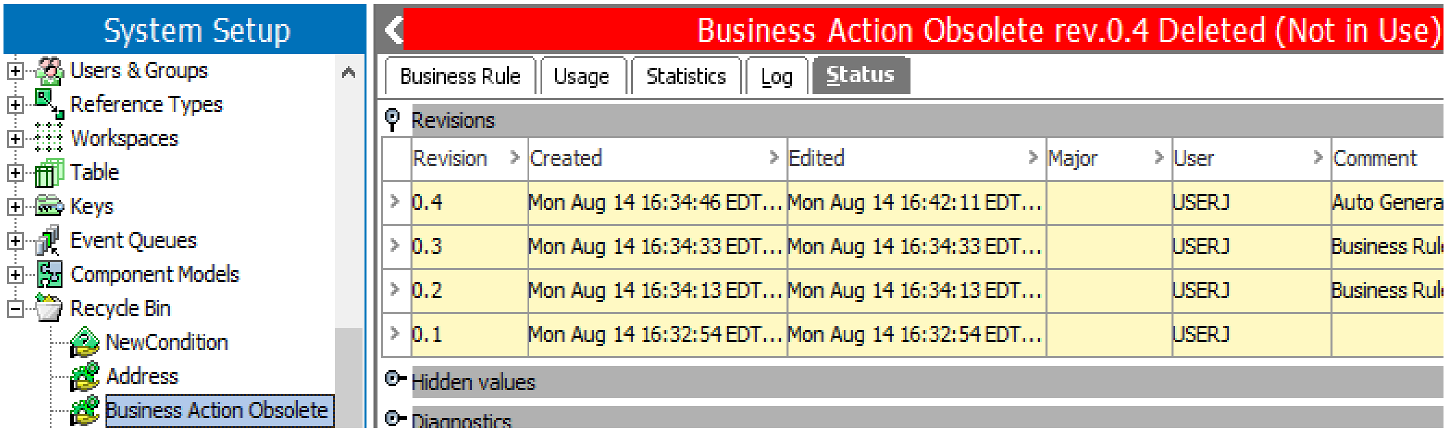


Delete report

Deleted 1 object successfully.

Name
Business Action Obsolete *ID = BusAction*

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, refer to the **Recycle Bin for System Setup** topic.



Business Action Obsolete rev.0.4 Deleted (Not in Use)

Revisions

Revision	Created	Edited	Major	User	Comment
0.4	Mon Aug 14 16:34:46 EDT…	Mon Aug 14 16:42:11 EDT…		USERJ	Auto Genera
0.3	Mon Aug 14 16:34:33 EDT…	Mon Aug 14 16:34:33 EDT…		USERJ	Business Rul
0.2	Mon Aug 14 16:34:13 EDT…	Mon Aug 14 16:34:13 EDT…		USERJ	Business Rul
0.1	Mon Aug 14 16:32:54 EDT…	Mon Aug 14 16:32:54 EDT…		USERJ	

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:

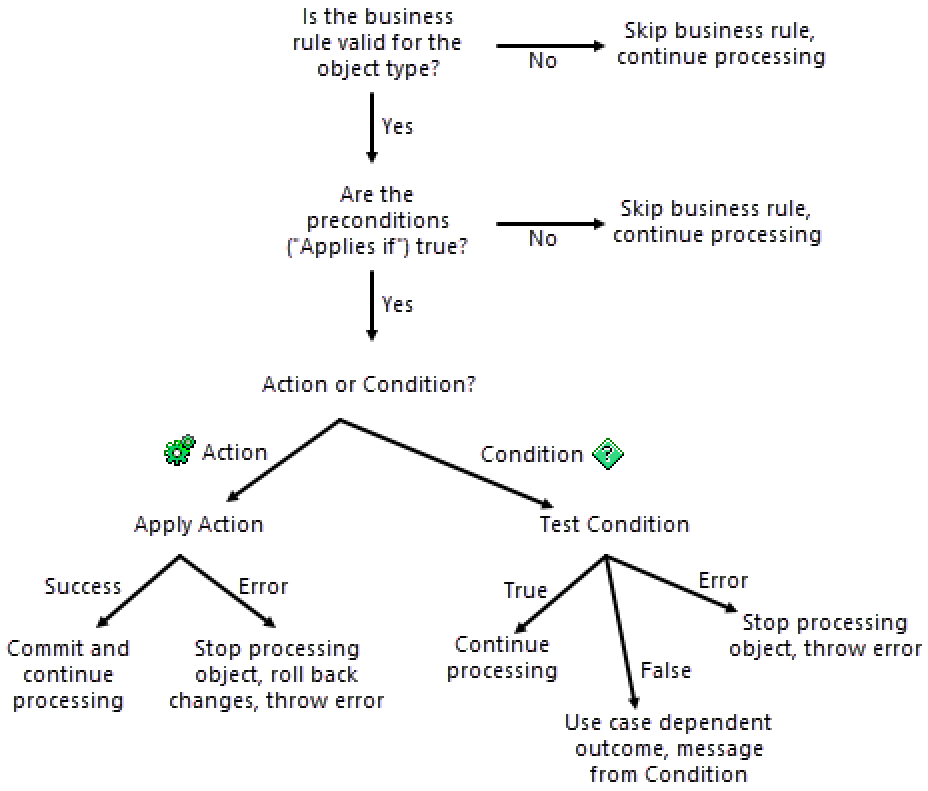
```
<BusinessRule...>
<!--Definition:[This will be the business rule definition. Removed for
brevity.-->
[Remaining configuration is included]
</BusinessRule>
```

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, refer to 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.



Is the business rule valid for the object type?
No
Skip business rule, continue processing
Yes
Are the preconditions ("Applies if") true?
No
Skip business rule, continue processing
Yes
Action or Condition?
Action
Condition
Apply Action
Test Condition
Success
Error
Commit and continue processing
Stop processing object, roll back changes, throw error
True
Error
Stop processing object, throw error
Continue processing
False
Use case dependent outcome, message from Condition

For more details and examples of test results, refer to 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, refer to **Business Conditions**. Alternatively, business functions will produce an output based on non-changing input parameters. For more information on business functions, refer to the **Business Function** topic.

Error Messages

Exceptions can be thrown during business rule execution, for instance by deliberately instantiating and throwing a localizable message, as defined in the **Adding a Localized Business Rule Message** topic. 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, refer to the **Specifying a Business Action Operation** topic.

Menu	Action	Description
Assets	**Create Assets from URL**	Works on a product level and looks at a specified attribute which contains asset file URLs. If a file URL is found, the asset is downloaded, created, and an asset reference is created according to the configuration. For more information, refer to **Configuring the Asset Download Component** in the **Digital Assets** documentation.

Menu	Action	Description
	Send Derived Event for Assets	Raises the Asset Keywords Event. Send assets to the Google Cloud Vision API and returns keywords, but does not set them. For more information, refer to **Configuring the Asset Analyzer Component** in the **Digital Assets** documentation.
	Set Asset Keywords	Handles the communication to and from the Google Vision API. Send assets to the Google Cloud Vision API, then and returns and sets keywords. For more information, refer to **Configuring the Asset Analyzer Component** in the **Digital Assets** documentation.
Attribute values	**Merge Attribute Values**	Merges the value of one attribute into another attribute. For more information, refer to **Business Action: Merge Attribute Values**.
Data quality	**Generate Match Codes**	Generates match code values as a part of a business action. For more information, refer to **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 older than a specified number of days, and enable CASS validation for US addresses. For more information, refer to **Business Action: Standardize Address**.
References and Links	**Add Attribute Link**	Adds a link to an attribute without additional configuration. For more information, refer to **Business Action: Add Attribute Link**.
	Add Reference	Adds a reference of a specified type to a target. For more information, refer to **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, refer to **Business Action: Add Referenced By**.
	Automatic Classification	Classifies the selected objects automatically according to a set of rules. For more information, refer to **Business Action: Automatic Classification**.
	Remove Attribute Link	Makes it possible to remove a link to an attribute without additional configuration. For more information, refer to **Business Action: Remove Attribute Link**.

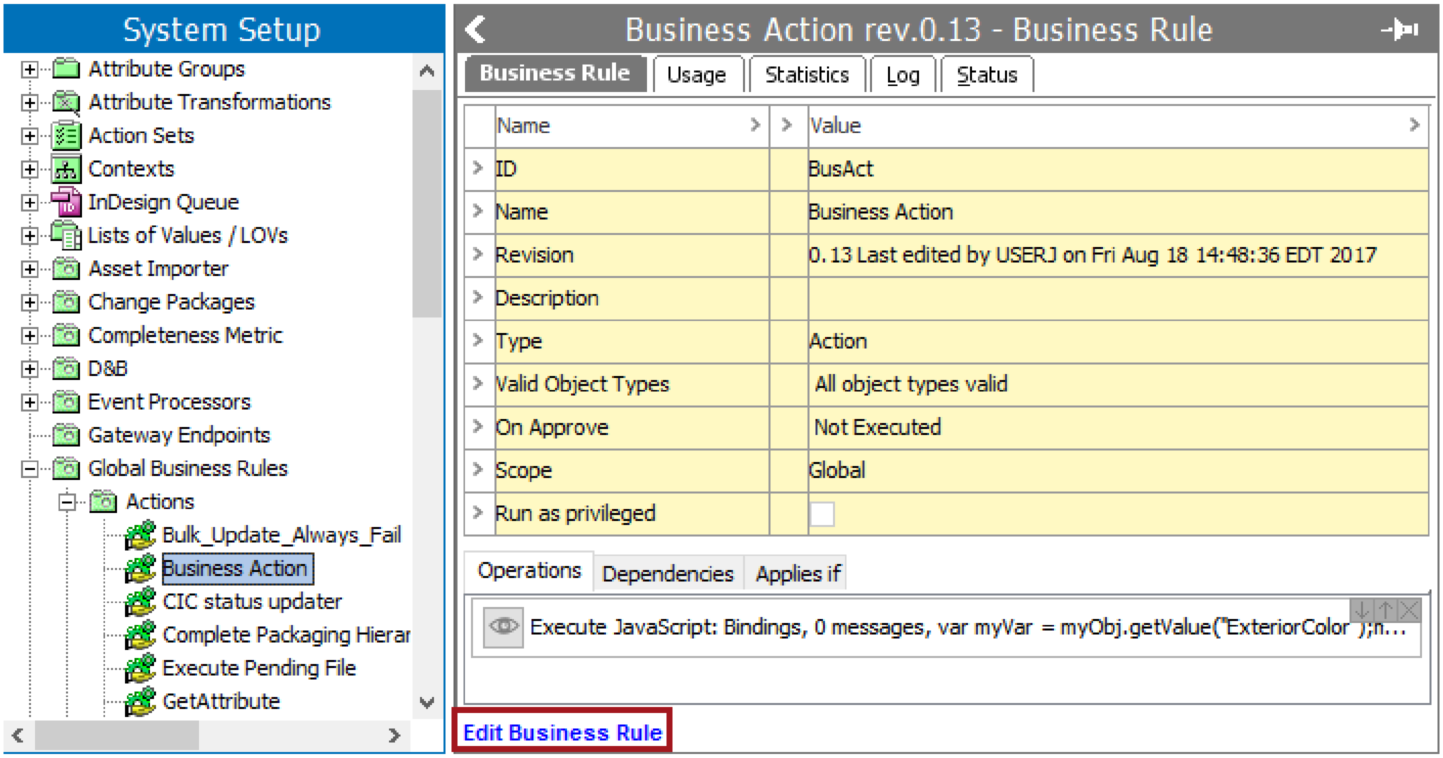
Menu	Action	Description
	Remove Reference	Removes references to a specific target or all references of a specific type. For more information, refer to **Business Action: Remove Reference**.
	Set Product to Classification Link Type	Assigns a new object type and classification link type to the selected product. For more information, refer to **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, refer to **Business Action: Claim**.
	Initiate Items in STEP Workflow	Initiates the object the action runs on in a specified workflow. For more information, refer to **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, refer to **Business Action: Remove Object from STEP Workflow**.
	Trigger STEP Workflow Event	Triggers a specified workflow event. For more information, refer to **Business Action: Trigger STEP Workflow Event**.
Execute JavaScript		Enables you to use the methods available in the STEP Public Java API. For more information, refer to **Business Action: 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, refer to **Business Action: Overlap Analysis**.
Reference Other Business Action		Allows the node to reference other business actions without additional configuration. For more information, refer to **Business Action: Reference Other Business Action**.
Send Email		Sends email to specified recipients. For more information, refer to **Business Action: Send Email**.
Send Republish Event		Sends a republish event to an event queue processor. For more

Menu	Action	Description
		information, refer to **Business Action: Send Republish Event**.
Set Attribute Value		Sets a specified attribute value for the object where the action is executed. For more information, refer to **Business Action: Set Attribute Value**.
Set Name		Sets the name of the object that the action runs on. For more information, refer to **Business Action: Set Name**.
Set Object Type		Changes the object type the action runs on. For more information, refer to **Business Action: Set Object Type**.
Set Workflow Variable		Enables assignment of workflow variables. For more information, refer to **Business Action: Set Workflow Variable**.
Start Translation		Initiates an asynchronous translation via a business rule. For more information, refer to **Business Rules for Asynchronous Translations**.
Value Generators	**Assign a Value**	Based on the configuration of the Value Generator, a value is assigned to the selected attribute. For more information, refer to any of the specific Value Generator topics included in the **Value Generators** topic.
	Release a Value	Based on the configuration of the Value Generator, a value is removed from the selected attribute. For more information, refer to any of the specific Value Generator topics included in the **Value Generators** topic.

Specifying a Business Action Operation

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

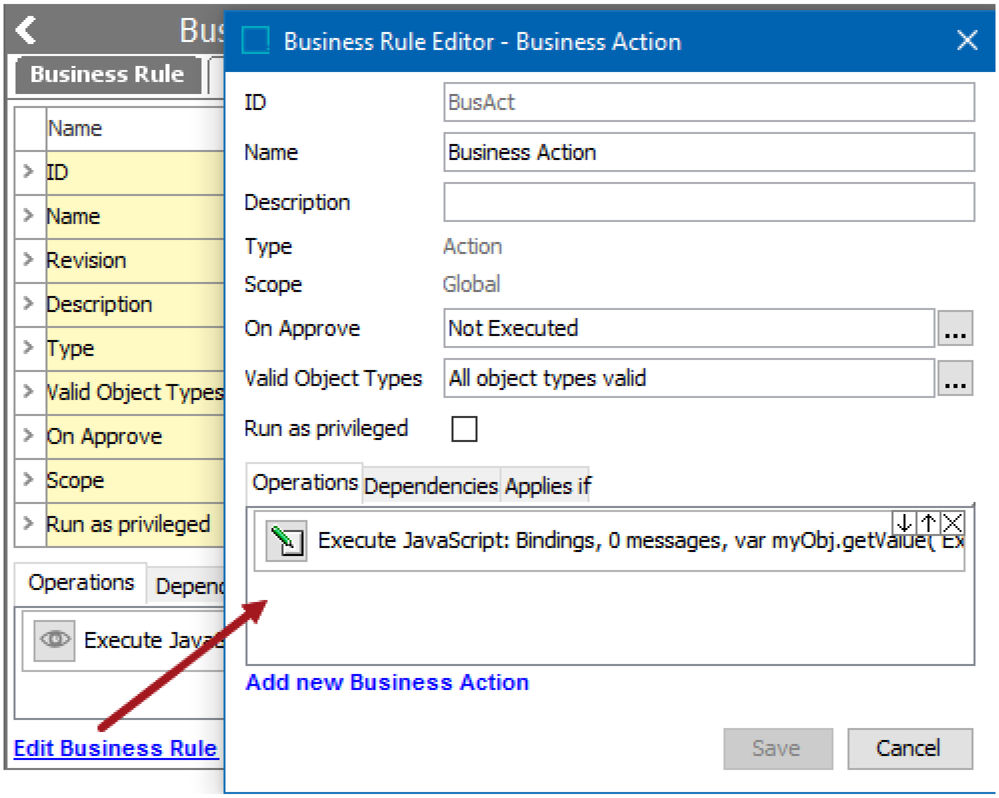
1. In System Setup, expand the **Business Rules** setup group and select an existing business action to display the business rule editor.



Name	Value
ID	BusAct
Name	Business Action
Revision	0.13 Last edited by USERJ on Fri Aug 18 14:48:36 EDT 2017
Description	
Type	Action
Valid Object Types	All object types valid
On Approve	Not Executed
Scope	Global
Run as privileged	□

Execute JavaScript: Bindings, 0 messages, var myVar = myObj.getValue("ExteriorColor");n...

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



ID: BusAct

Name: Business Action

Description:

Type: Action

Scope: Global

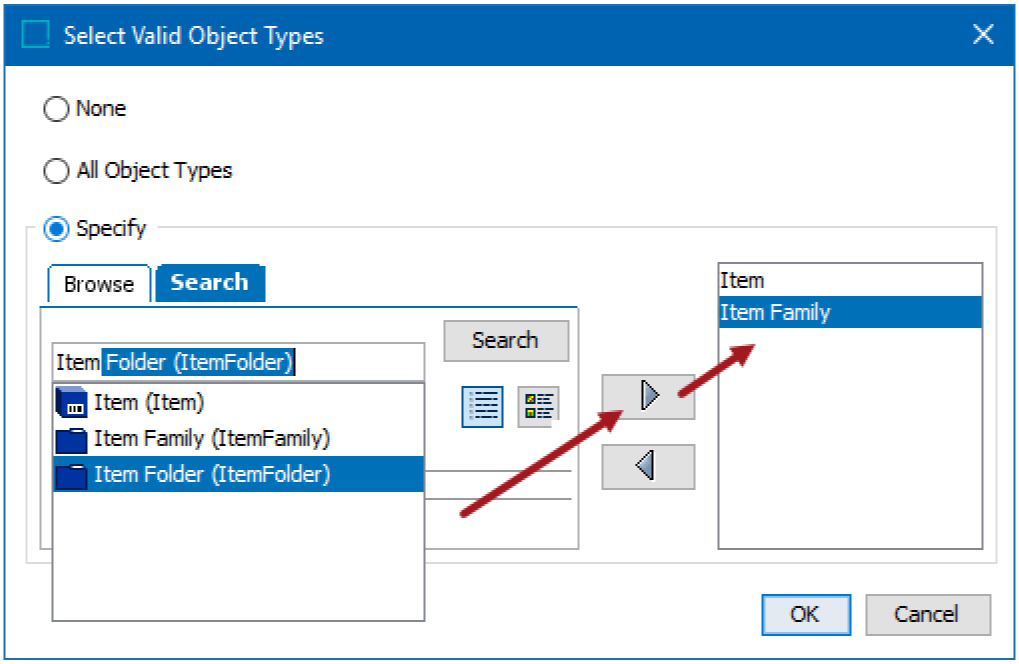
On Approve: Not Executed ...

Valid Object Types: All object types valid ...

Run as privileged: □

Execute JavaScript: Bindings, 0 messages, var myObj.getValue(Ex

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



○ None

○ All Object Types

● Specify

Item Folder (ItemFolder)

Item (Item)

Item Family (ItemFamily)

Item Folder (ItemFolder)

Item

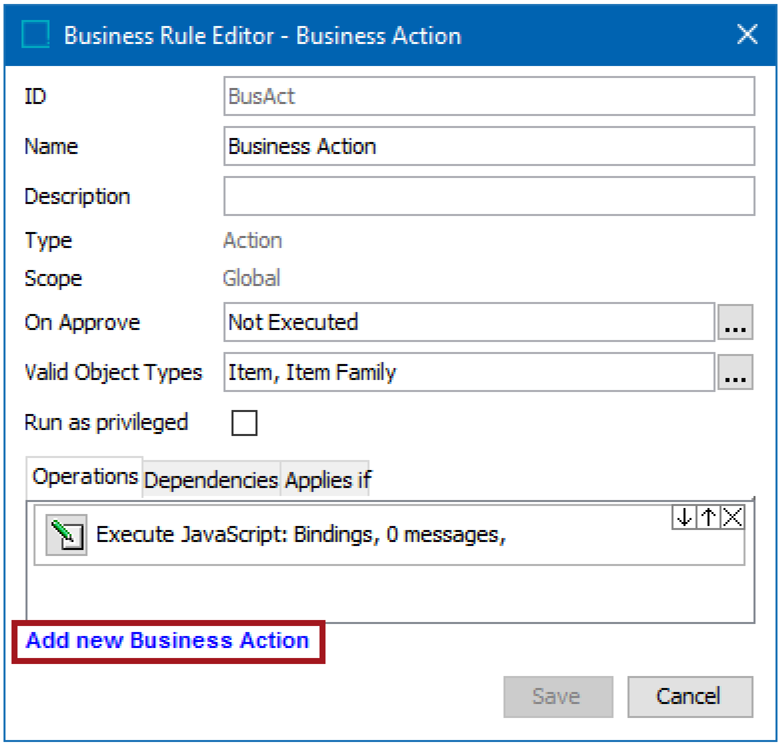
Item Family

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.



Business Rule Editor - Business Action

ID: BusAct

Name: Business Action

Description:

Type: Action

Scope: Global

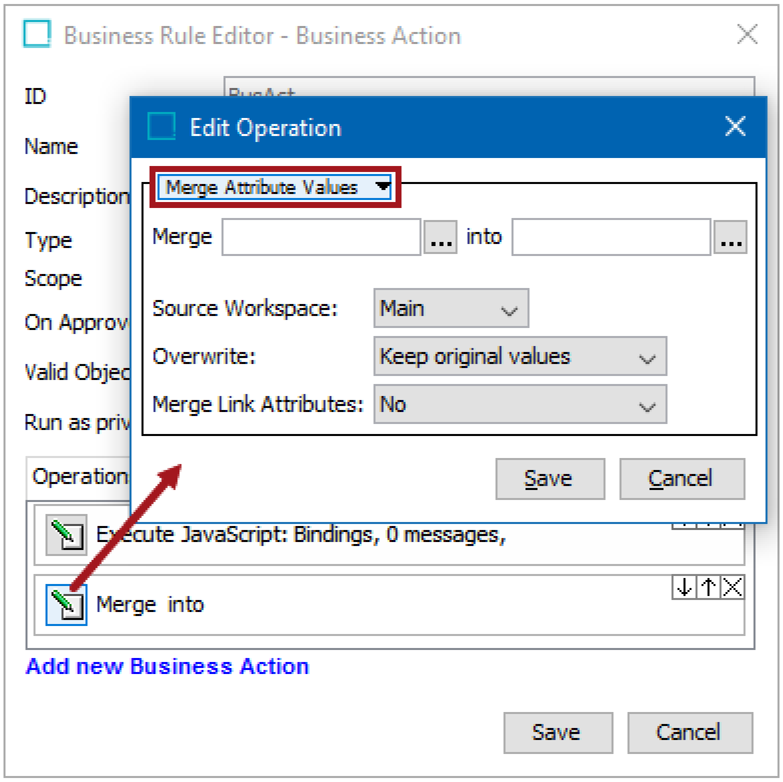
On Approve: Not Executed

Valid Object Types: Item, Item Family

Run as privileged: □

Execute JavaScript: Bindings, 0 messages,

6. Click the **Edit Operation** button (✎) for the newly added business action.



Business Rule Editor - Business Action

Edit Operation

Merge Attribute Values

Merge into

Source Workspace: Main

Overwrite: Keep original values

Merge Link Attributes: No

Execute JavaScript: Bindings, 0 messages,

Merge into

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

   Multiple business actions can also be configured using the Business Action bind in a JavaScript business rule, and the JavaScript controls the execution. For more information, refer to the **Business Action, Condition, and Function Binds** topic in the online help **Resource Materials** documentation.

   When actions are called through the 'Reference Other Business Action' plugin, all actions can be executed. For more information, refer to the Business Action: Reference Other Business Action topic.

Business Action: Add Attribute Link

This operation 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.

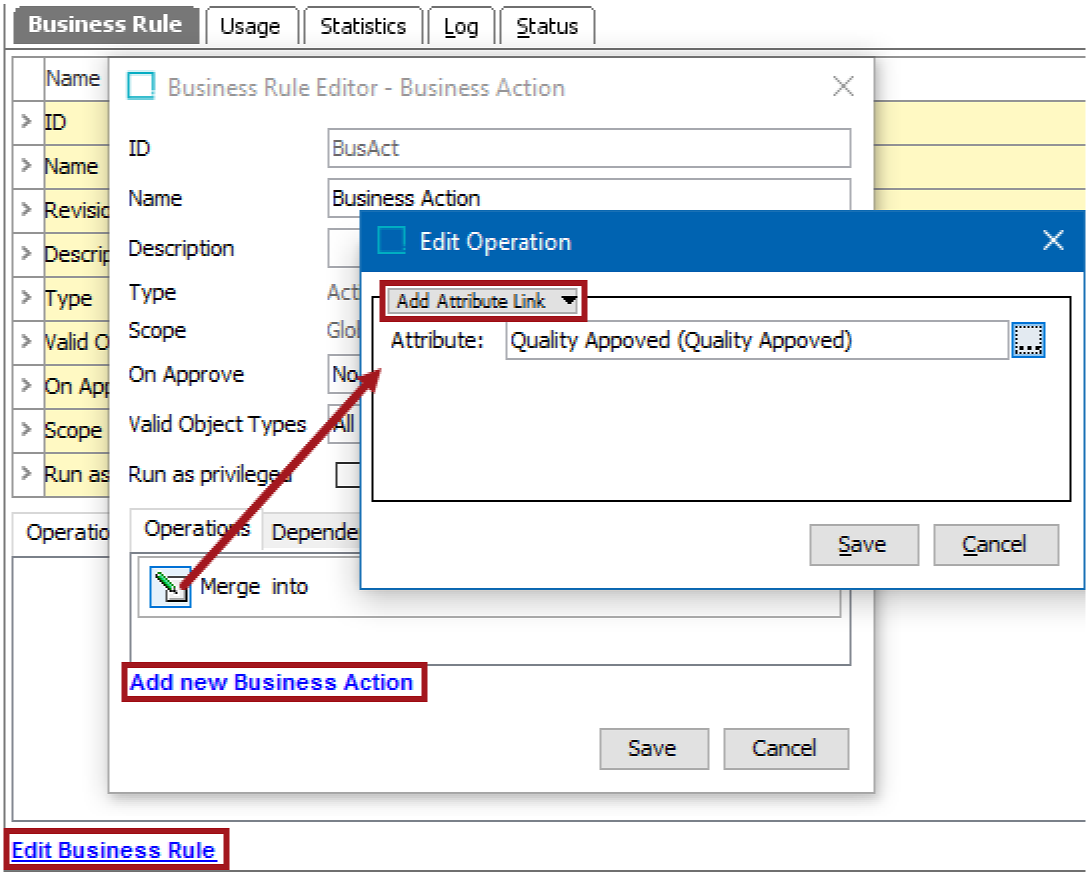
This operation can also be run via a Bulk Update and sample data is provided in the **References and Links: Add Attribute Link Operation** topic in the **Bulk Updates** documentation.

Prerequisites

Before using this operation:

1. Ensure the product attribute link type is set up. For more information, refer to the **Product Attribute Link Type** topic in the **System Setup** 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, Function, or Library** topic.

Configuration



ID: BusAct

Name: Business Action

Description:

Type: Act

Scope: Glo

On Approve: No

Valid Object Types: All

Run as privileged: □

Attribute: Quality Appoved (Quality Appoved)

1. On the Edit Operation dialog, from the dropdown select the References and Links group and then select the **Add Attribute Link** option.
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 operation 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.

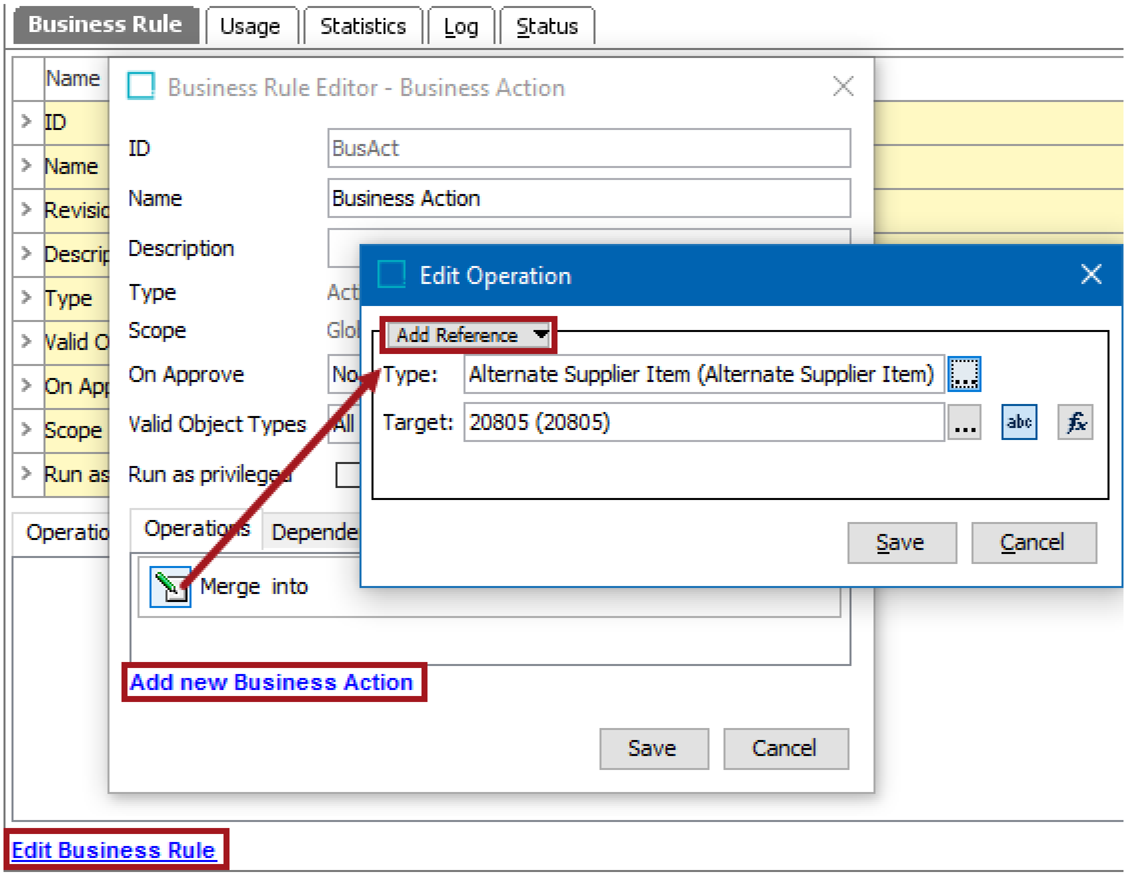
This operation can also be run via a Bulk Update and sample data is provided in the **References and Links: Add Reference or Add Referenced By Operations** topic in the **Bulk Updates** documentation.

Prerequisites

Before using this operation:

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, refer to the **Reference and Link Types** topic in the **System Setup** documentation.
2. Note that attempting to add a target that already exists on the object causes the action to fail with the message 'Reference already exists :< reference type>'
3. Create a business rule as defined in the **Creating a Business Rule, Function, or Library** topic.

Configuration



ID: BusAct

Name: Business Action

Description:

Type: Act

Scope: Glol

On Approve: No

Valid Object Types: All

Run as privileged: □

Add Reference

Type: Alternate Supplier Item (Alternate Supplier Item) ...

Target: 20805 (20805) ... abc fx

Merge into

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

> **Note:** The selected object dictates the label on the manual button. If the selected object requires a text value, the label is 'abc.' If the selected object is numeric, '123' is displayed, and so on for each validation base type.

- For manual selection, click the 'abc' button (abc), click the ellipsis button (...) to display the Select Reference Target dialog, select a valid target, and click the **Select** button.
- For function selection, click the 'fx' button (fx), click the ellipsis button (...) to display the Function Editor dialog. Add a function and evaluate, then click **OK**. For more information, refer to the **Using Function Editor** topic in the **Resource Materials** online help.

> **Note:** A function that cannot return the specific Target Type ID returns the 'Could not get the reference target.' error.

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

Business Action: Add Referenced By

This operation 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.

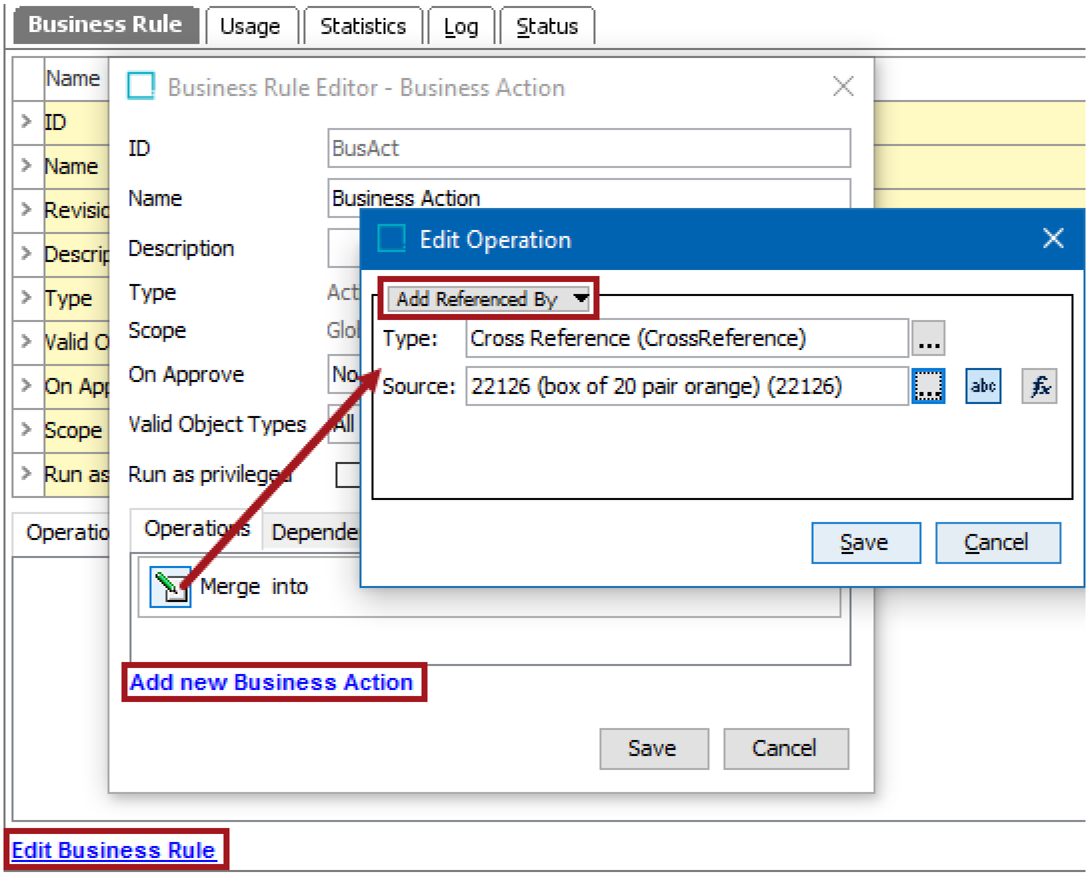
This operation can also be run via a Bulk Update and sample data is provided in the **References and Links: Add Reference or Add Referenced By Operations** topic in the **Bulk Updates** documentation.

Prerequisites

Before using this operation:

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, refer to the **Reference and Link Types** topic in the **System Setup** documentation.
2. Note that attempting to add a target that already exists on the object causes the action to fail with the message 'Reference already exists :< reference type>'
3. Create a business rule as defined in the **Creating a Business Rule, Function, or Library** topic.

Configuration



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

> **Note:** The selected object dictates the label on the manual button. If the selected object requires a text value, the label is 'abc.' If the selected object is numeric, '123' is displayed, and so on for each validation base type.

- For manual selection, click the 'abc' button (abc), click the ellipsis button (...) to display the Select Reference Source dialog, select a valid source, and click the **Select** button.
- For function selection, click the 'fx' button (fx), click the ellipsis button (...) to display the Function Editor dialog. Add a function and evaluate, then click **OK**. For more information, refer to the **Using Function Editor** topic in the **Resource Materials** online help.

> **Note:** A function that cannot return the specific Target Type ID returns the 'Could not get the reference target.' error.

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

Business Action: Automatic Classification

This operation 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 are automatically classified 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, refer to the **Using Automatic Classification with Business Actions** topic in the **Automatic Classification** documentation.

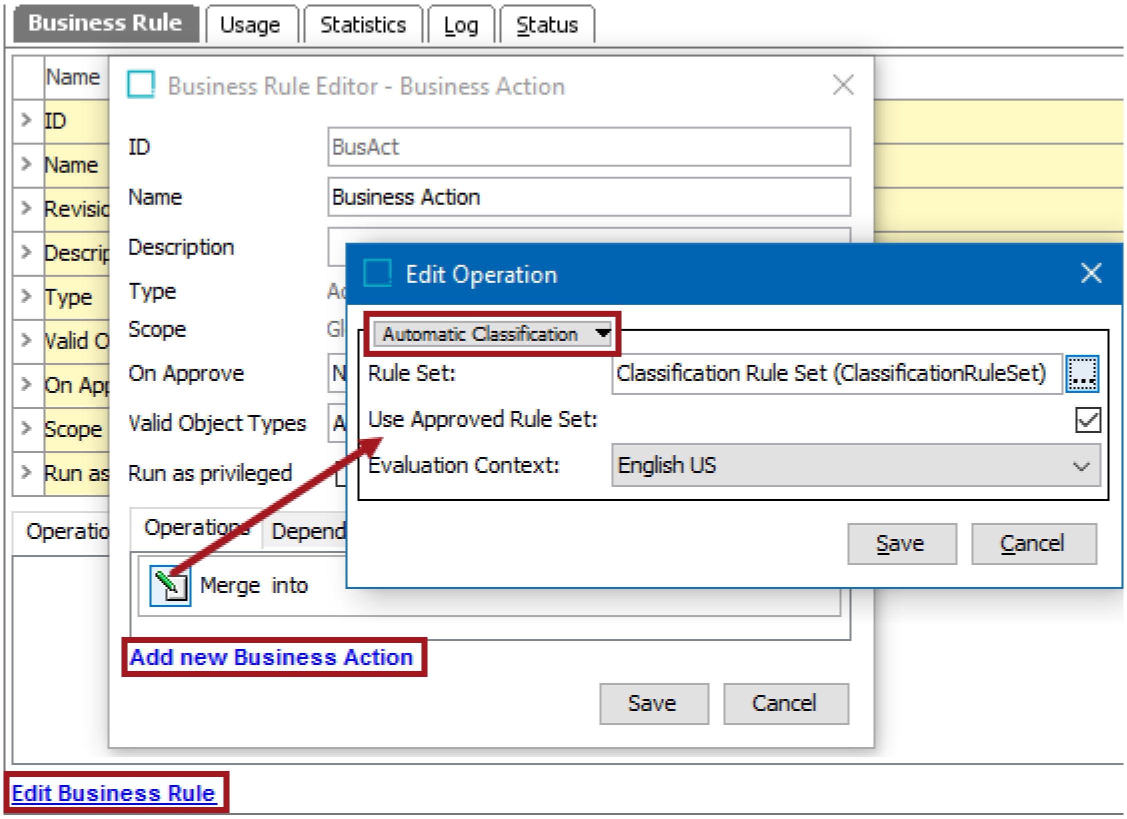
This operation can also be run via a Bulk Update and sample data is provided in the **References and Links: Automatic Classification Operation** topic in the **Bulk Updates** documentation.

Prerequisites

Before using this operation:

1. Ensure the Automatic Classification component is activated.
2. Ensure a rule set configuration for automatic classification exists, as defined in the **Initial Setup for Automatic Classification** topic.
3. Ensure the Rule Set object instance in Tree is configured per the component model definition.
4. Create a business rule as defined in the **Creating a Business Rule, Function, or Library** topic.

Configuration



Business Action Editor - Business Action

ID: BusAct

Name: Business Action

Edit Operation

Automatic Classification

Rule Set: Classification Rule Set (ClassificationRuleSet)

Use Approved Rule Set: ☑

Evaluation Context: English US

1. On the Edit Operation dialog, from the dropdown select the References and Link group and then select the **Automatic Classification** option.

2. For the **Rule Set** parameter, click the ellipsis button (**...**), and select the rule set to be executed.

3. For the **Use Approved Rule Set** parameter, check the checkbox to use the approved version of the rule set.

4. For the **Evaluation Context** parameter, 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 operation 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, refer to the **Business Rules and Workflows** topic in the **Workflows** documentation.

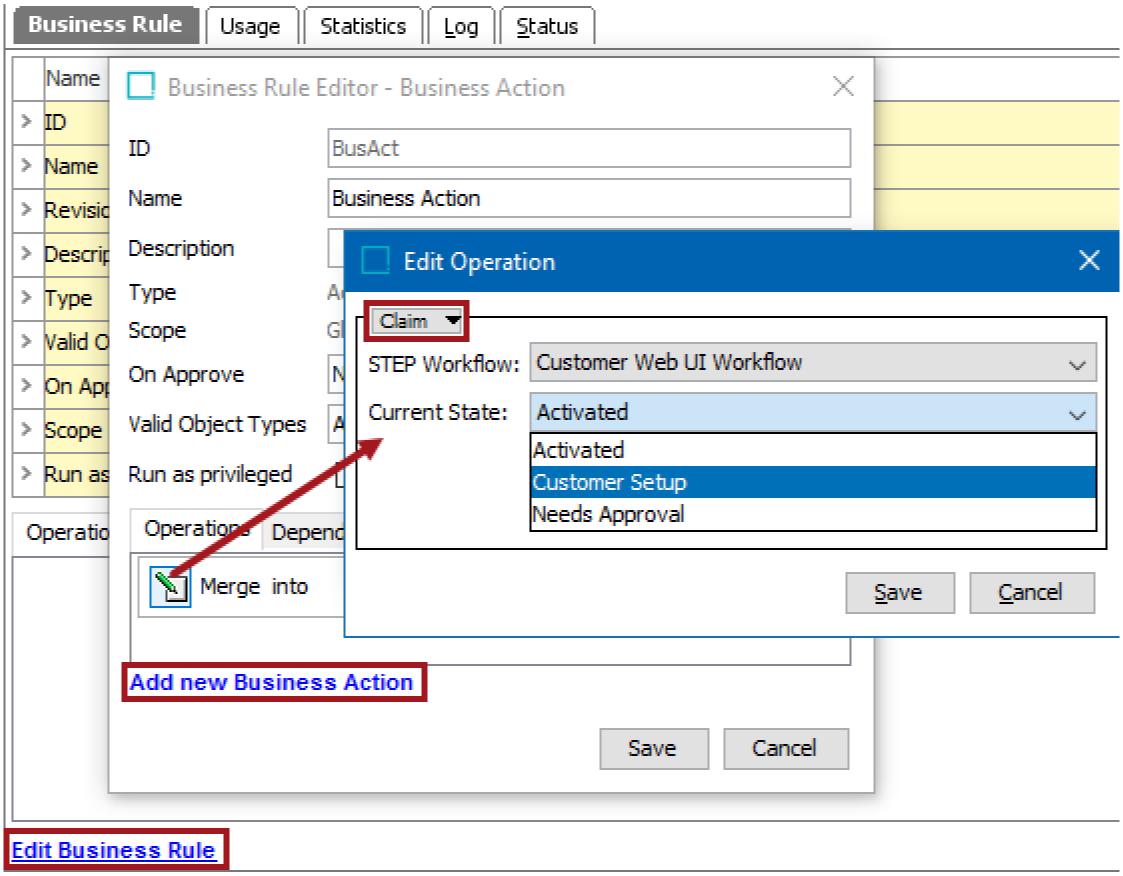
This operation can also be run via a Bulk Update and sample data is provided in the **Workflow: Claim Operation** topic in the **Bulk Updates** documentation.

Prerequisites

Before using this operation:

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

Configuration



Business Rule Editor - Business Action

ID: BusAct

Name: Business Action

Edit Operation

Claim

STEP Workflow: Customer Web UI Workflow

Current State: Activated

1. On the Edit Operation dialog, from the dropdown select the Workflow group and then select the **Claim** option.
2. For the **STEP Workflow** parameter, select the relevant workflow from the dropdown.
3. For the **Current State** parameter, 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, refer to **Business Actions**.
- **JavaScript Binds** for action and condition operations. Binds provide access to the data being worked on. For more information, refer to **JavaScript Binds** in the **Resource Materials** online help.

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

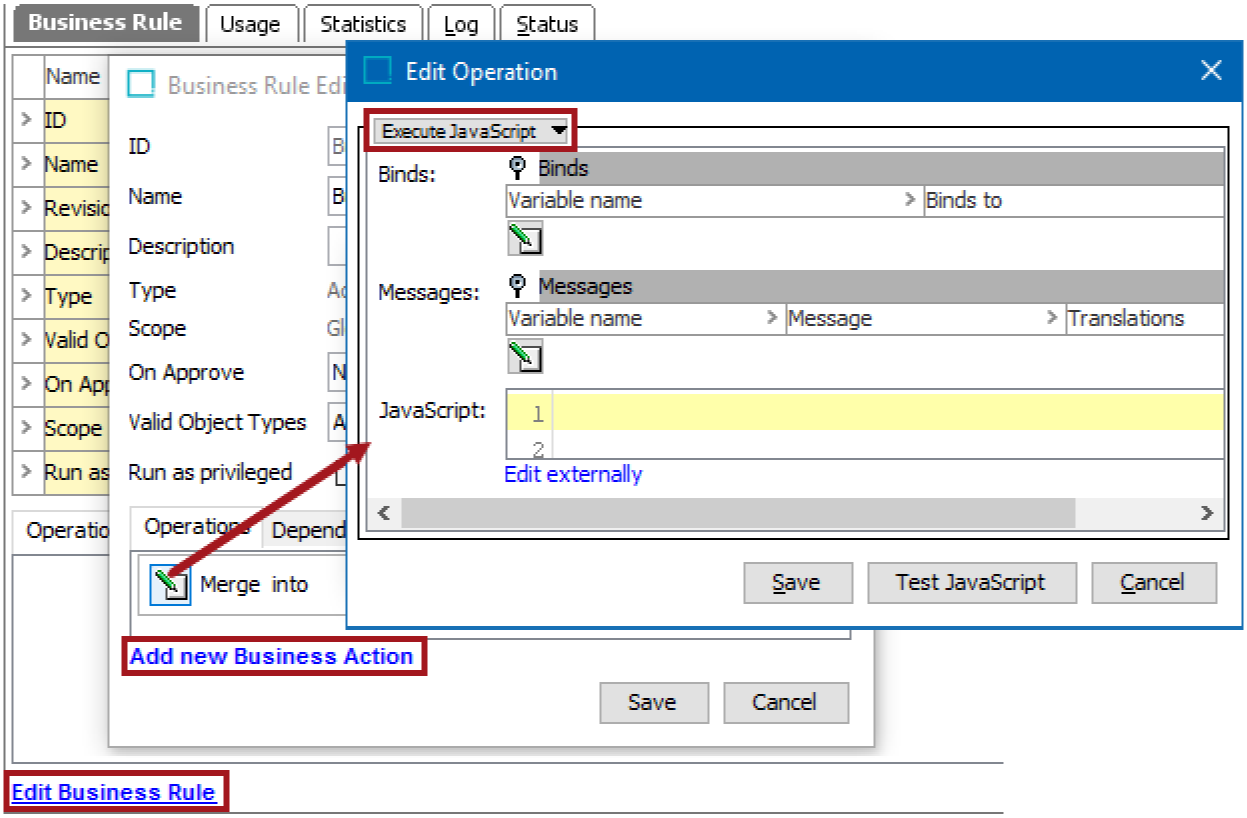
- **Business Libraries** is a set of JavaScript functions that can be reused in multiple business rules. For more information, refer to **Business Libraries**.
- **Scripting API** documentation is accessed via the **STEP API Documentation**, at [system]/sdk or accessible from the Start Page.

Prerequisites

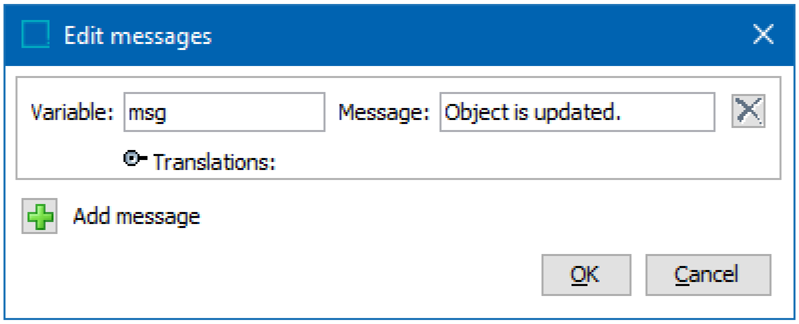
Before using this action:

- Understand the implications of using JavaScript as defined in the **Java vs. JavaScript** topic in the **Resource Materials** online help.
- Understand the considerations that should be taken as defined in the **JavaScript Considerations** topic in the **Resource Materials** online help.
- Create a business rule as defined in the **Creating a Business Rule, Function, 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 in the **Resource Materials** online help.
3. For the **Messages** parameter, create messages needed for the JavaScript code as follows:
   - Click the **Edit** button (✎) to display the Edit Messages dialog.



Variable: msg

Message: Object is updated.

   - 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, refer to 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 using 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).

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, refer to **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, refer to **Adding a Localized Business Rule Message** in the **Business Rules** documentation.

Adding a Localized Business Rule Message

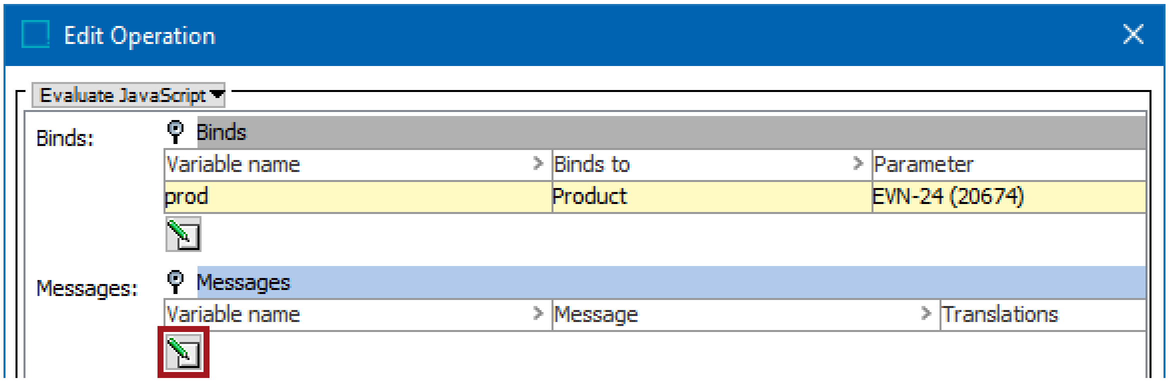
A message can display static text or dynamic text, based on the value of a variable.

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, refer to the **Editing a Business Rule or Function** topic.

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

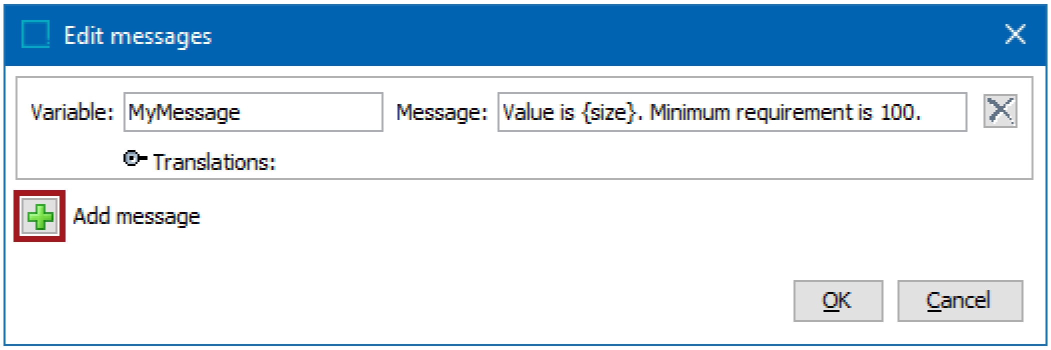
2. On the Edit Operation dialog, for the Messages parameter, click the **Edit** button (✎) to display the 'Edit messages' dialog.



Variable name	Binds to	Parameter
prod	Product	EVN-24 (20674)

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

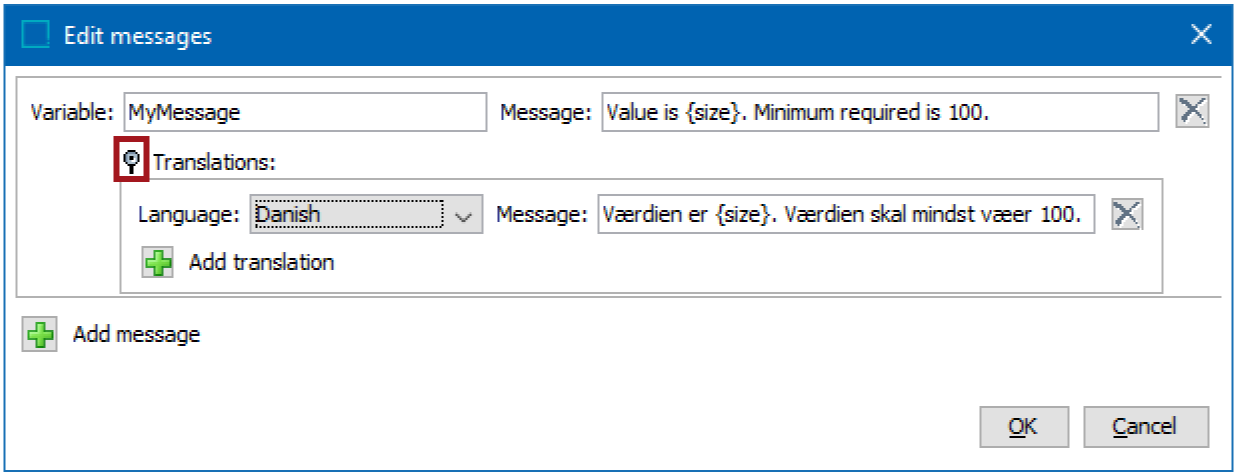
   A variable message can include the name of the business rule or any other data that would be helpful in resolving the problem reported by the message.



Variable: MyMessage
Message: Value is {size}. Minimum requirement is 100.

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.



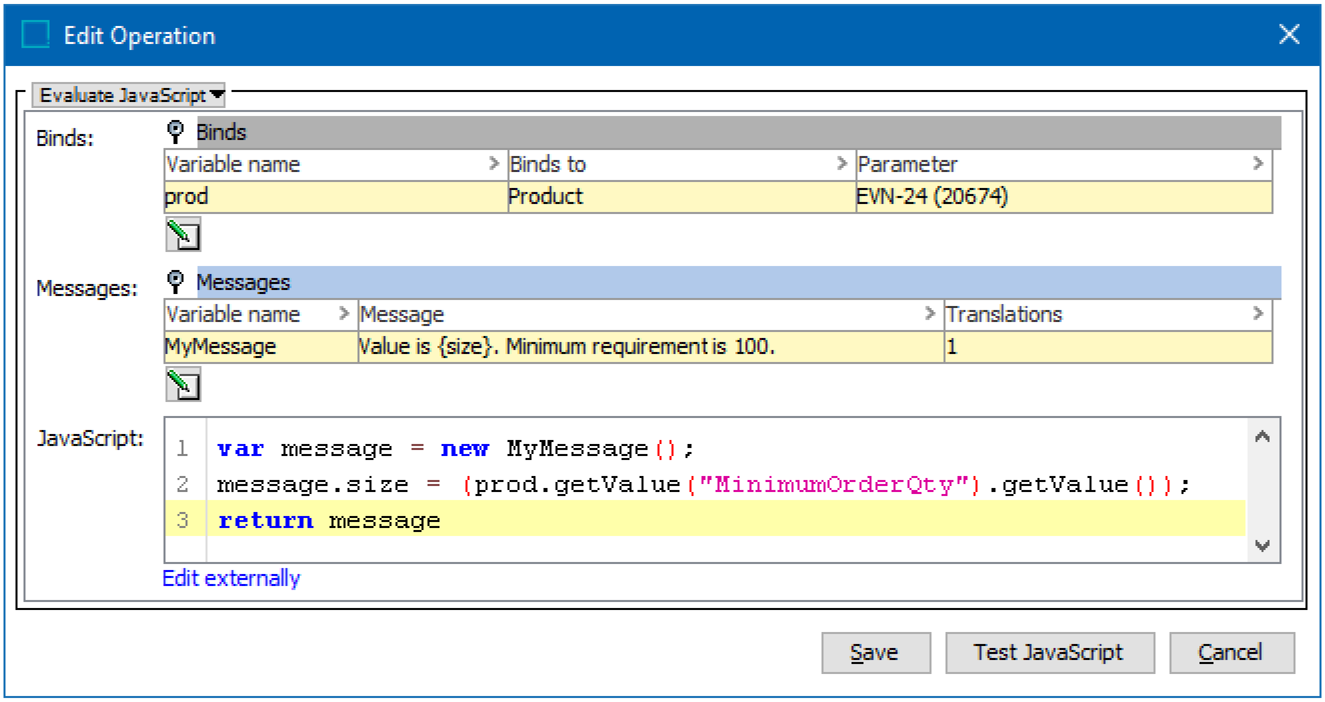
Edit messages

Variable: MyMessage Message: Value is {size}. Minimum required is 100.

Translations:

Language: Danish Message: Værdien er {size}. Værdien skal mindst væer 100.

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.



Edit Operation

Evaluate JavaScript

Binds:

Variable name	Binds to	Parameter
prod	Product	EVN-24 (20674)

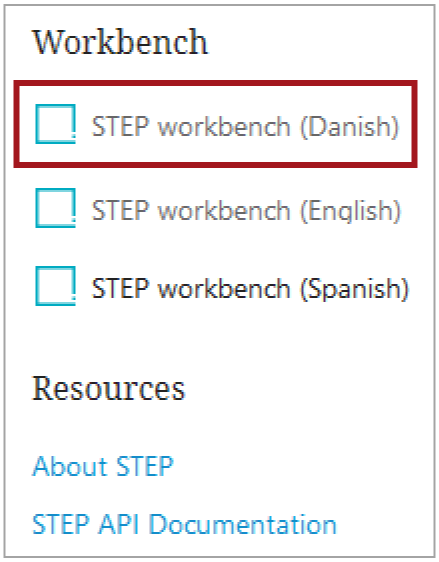
Messages:

Variable name	Message	Translations
MyMessage	Value is {size}. Minimum requirement is 100.	1

JavaScript:

```
var message = new MyMessage();
message.size = (prod.getValue("MinimumOrderQty").getValue());
return message
```

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



Workbench

STEP workbench (Danish)

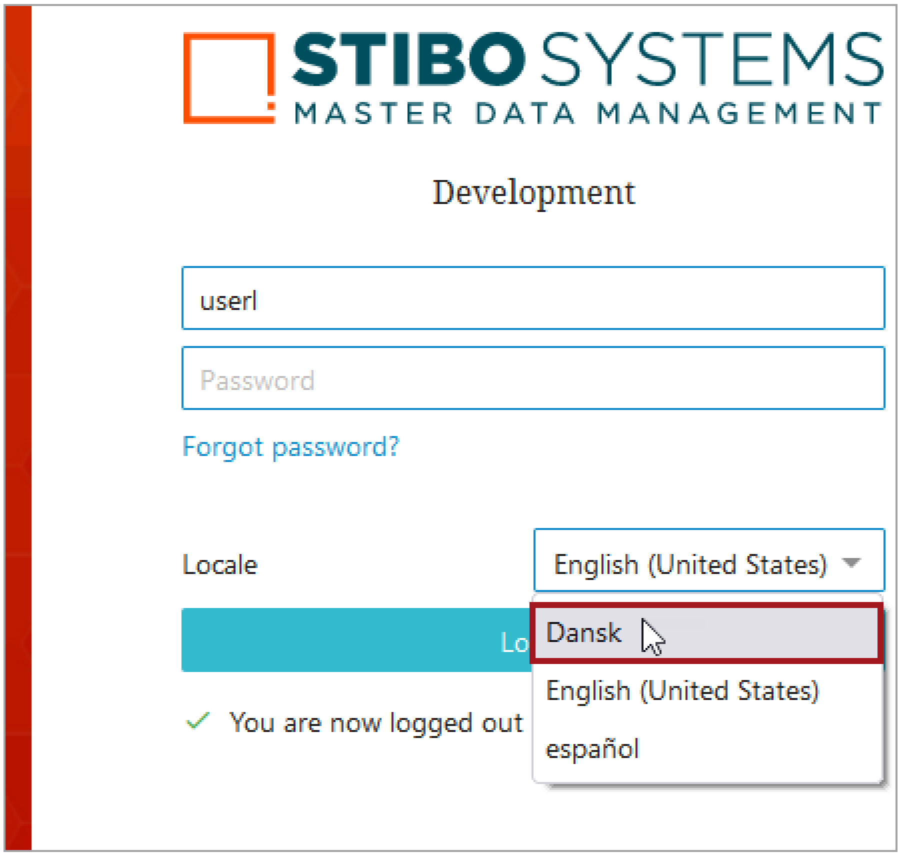
STEP workbench (English)

STEP workbench (Spanish)

Resources

About STEP

STEP API Documentation



STIBO SYSTEMS MASTER DATA MANAGEMENT

Development

userl

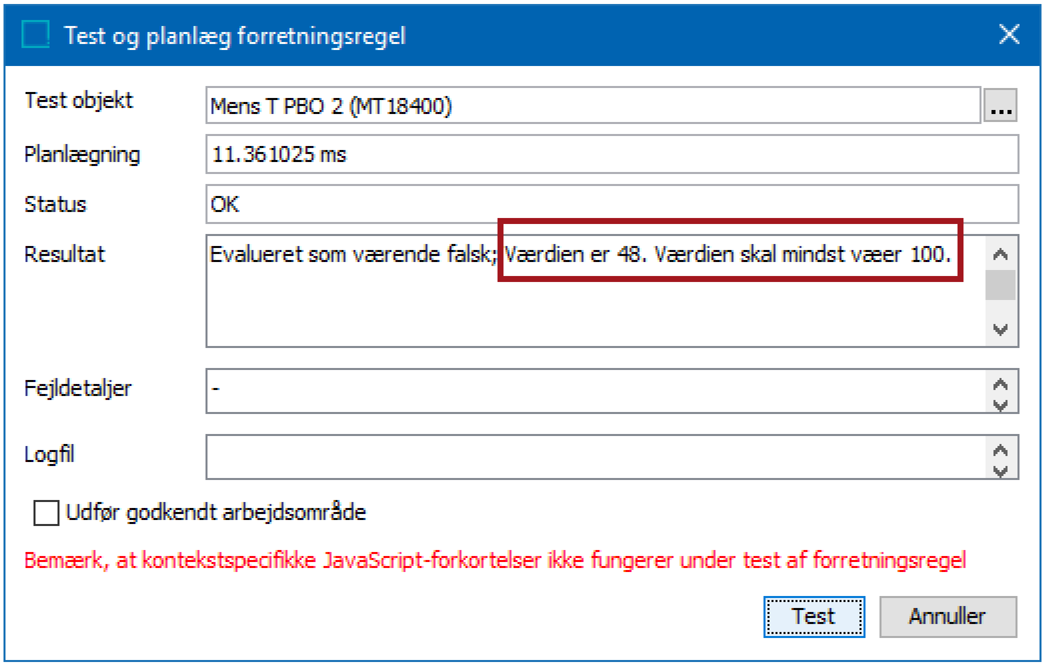
Password

Forgot password?

Locale: English (United States)

✓ You are now logged out

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



Test og planlæg forretningsregel

Test objekt	Mens T PBO 2 (MT18400)
Planlægning	11.361025 ms
Status	OK
Resultat	Evalueret som værende falsk; Værdien er 48. Værdien skal mindst væer 100.
Fejldetaljer	-
Logfil	

☐ Udfør godkendt arbejdsområde

Bemærk, at kontekstspecifikke JavaScript-forkortelser ikke fungerer under test af forretningsregel

Business Action: Generate Match Codes

This operation 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.

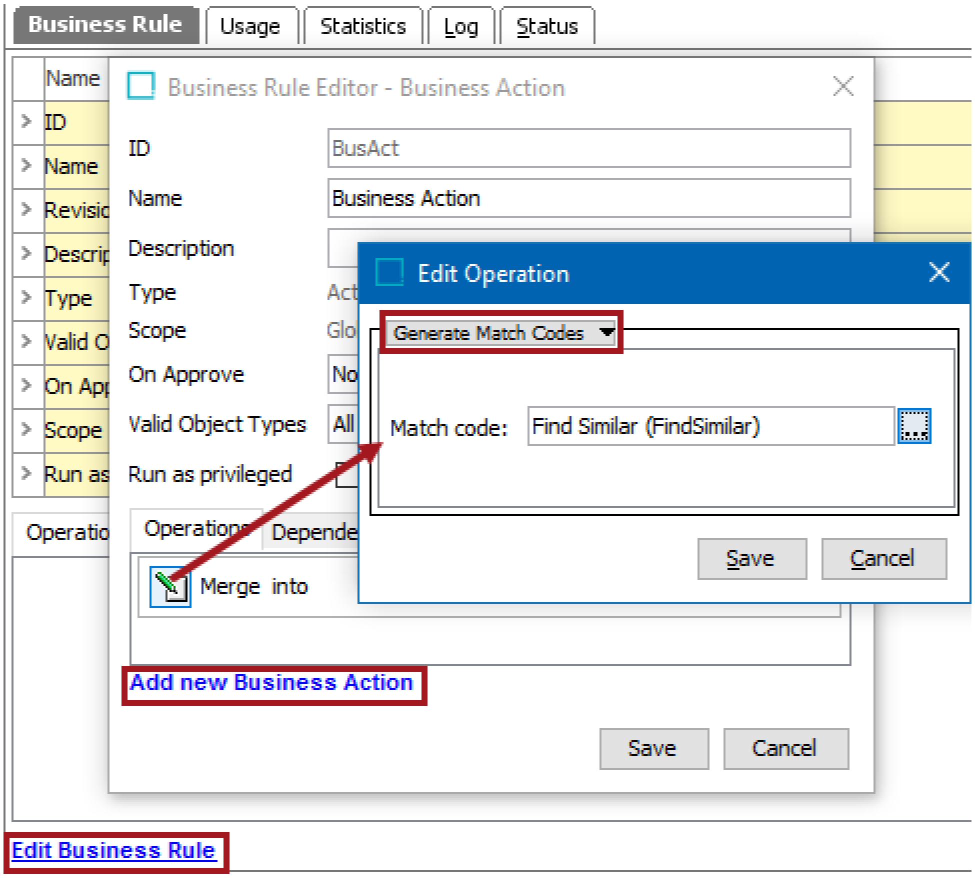
This operation can also be run via a Bulk Update and sample data is provided in the **Data Quality: Generate Match Codes Operation** topic in the **Bulk Updates** documentation.

Prerequisites

Before using this operation:

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

Configuration



Business Rule Editor - Business Action

ID: BusAct

Name: Business Action

Edit Operation

Generate Match Codes

Match code: Find Similar (FindSimilar)

Merge into

Add new Business Action

Edit Business Rule

1. On the Edit Operation dialog, from the dropdown select the Data Quality group and then select the **Generate Match Codes** option.
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 operation 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, refer to the **Business Rules and Workflows** topic in the **Workflows** documentation.

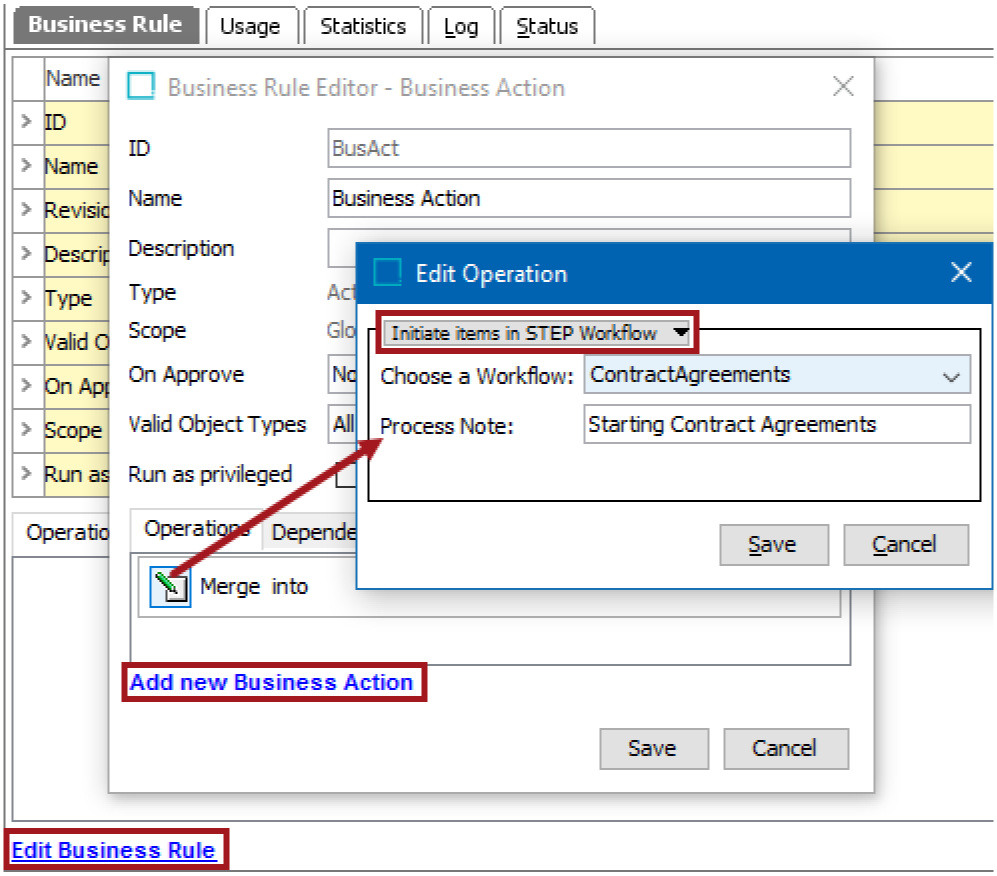
This operation can also be run via a Bulk Update, as described in the **Workflow: Initiate Items In STEP Workflow Operation** topic in the **Bulk Updates** documentation.

Prerequisites

Before using this operation:

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, Function, or Library** topic.

Configuration



Business Rule Editor - Business Action

ID: BusAct

Name: Business Action

Edit Operation

Initiate items in STEP Workflow

Choose a Workflow: ContractAgreements

Process Note: Starting Contract Agreements

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

2. For the **Choose a Workflow** parameter, select a workflow from the dropdown.

3. For the **Process Note** parameter, type a note for the background process.

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

Business Action: Merge Attribute Values

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

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

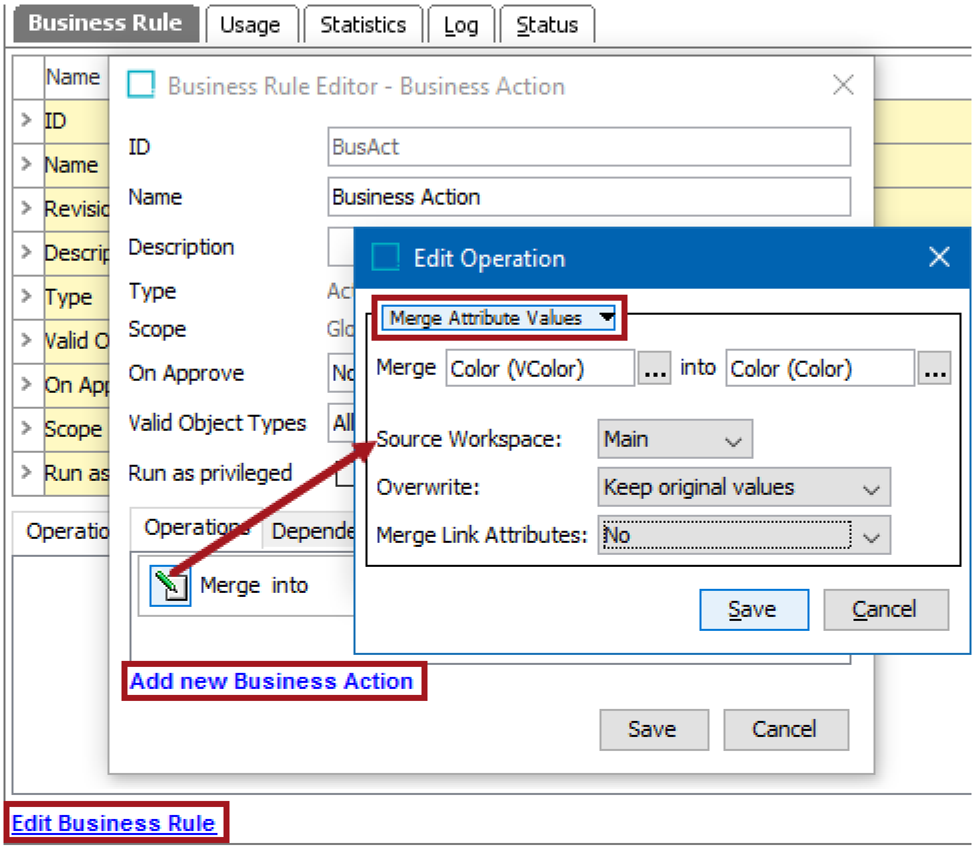
This operation can also be run via a Bulk Update and sample data is provided in the **Attribute Values: Merge Attribute Values Operation** topic in the **Bulk Updates** documentation.

Prerequisites

Before using this operation:

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, Function, or Library** topic.

Configuration



ID: BusAct

Name: Business Action

Merge Attribute Values

Merge Color (VColor) into Color (Color)

Source Workspace: Main

Overwrite: Keep original values

Merge Link Attributes: No

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, from the dropdown select the Workflow group, then select the **Merge Attribute Values** option.
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. Refer to 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.

Business Action: Overlap Analysis

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

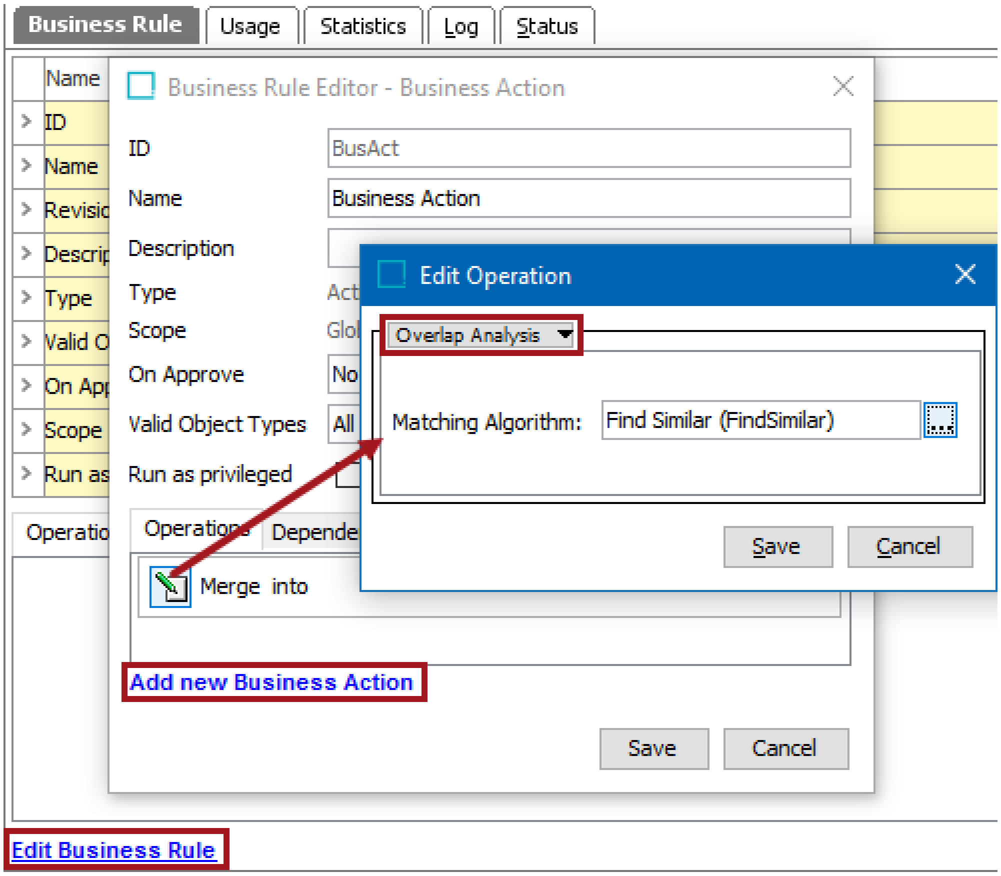
Important: This operation is only intended to be used together with an import running in Test mode. In this mode, statistics are reported on how many objects in import file will match existing objects in STEP, without actually importing the data.

Prerequisites

Before using this operation:

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

Configuration



ID: BusAct

Name: Business Action

Description:

Type: Act

Scope: Glo

On Approve: No

Valid Object Types: All

Run as privileged: □

Overlap Analysis

Matching Algorithm: Find Similar (FindSimilar)

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

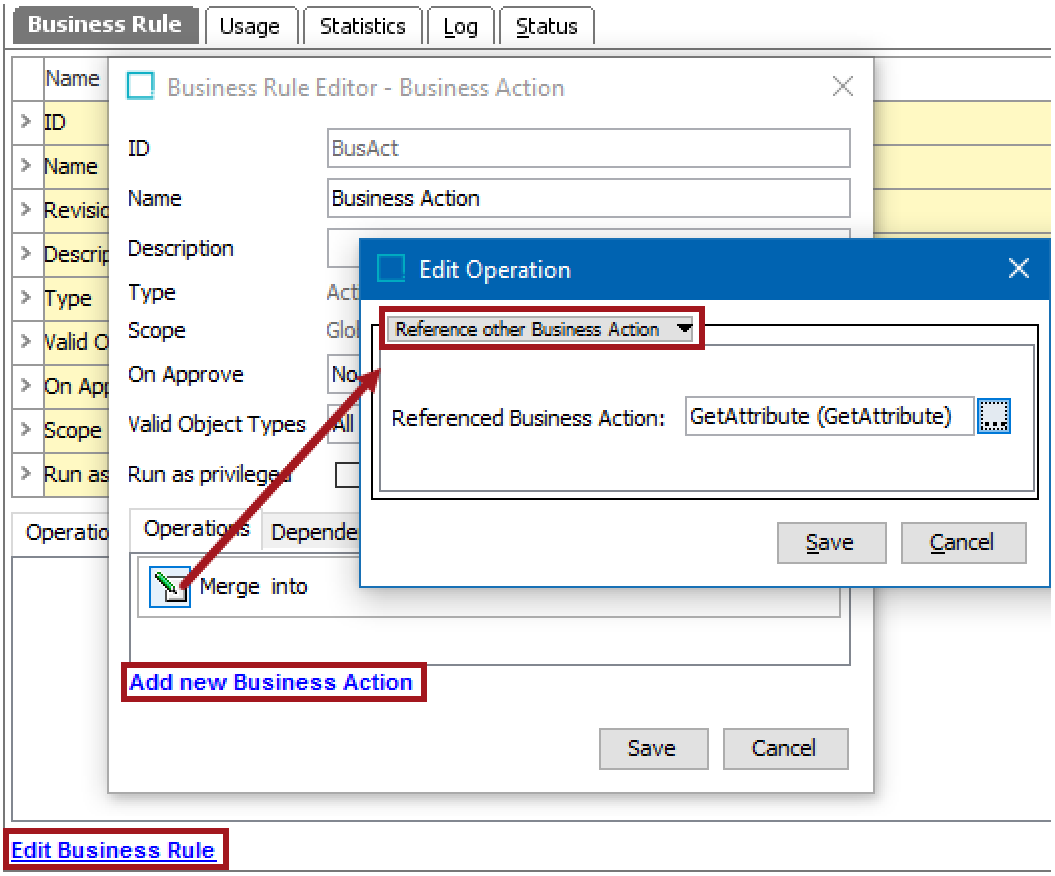
For example, a business action that updates an object name needs to be executed within other business rules run by the Import Manager or in a Web UI.

Prerequisites

Before using this operation:

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, Function, or Library** topic.

Configuration



ID: BusAct

Name: Business Action

Reference other Business Action

Referenced Business Action: GetAttribute (GetAttribute)

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 operation 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.

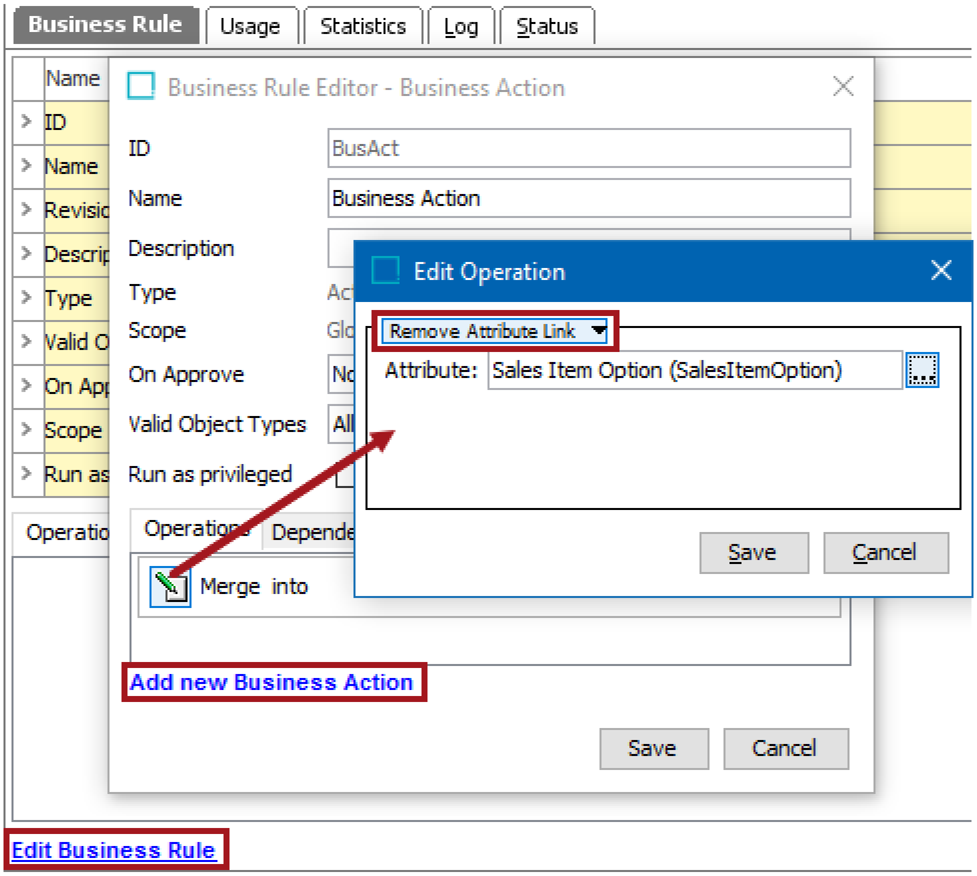
This operation can also be run via a Bulk Update and sample data is provided in the **References and Links: Remove Attribute Link Operation** topic in the **Bulk Updates** documentation.

Prerequisites

Before using this operation:

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

Configuration



Business Rule Editor - Business Action

ID: BusAct

Name: Business Action

Edit Operation

Remove Attribute Link

Attribute: Sales Item Option (SalesItemOption)

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

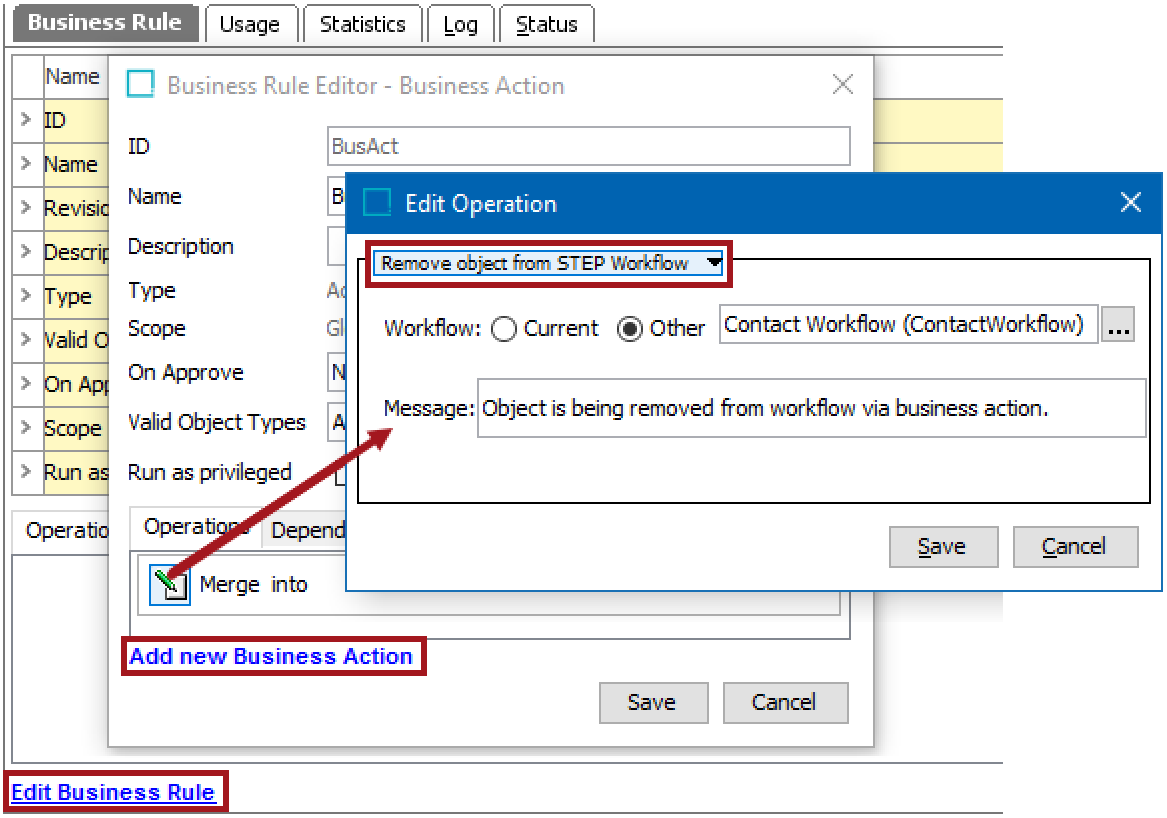
This operation is similar to one in Bulk Update, as described in the **Workflow: Remove Items from STEP Workflow Operation** topic in the **Bulk Updates** documentation.

Prerequisites

Before using this operation:

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, Function, or Library** topic.

Configuration



Business Rule Editor - Business Action

ID: BusAct

Edit Operation

Remove object from STEP Workflow

Workflow: ○ Current ● Other Contact Workflow (ContactWorkflow)

Message: Object is being removed from workflow via business action.

1. On the Edit Operation dialog, from the dropdown select the Workflow group and then select the **Remove object from STEP Workflow** option.
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 operation removes a reference to a specific target or all references of a specific type.

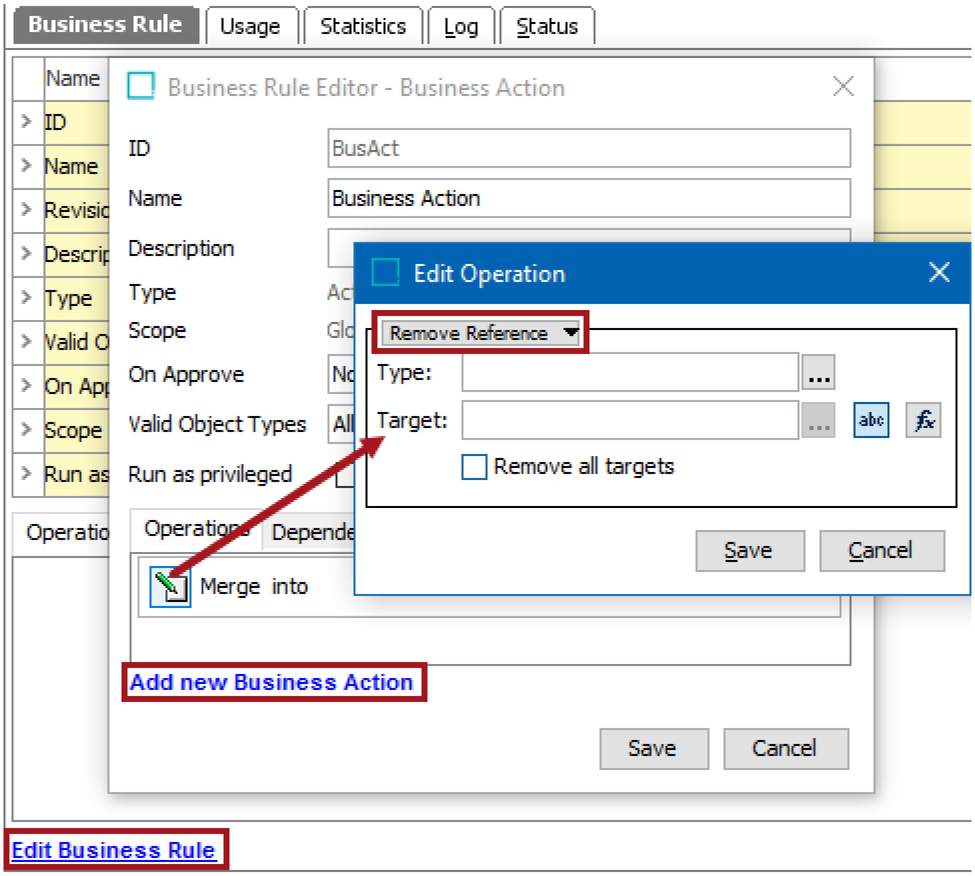
For example, the 'free_objects' reference type is used to indicate a seasonal sales offer items. When the sale period has ended, removal of all 'free_objects' references must be removed. This can be achieved by creating a collection of all nodes that include the reference, and then running a bulk update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

This operation can also be run via a Bulk Update and sample data is provided in the **References and Links: Remove Reference Operation** topic in the **Bulk Updates** documentation.

Prerequisite

Before using this operation, create a business rule as defined in the **Creating a Business Rule, Function, or Library** topic.

Configuration



1. On the Edit Operation dialog, from the dropdown select the References and Links group and then select the **Remove Reference** option.
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 (abc), click the ellipsis button (...) to display the Select Reference Target picker, select a valid target, and click the **Select** button.
    - Click the 'fx' button (fx), click the ellipsis button (...) to display the Function Editor dialog, write a function to select a valid target, and click the **Select** button. For more information, refer to the **Using Function Editor** topic in the **Resource Materials** online help.
4. For the **Remove all targets** parameter, set the checkbox as follows:
    - Unchecked means references of the specified type are removed from the modified objects.
    - Checked with a blank Target parameter means all references of the specified type are removed from the modified objects.
5. Click the **Save** button to add the operation to the business rule editor.

Business Action: Send Email

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

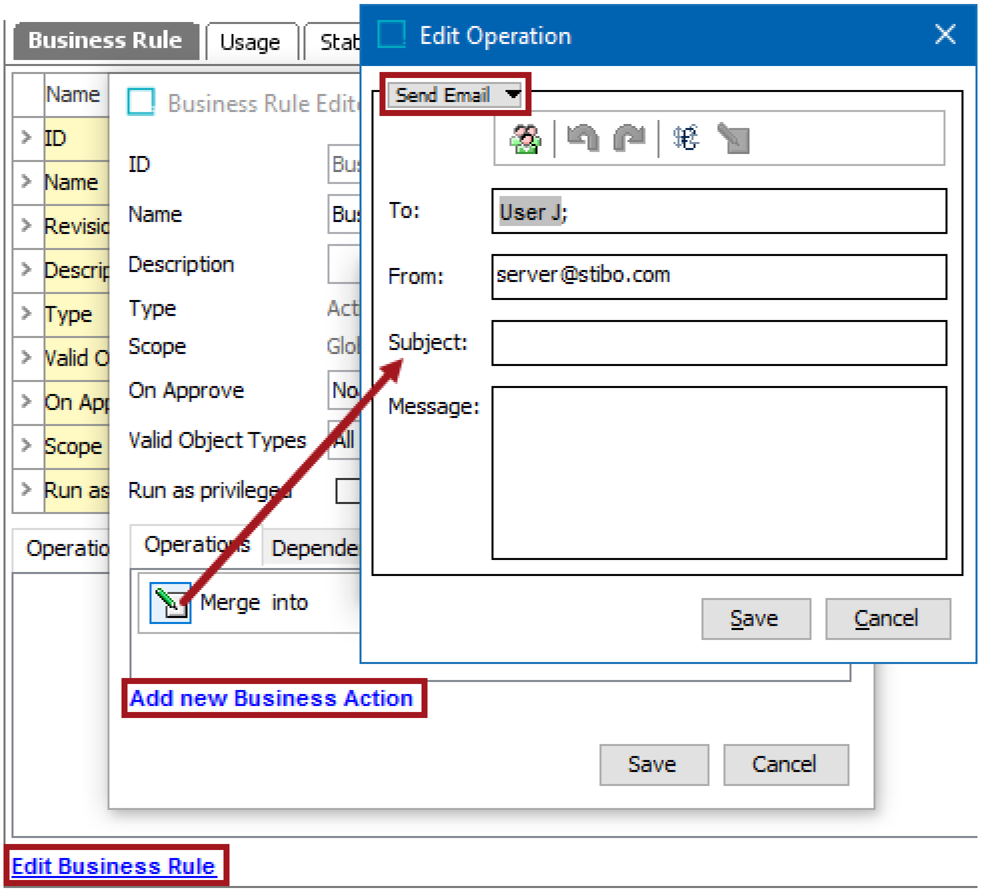
For example, an email can be sent when an object moves from one state of a workflow to another, or upon successful approval of an object.

Prerequisites

Before using this operation:

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 under the User & Groups node. Open the group and 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, Function, or Library** topic.

Configuration



To: User J;

From: server@stibo.com

Subject:

Message:

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

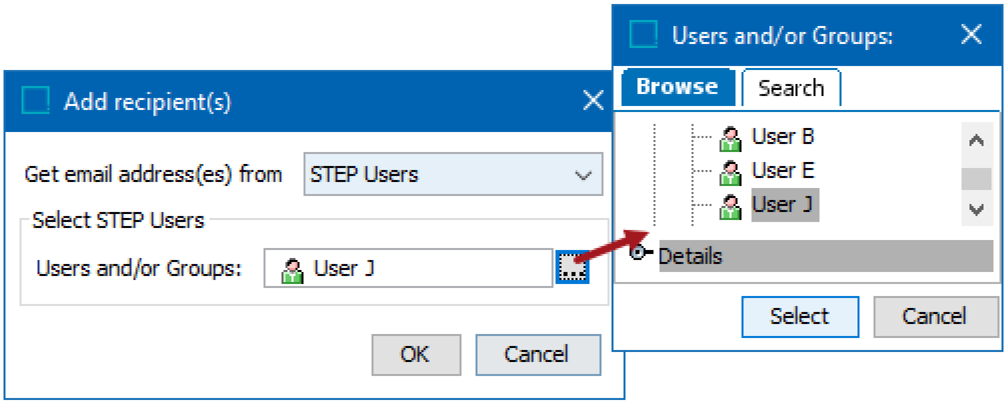
   Action buttons are enabled as the appropriate field is selected or action is performed.

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.

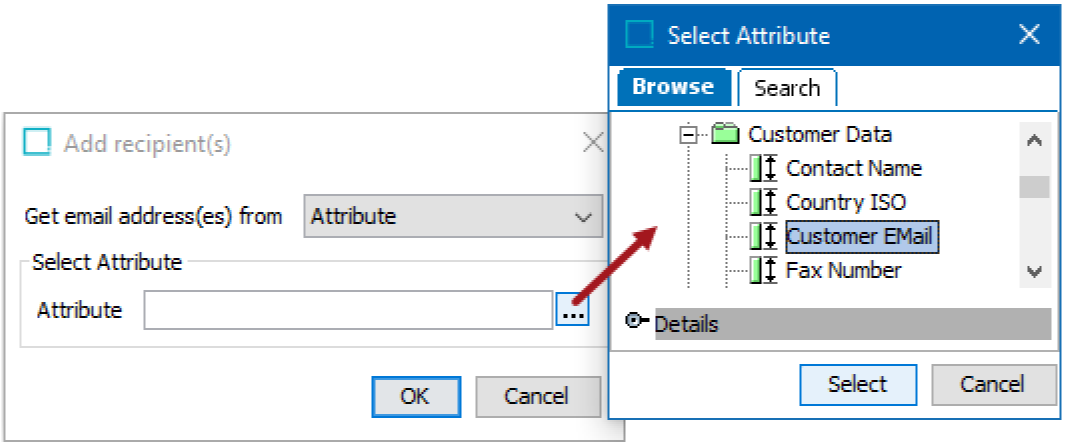


Get email address(es) from: STEP Users

Select STEP Users

Users and/or Groups: User J

- 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.



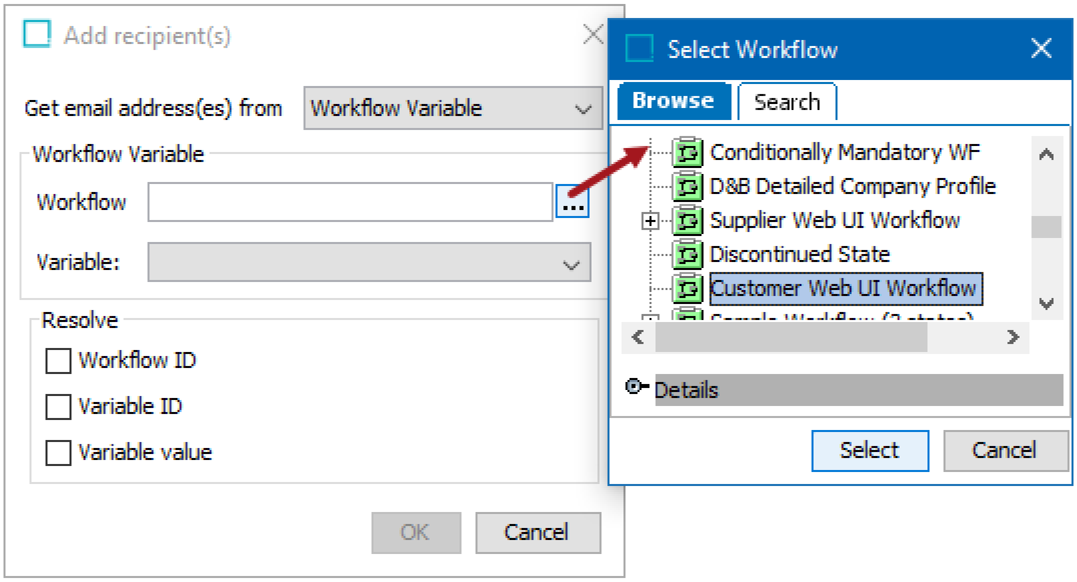
Get email address(es) from: Attribute

Select Attribute

Attribute:

- 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, refer to 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.



Get email address(es) from: Workflow Variable

□ Workflow ID
□ Variable ID
□ Variable value

Conditionally Mandatory WF
D&B Detailed Company Profile
Supplier Web UI Workflow
Discontinued State
Customer Web UI Workflow

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. The server name is prepended to the subject text on the email. 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 available to add text for the email message. Use the undo and redo buttons (↶ ↷) as needed.

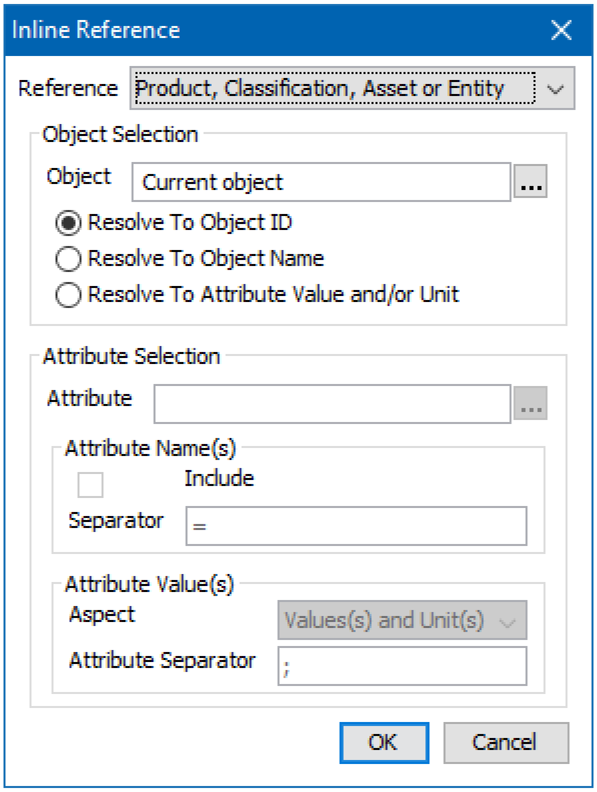
   - **Special characters** can be added using the rich text editor button ( ).

   - **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.



Inline Reference

Reference: Product, Classification, Asset or Entity

Object Selection

Object: Current object

● Resolve To Object ID

○ Resolve To Object Name

○ Resolve To Attribute Value and/or Unit

Attribute Selection

Attribute:

Attribute Name(s)

□ Include

Separator: =

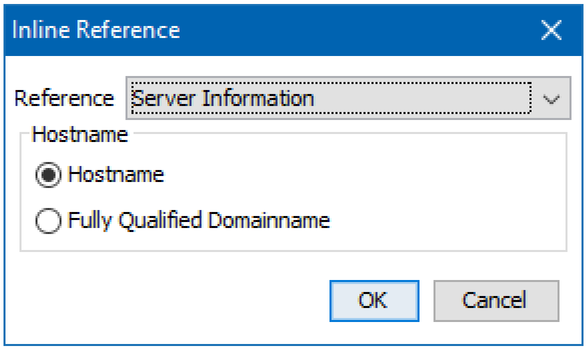
Attribute Value(s)

Aspect: Values(s) and Unit(s)

Attribute Separator: ;

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.



Inline Reference

Reference: Server Information

Hostname

● Hostname

○ Fully Qualified Domainname

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 operation 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.

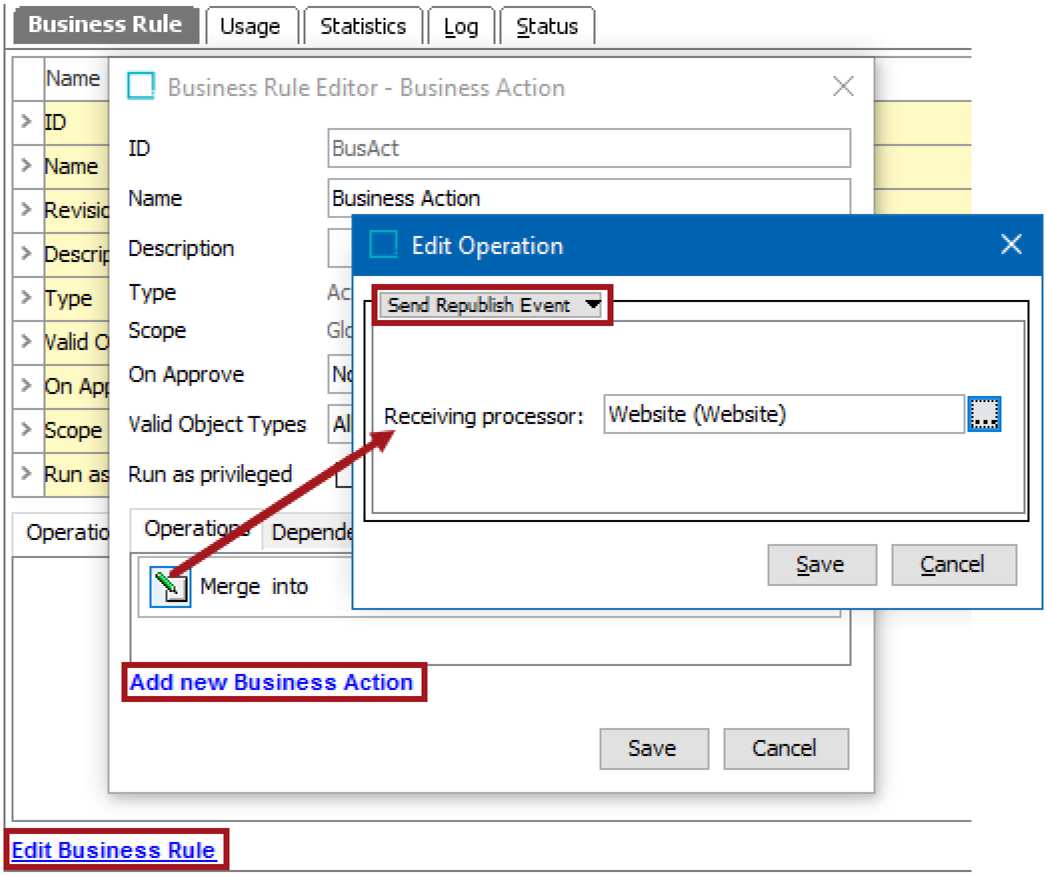
This operation can also be run via a Bulk Update and sample data is provided in the **Send Republish Event Operation** topic in the **Bulk Updates** documentation.

Prerequisites

Before using this operation:

1. Ensure the event queue is enabled and set to Read Events as defined in the **Maintaining Event Queues** topic.
2. Create a business rule as defined in the **Creating a Business Rule, Function, 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.

   This can be an event processor, outbound integration endpoint with an event queue, or an event queue directly.
3. Click the **Save** button to add the operation to the business rule editor.

Business Action: Set Attribute Value

This operation 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.

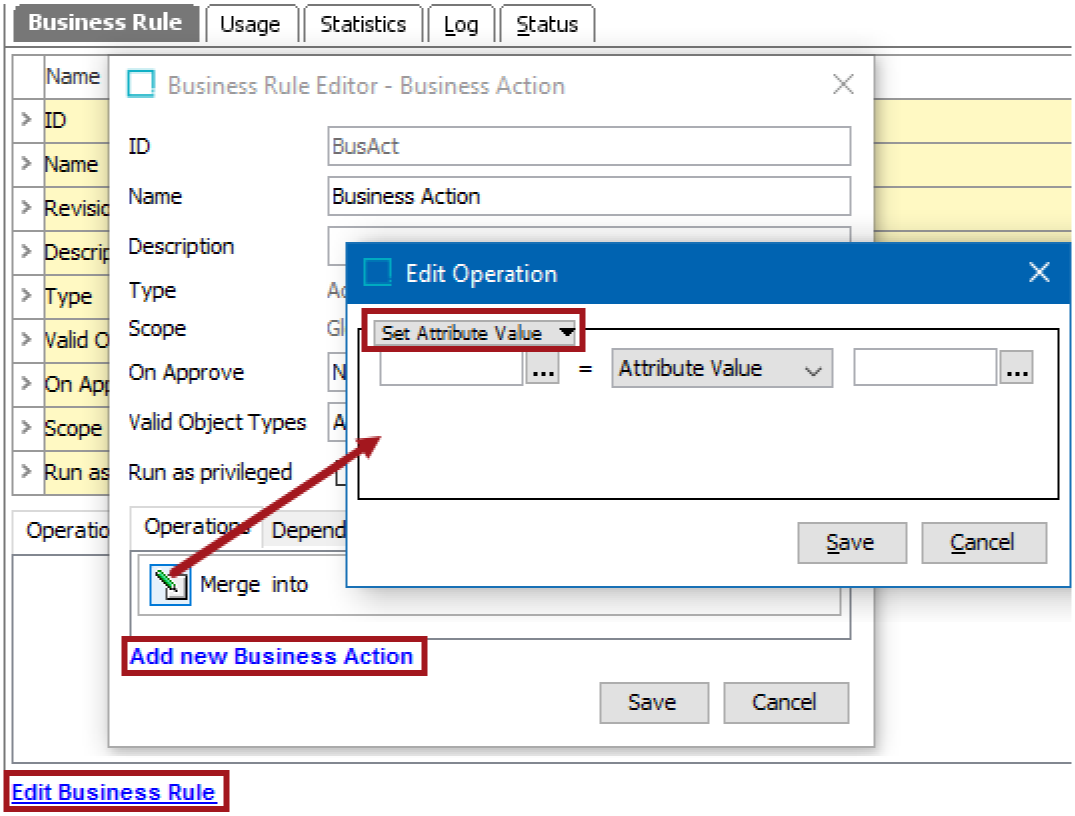
This operation is similar to one in Bulk Update, as described in the **Attribute Values: Set Value Operation** topic in the **Bulk Updates** documentation.

Prerequisite

Before using this operation:

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

Configuration



ID: BusAct

Name: Business Action

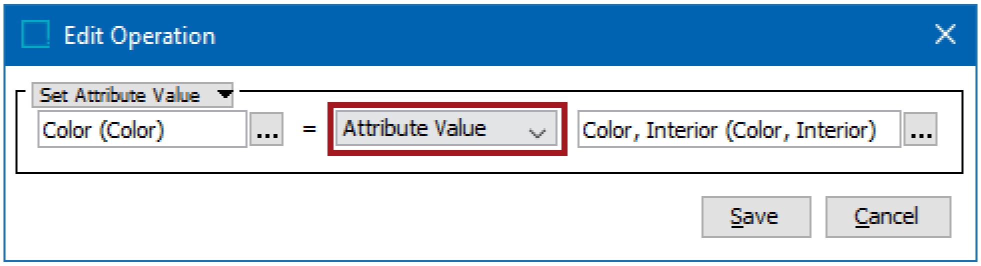
Set Attribute Value

= Attribute Value

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.



Edit Operation

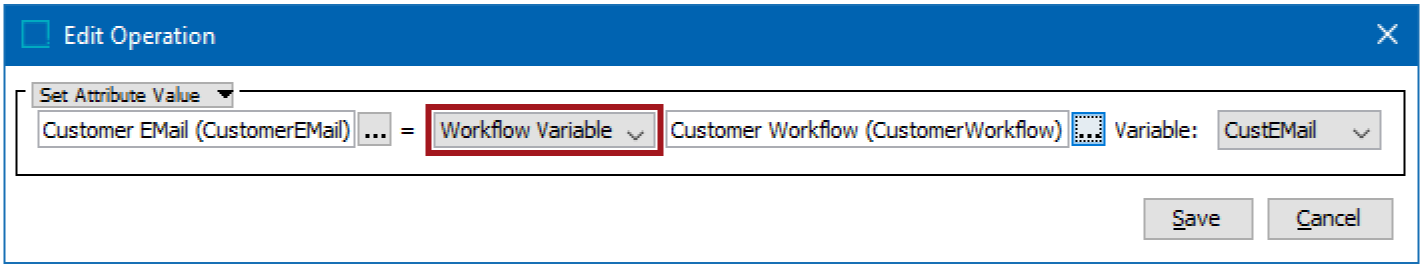
Set Attribute Value

Color (Color) ... = Attribute Value Color, Interior (Color, Interior) ...

- When **Workflow Variable** is selected, the select source workflow and select source variable parameters are displayed. For more information on workflow variables, refer to 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.



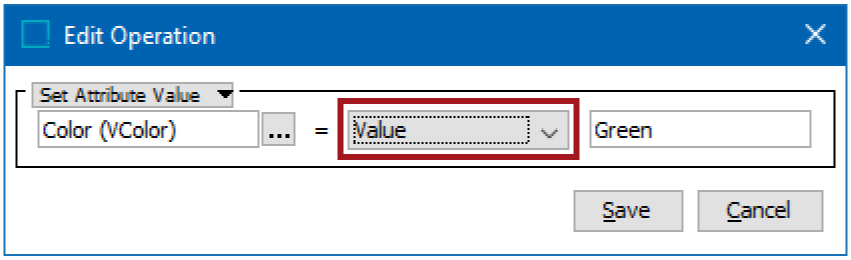
Edit Operation

Set Attribute Value

Customer EMail (CustomerEMail) ... = Workflow Variable Customer Workflow (CustomerWorkflow) ... Variable: CustEMail

- 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.



Edit Operation

Set Attribute Value

Color (VColor) = Value Green

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

Business Action: Set Name

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

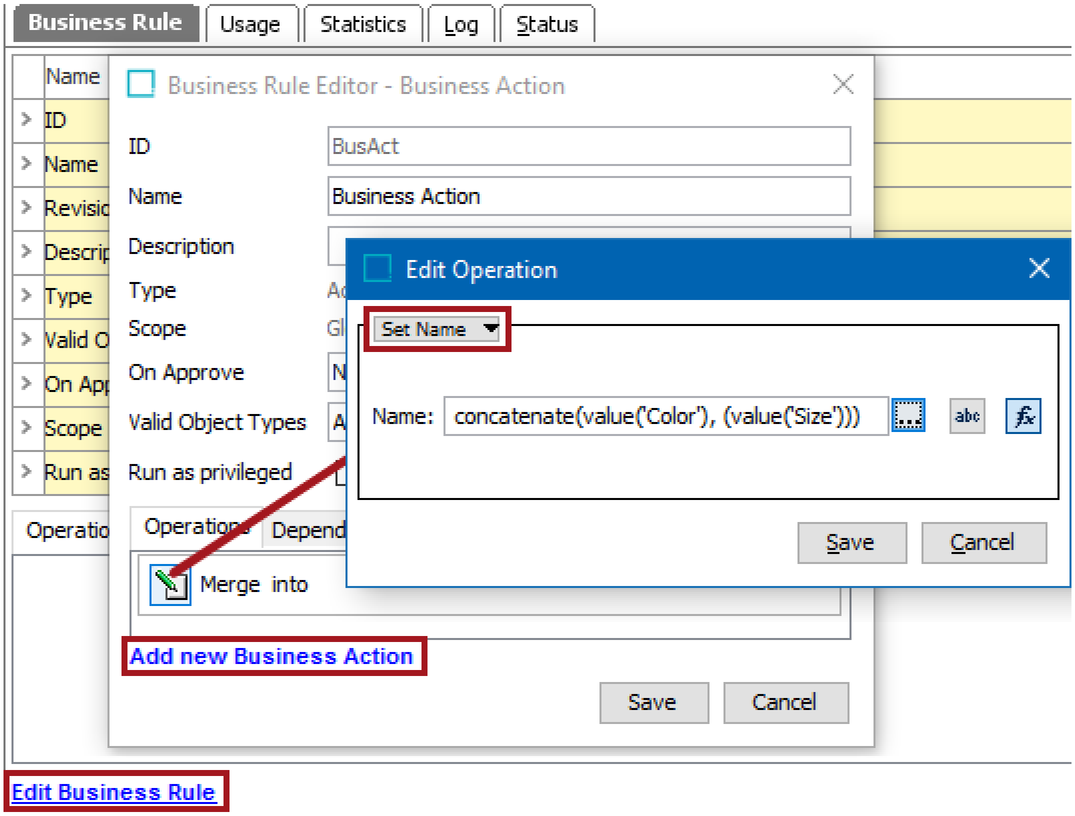
For example, when the color and size attribute values are concatenated to make the name of the object, use this action and a STEP function to generate that name.

This operation can also be run via a Bulk Update, as described in the **Set Name Operation** topic in the **Bulk Updates** documentation.

Prerequisite

Before using this operation, create a business rule as defined in the **Creating a Business Rule, Function, or Library** topic.

Configuration



ID: BusAct

Name: Business Action

Name: concatenate(value('Color'), (value('Size')))

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.
   - For manual selection, click the 'abc' button (abc), type in a constant value that will be used to update the name.

- For function selection, click the 'fx' button (fx), then click the ellipsis button (...) to display the Function Editor dialog. Write a function to set a name and click the **Select** button. For more information, refer to the **Using Function Editor** topic in the **Resource Materials** online help.

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

Business Action: Set Object Type

This operation 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 operation to choose the new object type and run it via a business rule.

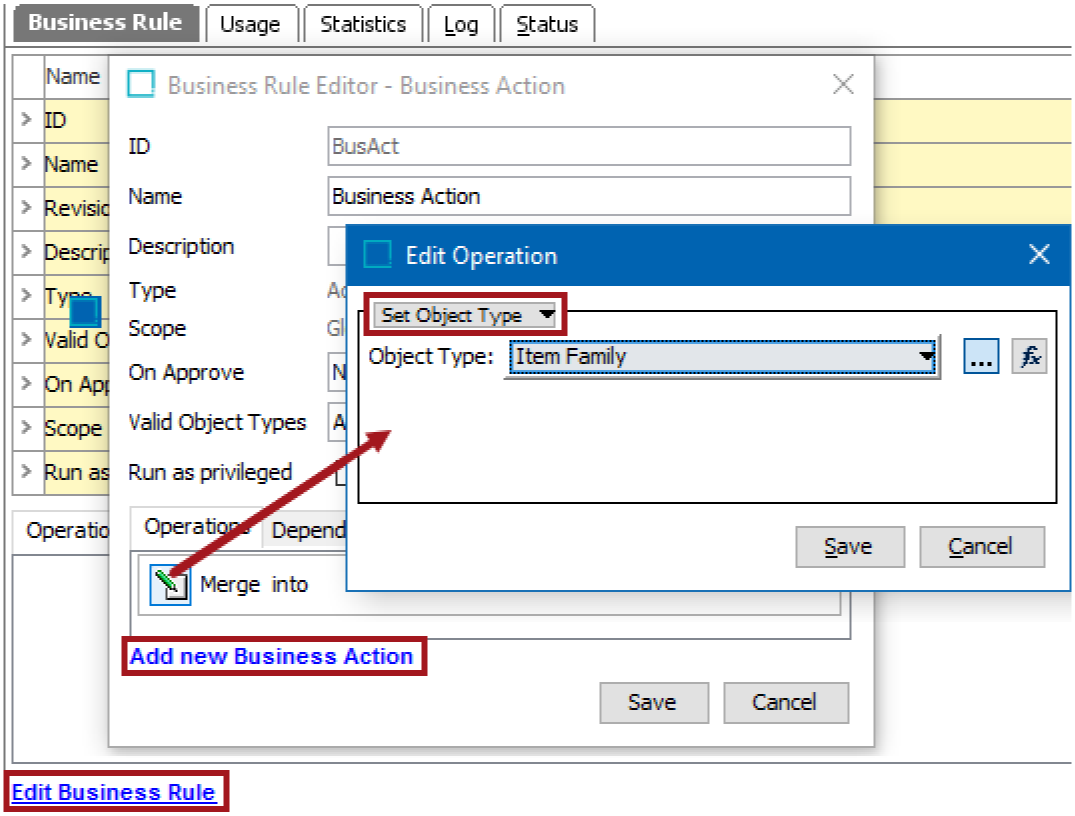
This operation can also be run via a Bulk Update, as described in the **Set Object Type Operation** topic in the **Bulk Updates** documentation.

Prerequisites

Before using this operation:

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, Function, or Library** topic.

Configuration



Business Rule Editor - Business Action

ID: BusAct

Name: Business Action

Edit Operation

Set Object Type

Object Type: Item Family

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 object type.

- For manual selection, click the dropdown to display the available object types in your system. Choose an item from the dropdown.

  When available, you can choose the **Other...** option to display the Choose Object Type dialog. Then use the Browse or Search tabs to select an object type and click the **Select** button.

- For function selection, click the 'fx' button (fx), then click the ellipsis button (...) to display the Function Editor dialog and write a calculation to select an object type. Click the **OK** button to save the function. For more information, refer to the **Using Function Editor** topic in the **Resource Materials** online help.

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

Business Action: Set Product to Classification Link Type

This operation 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.

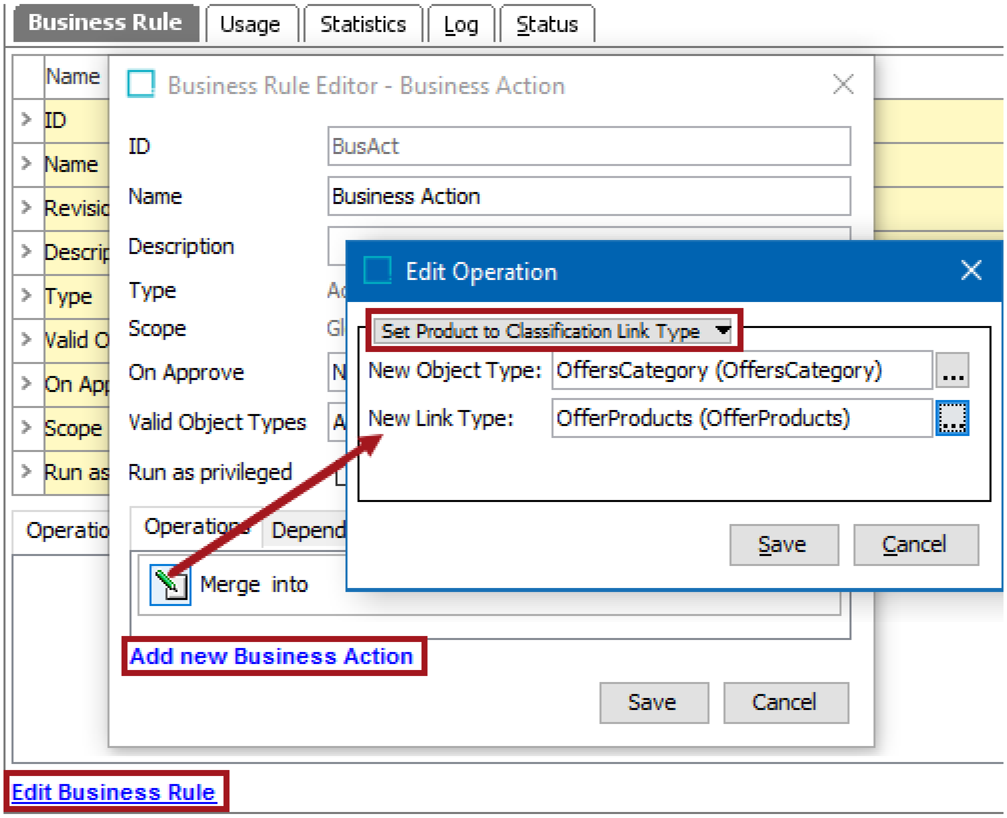
This operation can also be run via a Bulk Update and sample data is provided in the **References and Links: Set Product to Classification Link Type Operation** topic in the **Bulk Updates** documentation.

Prerequisites

Before using this operation:

1. Ensure a valid Classification Link Type and the Target Classification Object Type exists. For more information, refer to the **Reference and Link Types** topic in the **System Setup** 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, Function, or Library** topic.

Configuration



ID: BusAct

Name: Business Action

Set Product to Classification Link Type

New Object Type: OffersCategory (OffersCategory)

New Link Type: OfferProducts (OfferProducts)

1. On the Edit Operation dialog, from the dropdown select the References and Links group, then select the **Set Product to Classification Link Type** option.

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

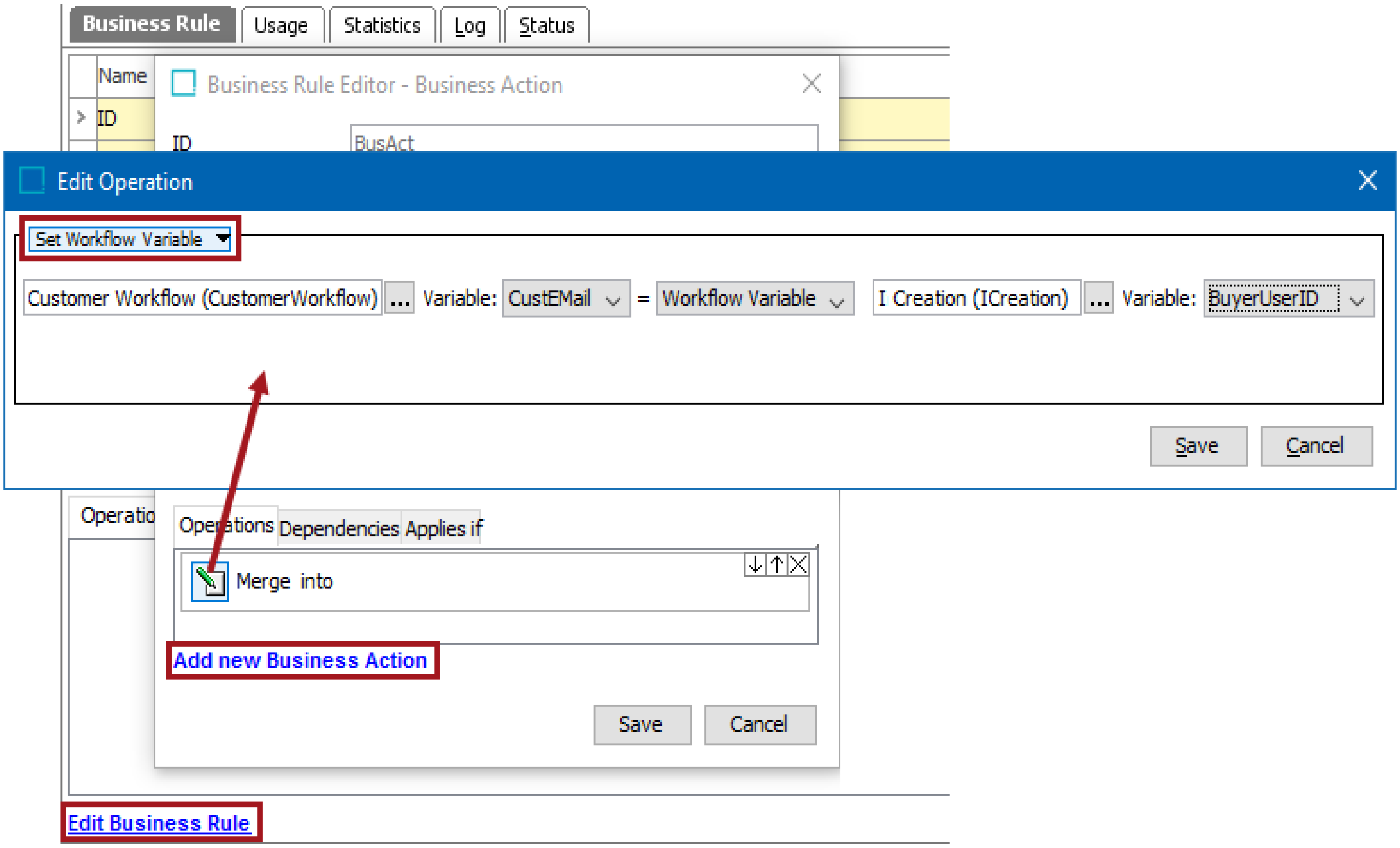
For example, a group of five users have access to a workflow, and can submit a product from the Edit state to the Review state. There requirement is to identify which of these users submitted a particular product. This action is added to a business rule to save the details to an attribute. The business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

Prerequisites

Before using this operation:

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

Configuration



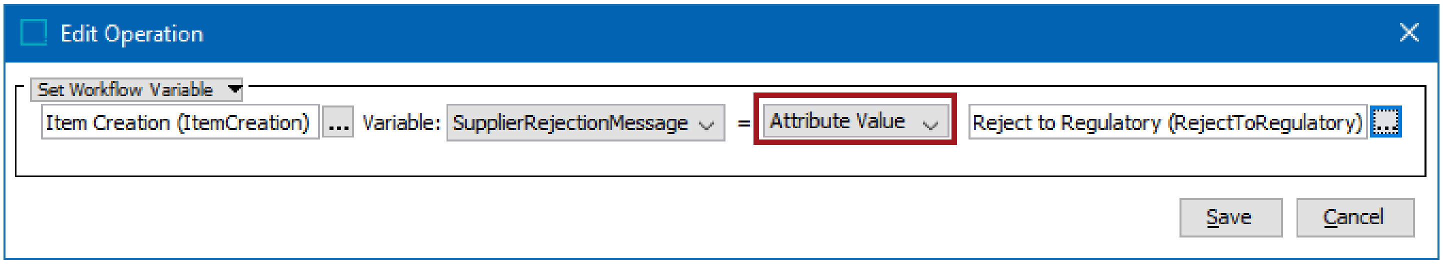
Business Rule Editor - Business Action
ID: BusAct

Edit Operation
Set Workflow Variable
Customer Workflow (CustomerWorkflow) ... Variable: CustEMail = Workflow Variable I Creation (ICreation) ... Variable: BuyerUserID

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.



Edit Operation

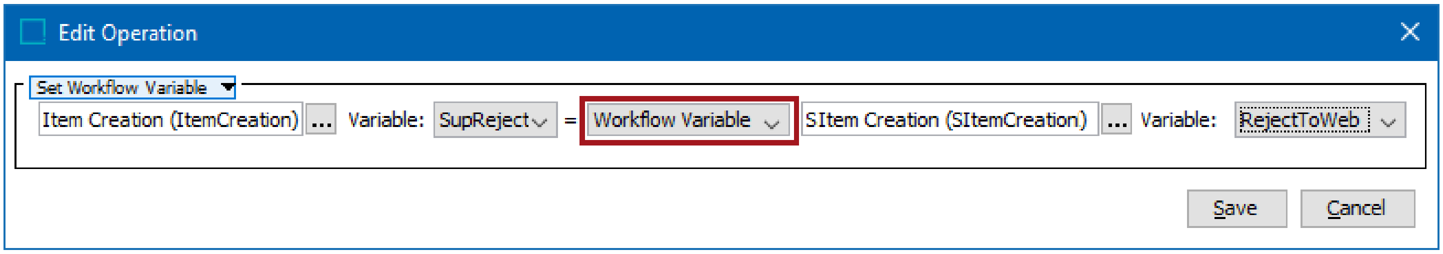
Set Workflow Variable

Item Creation (ItemCreation) ... Variable: SupplierRejectionMessage = Attribute Value Reject to Regulatory (RejectToRegulatory) ...

- 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.



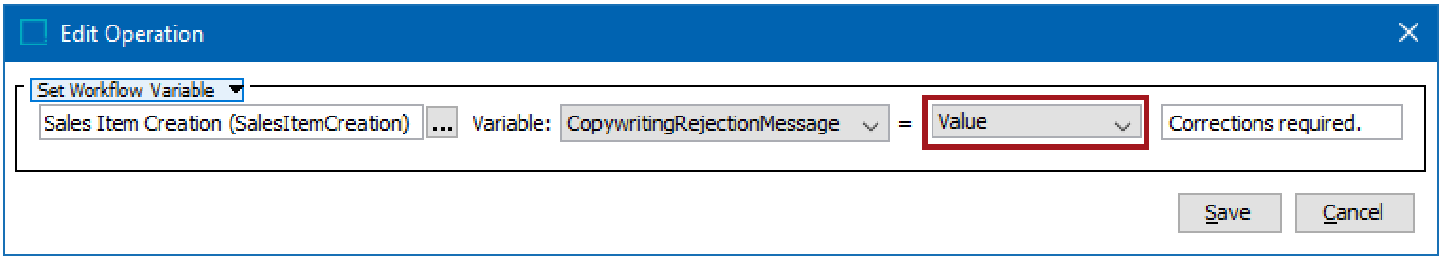
Edit Operation

Set Workflow Variable

Item Creation (ItemCreation) ... Variable: SupReject = Workflow Variable SItem Creation (SItemCreation) ... Variable: RejectToWeb

- 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.



Edit Operation

Set Workflow Variable

Sales Item Creation (SalesItemCreation) ... Variable: CopywritingRejectionMessage = Value Corrections required.

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

Business Action: Standardize Address

This operation 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.

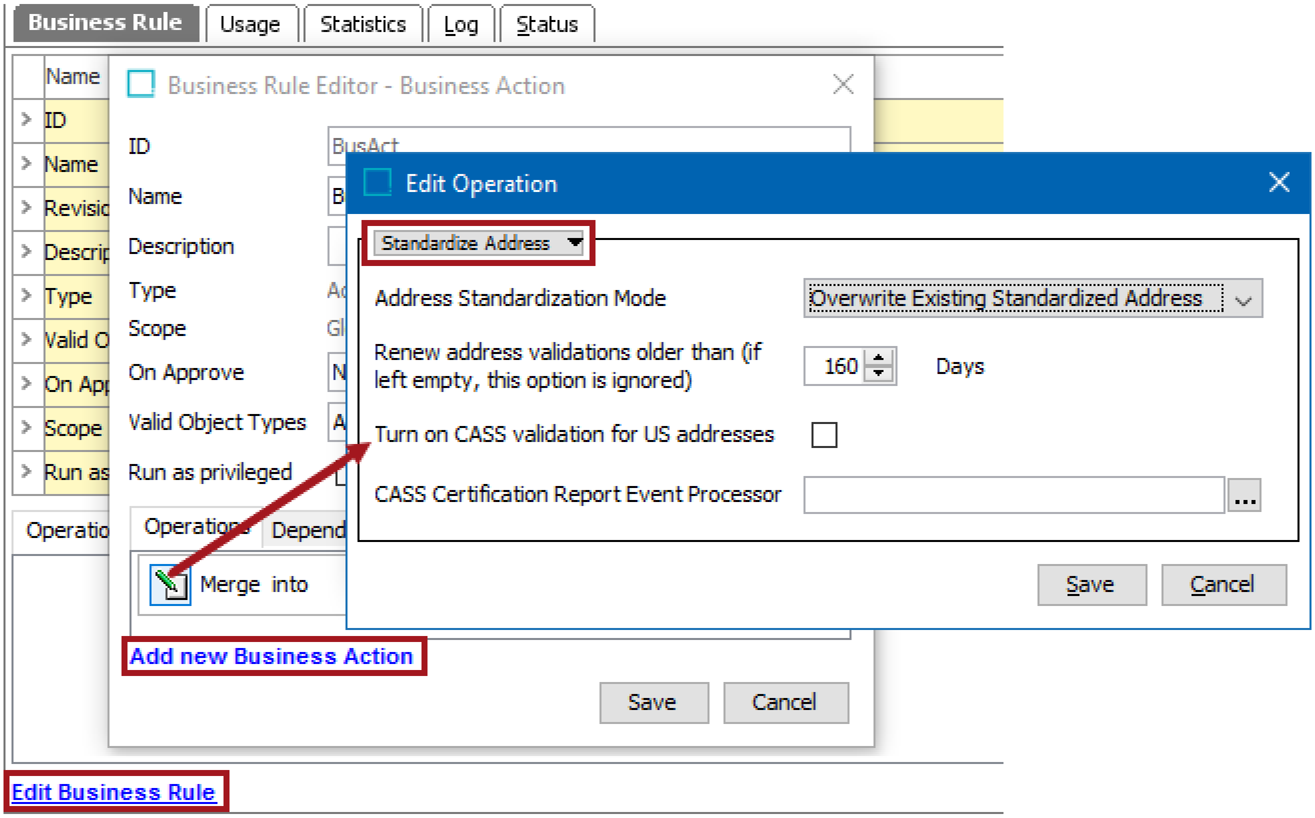
This operation can also be run via a Bulk Update, as described in the **Data Quality: Standardize Address Operation** topic in the **Bulk Updates** documentation.

Prerequisites

Before a Standardize Address operation 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, refer to 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, refer to the **CASS Address Component Model** topic in the **Loqate** documentation.
- Create a business rule as defined in the **Creating a Business Rule, Function, or Library** topic.

Configuration



Address Standardization Mode: Overwrite Existing Standardized Address

Renew address validations older than (if left empty, this option is ignored): 160 Days

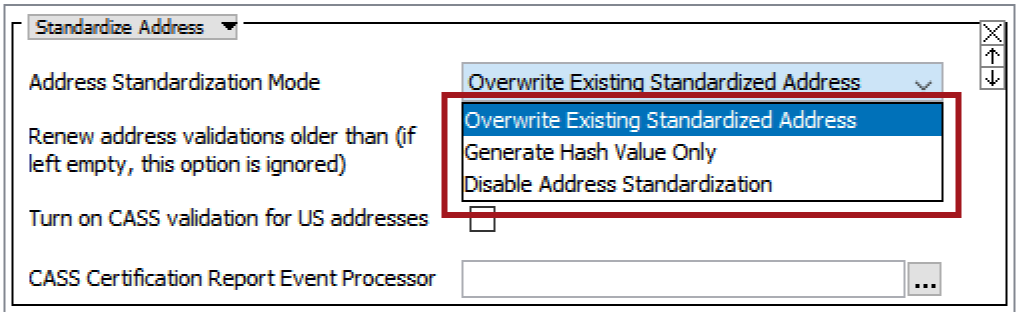
Turn on CASS validation for US addresses ☐

CASS Certification Report Event Processor: ...

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, refer to the **Loqate Integration** section of the **Data Integration** documentation.

Address Standardization Modes

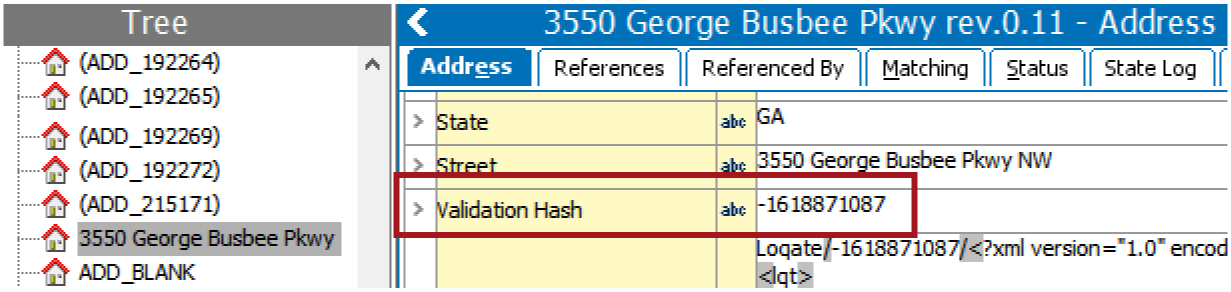
Three options are available from the Address Standardization Mode dropdown list.


Standardize Address
Address Standardization Mode
Overwrite Existing Standardized Address
Overwrite Existing Standardized Address
Generate Hash Value Only
Disable Address Standardization
Renew address validations older than (if left empty, this option is ignored)
Turn on CASS validation for US addresses
CASS Certification Report Event Processor
...

- **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.



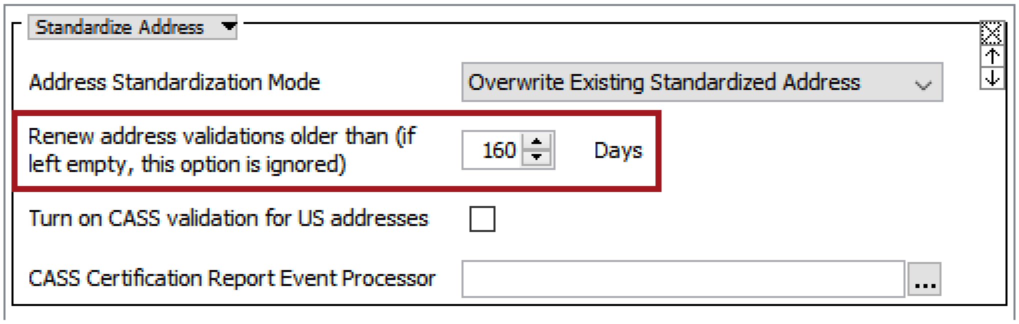
(ADD_192264)
(ADD_192265)
(ADD_192269)
(ADD_192272)
(ADD_215171)
3550 George Busbee Pkwy
ADD_BLANK

State	GA
Street	3550 George Busbee Pkwy NW
Validation Hash	-1618871087
	Loqate/-1618871087/<?xml version="1.0" encod <lqt>

- **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.

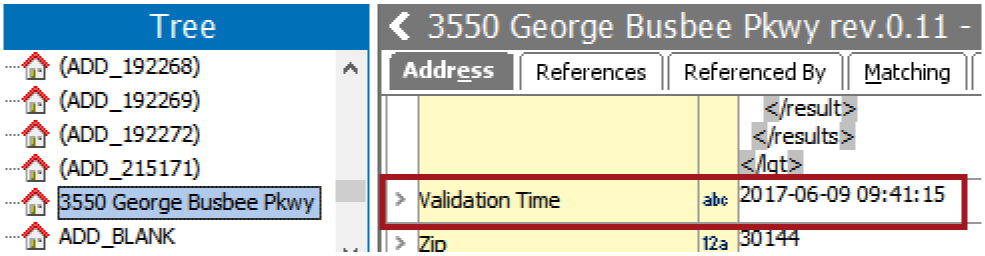
Renew Address Validations

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.'



Address Standardization Mode	Overwrite Existing Standardized Address
Renew address validations older than (if left empty, this option is ignored)	160 Days
Turn on CASS validation for US addresses	□
CASS Certification Report Event Processor	...

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



Tree
- (ADD_192268)
- (ADD_192269)
- (ADD_192272)
- (ADD_215171)
- 3550 George Busbee Pkwy
- ADD_BLANK

3550 George Busbee Pkwy rev.0.11 -

Address | References | Referenced By | Matching

```
</result>
</results>
</lqt>
```

Validation Time	abc	2017-06-09 09:41:15
Zip	12a	30144

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.

Re-Validating an Address

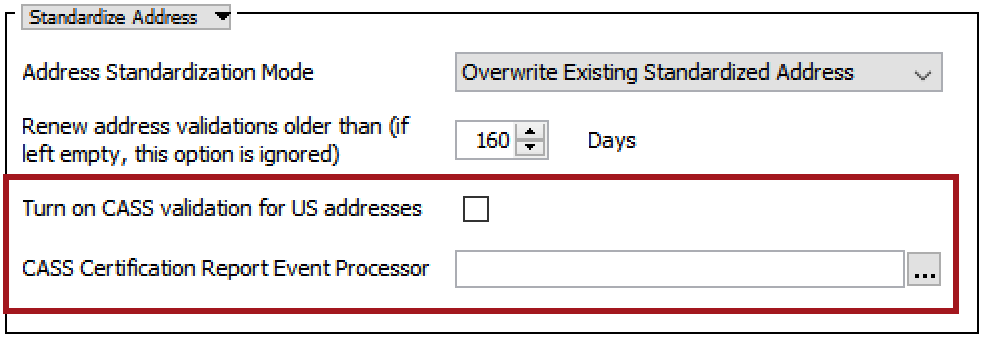
In cases where the Standardize Address business action encounters an address that has previously been validated, the Standardize Address business action will only make an actual request to Loqate if one or more of the following occurs:

- The address has been changed since it was last validated. The system determines if the address has been validated based on the value of the Validation Hash attribute.
- The last address validation failed. The system determines if address validation failed based on the value of the Validation Integration Status attribute.
- The 'Renew Address Validations' has been configured, and the address validation is older than the time period configured. The system determines if the date has elapsed based on the value of the Validation Time attribute.

This method improves performance of the Standardize Address business action as it lessens the number of calls that are made, thus reducing costs in cases where Loqate Cloud is used. For more information on these scenarios, refer to the **Address Component Model** in the **Data Integration** documentation

CASS Validation

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



Standardize Address

Address Standardization Mode	Overwrite Existing Standardized Address
Renew address validations older than (if left empty, this option is ignored)	160 Days
Turn on CASS validation for US addresses	□
CASS Certification Report Event Processor	...

- **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, refer to the **CASS Certification Report Processing Plugin Parameters and Triggers** section of the **Event Processors** documentation.

Business Action: Trigger STEP Workflow Event

This operation 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, refer to the **Business Rules and Workflows** topic in the **Workflows** documentation.

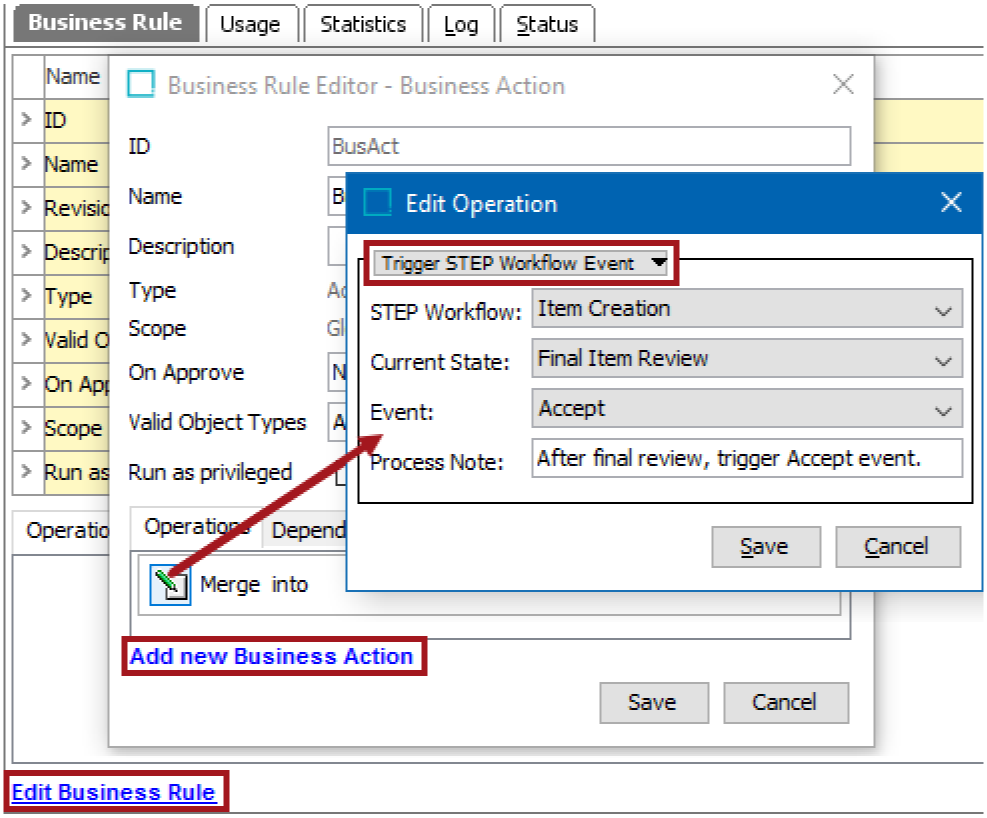
This operation can also be run via a Bulk Update, as described in the **Workflow: Trigger STEP Workflow Event Operation** topic in the **Bulk Updates** documentation.

Prerequisites

Before using this operation:

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, Function, or Library** topic.

Configuration



ID: BusAct

Name

Description

Type

Scope

On Approve

Valid Object Types

Run as privileged

Trigger STEP Workflow Event

STEP Workflow: Item Creation

Current State: Final Item Review

Event: Accept

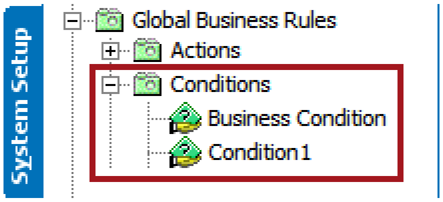
Process Note: After final review, trigger Accept event.

1. On the Edit Operation dialog, from the dropdown select the Workflow group and then the **Trigger STEP Workflow Event** option.
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.
6. Click the **Save** button to add the operation to the business rule editor.

Business Conditions

While complex business conditions can be created using either STEP Functions or JavaScript, simple conditions can be implemented using the operations in the table below. Each is described in further detail in individual sections.

To access the available business condition operations, 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, refer to **Business Actions**. To generate an output on not changing input parameters, use business functions. For more information on business functions, refer to **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, refer to **Business Condition: 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', as defined in **Business Action: Execute JavaScript**. **Important:** Although the same binds are available, changing data via an Evaluate JavaScript business condition is not supported.

Condition	Description
Function	Makes it possible to use functions to define business conditions. For more information, refer to **Business Condition: Function**
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, refer to **Business Condition: LOV Cross-Validation**.
OR Condition	Returns true if one or more of the referenced business conditions returns true. For more information, refer to the **Business Condition: OR Condition**.
Reference other Business Condition	Allows the current business condition to reference other business conditions. For more information, refer to **Business Condition: Reference other Business Condition**.
Valid Hierarchies	Specifies sub-hierarchies where the condition is true. For more information, refer to **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, refer to **Business Condition: Validate Product Variant**.

To add a business condition, refer to **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.

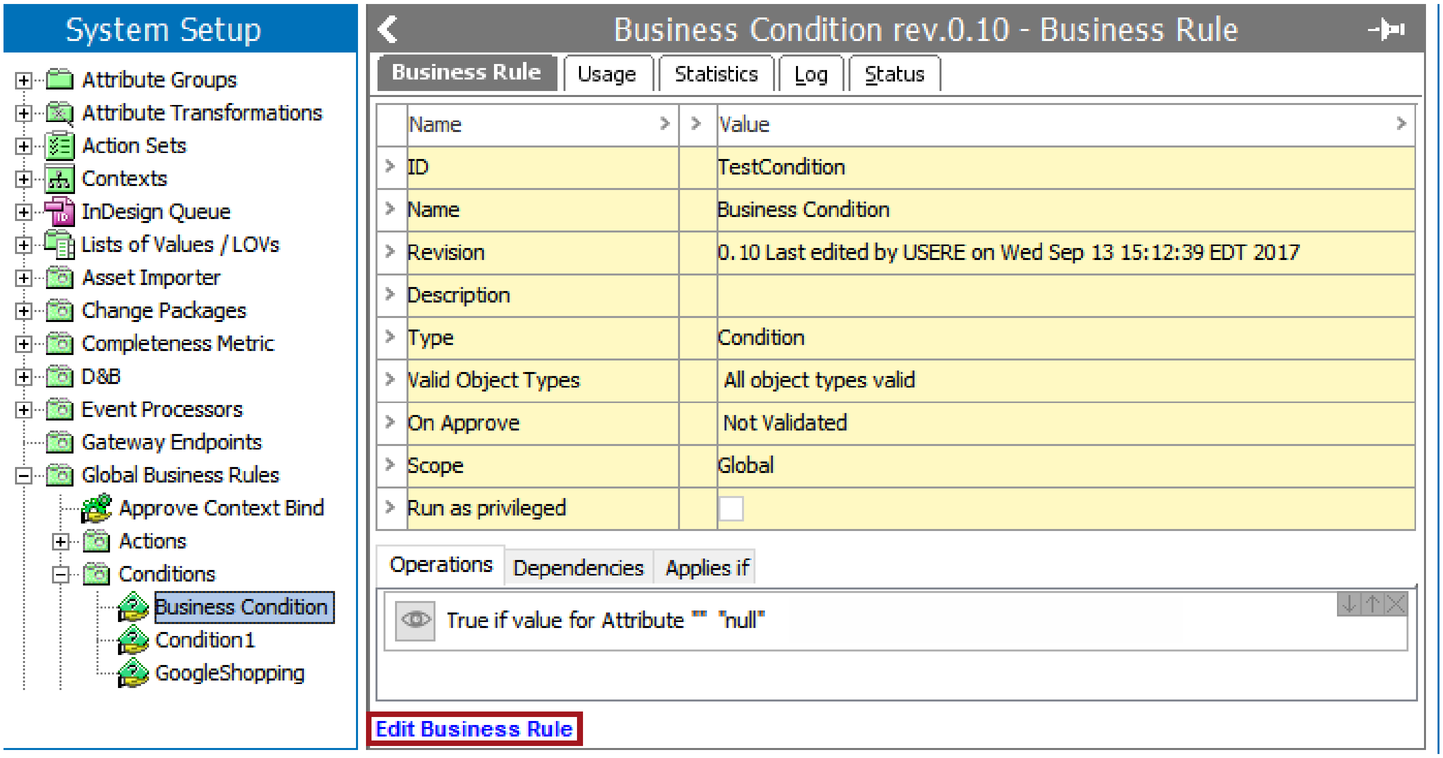
- **Required value on approval:**To determine if an attribute has a value before approval, use the 'mandatory' parameter on the attribute itself, as described in the **Mandatory Attributes** topic of the **System Setup** documentation.
- **Required value on import:** When mapping on an IIEP or Import Manager, the Mandatory checkbox on each of the mappable elements allows the importer to reject the objects when values are missing, as described in the **Mapping Options** section of the **Inbound Map Data - Map** topic of the **Data Exchange** 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** 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** 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** 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, refer to the **Creating a Business Rule, Function, 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.

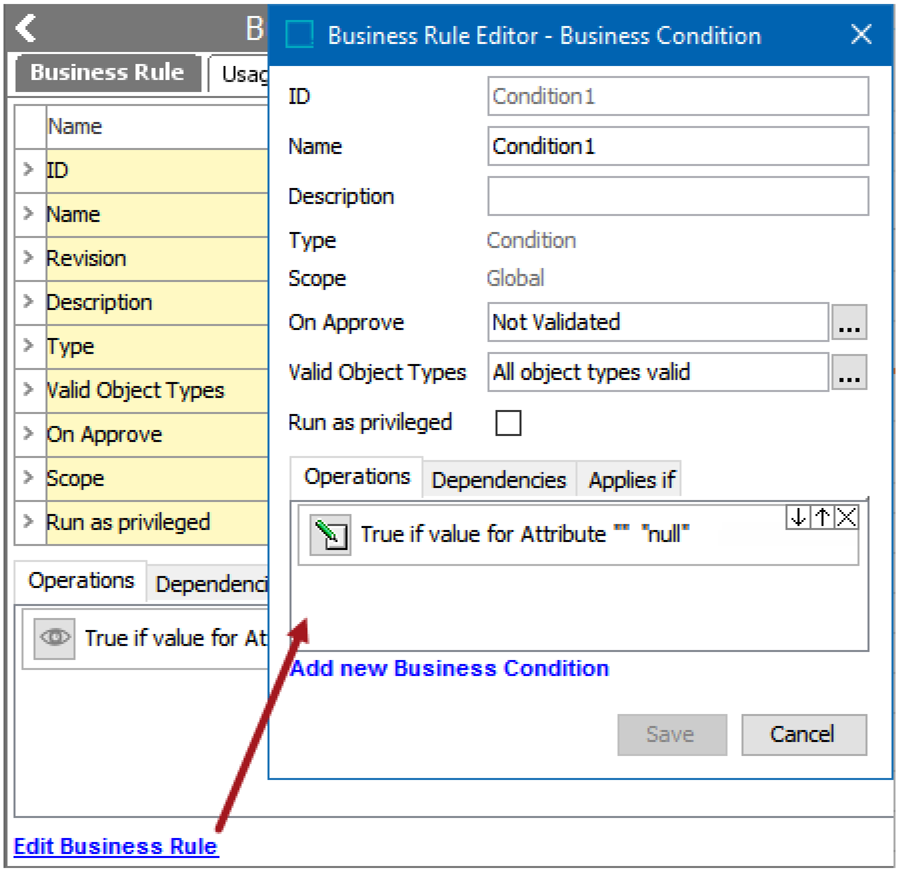


Name	Value
ID	TestCondition
Name	Business Condition
Revision	0.10 Last edited by USERE on Wed Sep 13 15:12:39 EDT 2017
Description	
Type	Condition
Valid Object Types	All object types valid
On Approve	Not Validated
Scope	Global
Run as privileged	□

True if value for Attribute "" "null"

Edit Business Rule

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



Business Rule Editor - Business Condition

ID: Condition1

Name: Condition1

Description:

Type: Condition

Scope: Global

On Approve: Not Validated

Valid Object Types: All object types valid

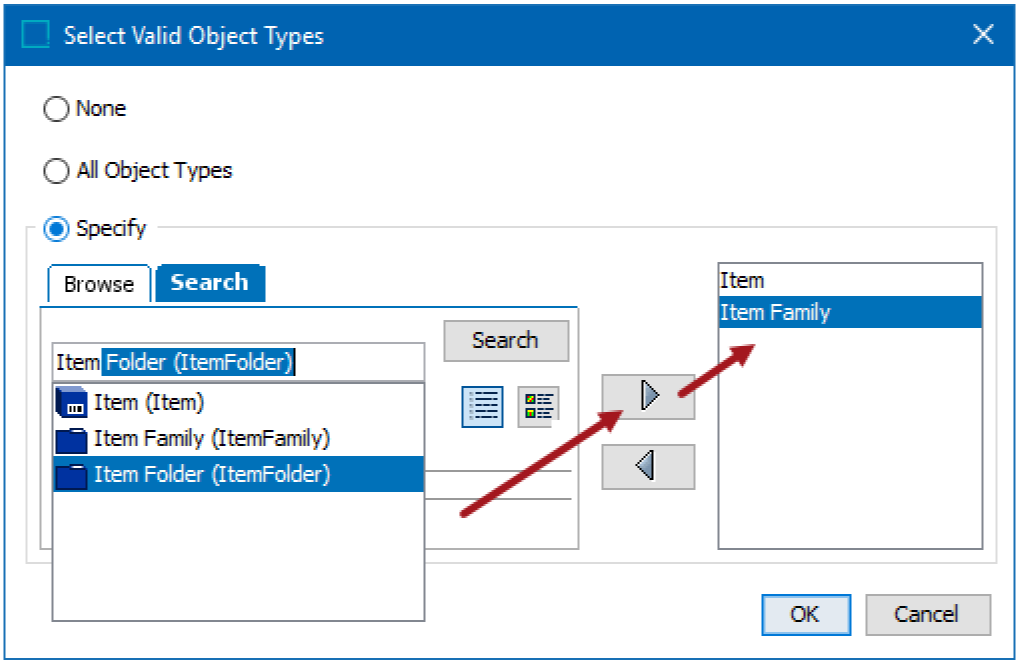
Run as privileged: □

True if value for Attribute "" "null"

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.



Select Valid Object Types

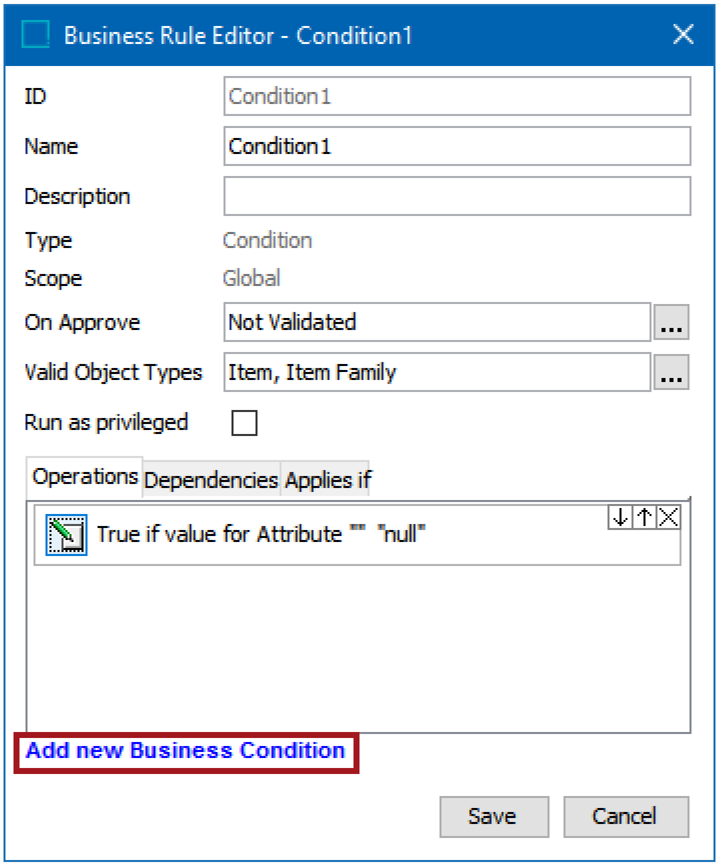
○ None

○ All Object Types

● Specify

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.

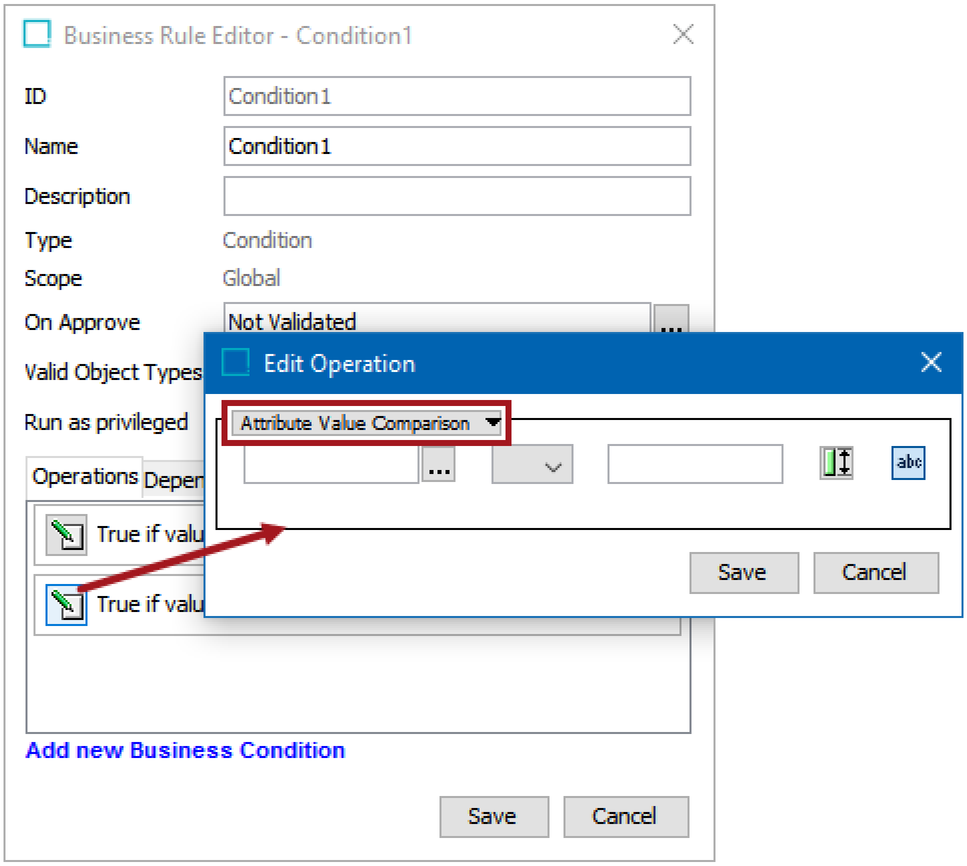


Business Rule Editor - Condition1

ID	Condition1
Name	Condition1
Description	
Type	Condition
Scope	Global
On Approve	Not Validated
Valid Object Types	Item, Item Family
Run as privileged	□

Operations
Dependencies
Applies if
True if value for Attribute "" "null"
Add new Business Condition
Save
Cancel

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



Business Rule Editor - Condition1

ID: Condition1

Name: Condition1

Description:

Type: Condition

Scope: Global

On Approve: Not Validated

Valid Object Types

Run as privileged

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.
- Multiple conditions are evaluated in order using an AND statement. This means that when a condition fails, the remaining conditions are not evaluated, nor are all of the failures reported.

  If needed, bind multiple conditions in JavaScript to evaluate all and use the JavaScript `return` function or a Logger bind to record the desired results. Refer to the **Business Action, Condition, and Function Binds** topic in the online help **Resource Materials** documentation.

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

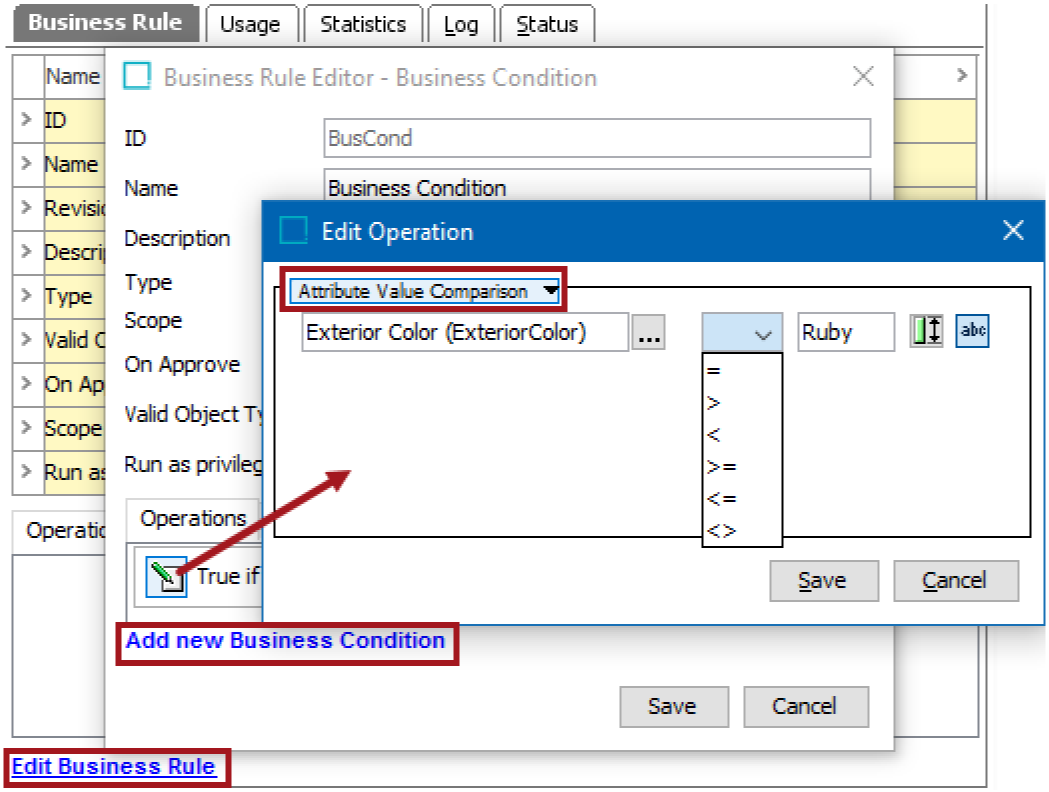
- Numeric comparison is used when both values are number based (fractions, integers, or numbers).
- When numerical values are compared with units, the system attempts to convert the values to base units.
- Multivalued attributes and attributes of the validation base type **Embedded Number** are not supported.

Prerequisites

Before using this business condition:

1. Create a business rule as defined in the **Creating a Business Rule, Function, or Library** topic.
2. Edit the business rule to configure the operation as defined in **Editing a Business Rule or Function** topic.

Configuration



ID: BusCond

Name: Business Condition

Attribute Value Comparison

Exterior Color (ExteriorColor) ... Ruby

Add new Business Condition

Edit Business Rule

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 attribute 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, refer to the **Execute JavaScript** section of the **Business Rules** documentation.

Using JavaScript in business rules can include:

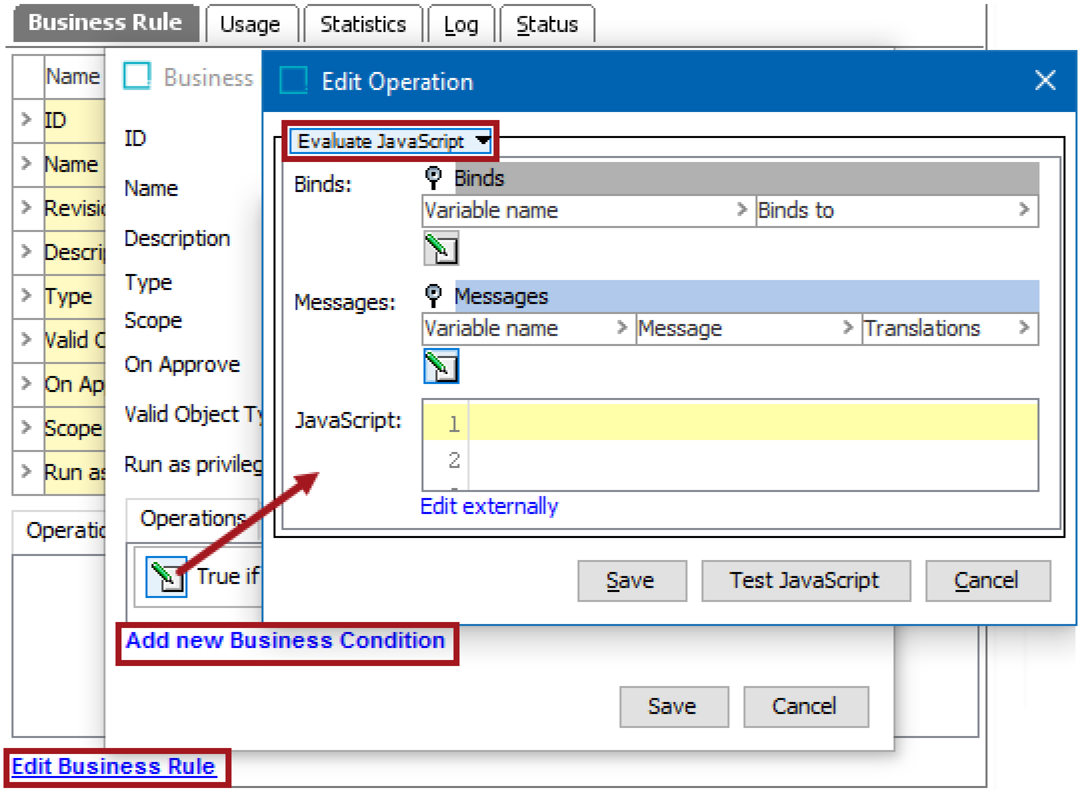
- **Evaluate JavaScript** operation is used for conditions. For more information about business rule conditions, refer to **Business Conditions**.
- **JavaScript Binds** for both action. condition. and function operations. Binds provide access to the data being worked on. For more information, refer to **JavaScript Binds** in the **Resource Materials** online help.
- **Business Libraries** is a set of JavaScript functions that can be reused in multiple business rules. For more information, refer to **Business Libraries**.
- **Scripting API** documentation is accessed via the **STEP API Documentation**, at [system]/sdk or from the Start Page.

Prerequisites

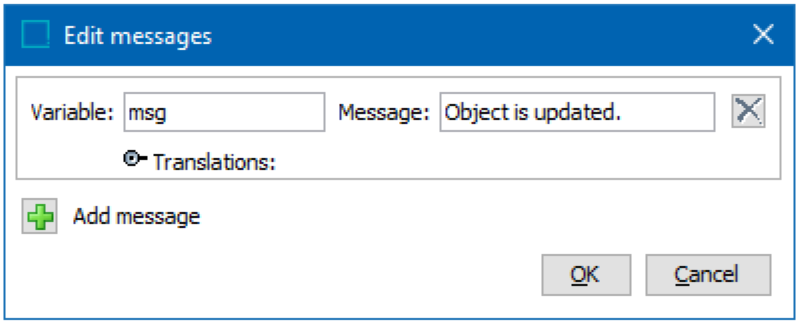
Before using this business condition:

1. Understand the implications of using JavaScript as defined in the **Java vs. JavaScript** topic in the **Resource Materials** online help.
2. Understand the considerations that should be taken as defined in the **JavaScript Considerations** topic in the **Resource Materials** online help.
3. Create a business rule as defined in the **Creating a Business Rule, Function, or Library** topic.
4. Edit the business rule to configure the operation as defined in **Editing a Business Rule or Function** 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 in the **Resource Materials** online help.
3. For the **Messages** parameter, create messages needed for the JavaScript code as follows:
   - Click the **Edit** button (✎) to display the Edit Messages dialog.



Variable: msg

Message: Object is updated.

   - 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, refer to 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.

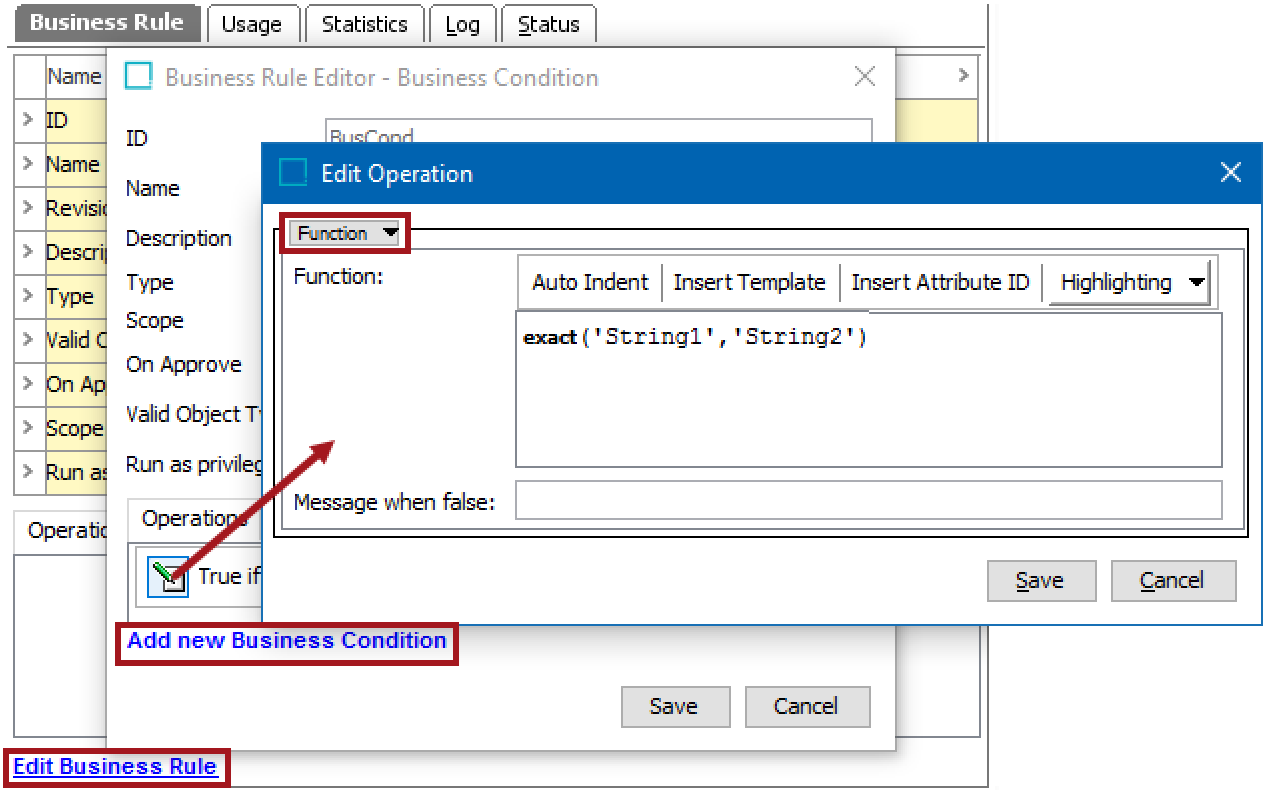
For more information, refer to the **STEP Functions** topic of the **Resource Materials** online help.

Prerequisites

Before using this business condition:

1. Create a business rule as defined in the **Creating a Business Rule, Function, or Library** topic.
2. Edit the business rule to configure the operation as defined in **Editing a Business Rule or Function** topic.

Configuration

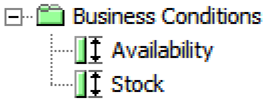


```
exact('String1','String2')
```

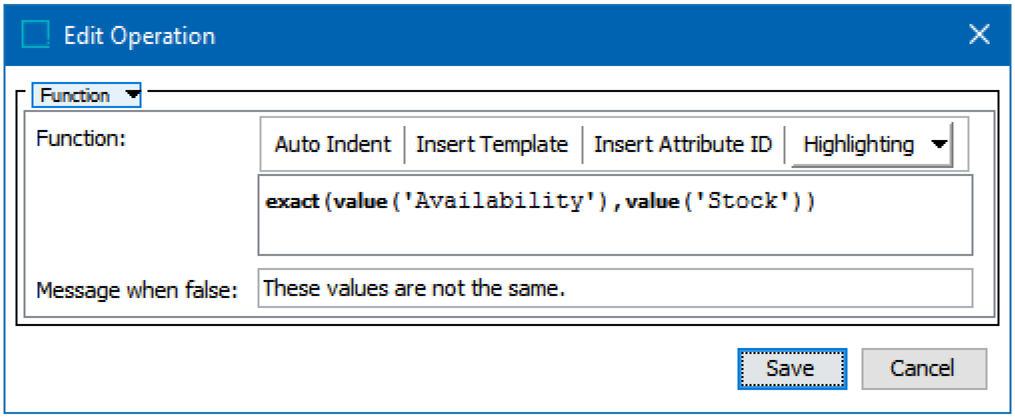
1. On the Edit Operation dialog, select **Function** from the dropdown.
2. For the **Function** parameter, add the function in the same way as you would in the Function Editor. For more information, refer to the **Function Editor** topic or **Using Function Editor** in the **Resource Materials** online help.
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**.
4. 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:

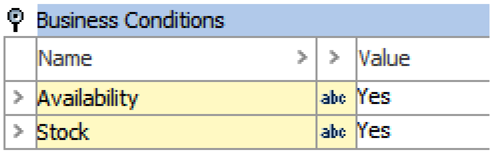


Function:

```
exact(value('Availability'),value('Stock'))
```

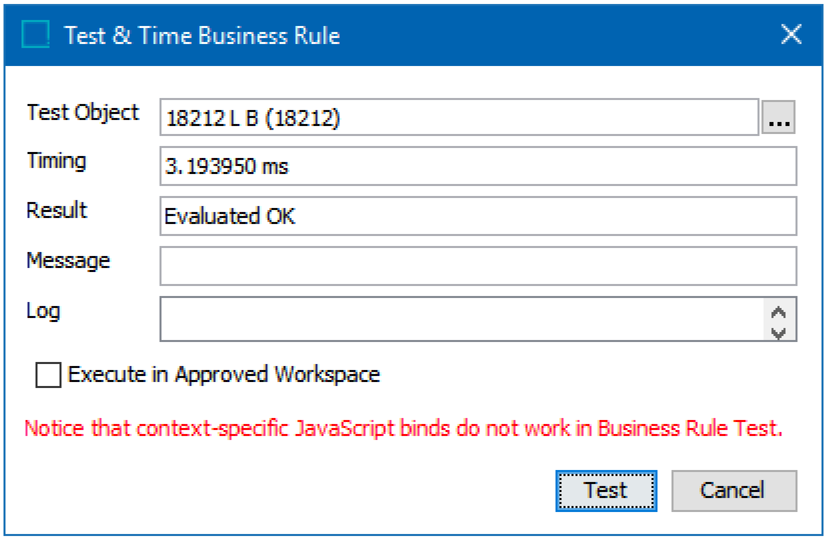
Message when false: These values are not the same.

Now that the business condition is set up, navigate to an object with these attributes and set both values to the same.



Name		Value
Availability	abc	Yes
Stock	abc	Yes

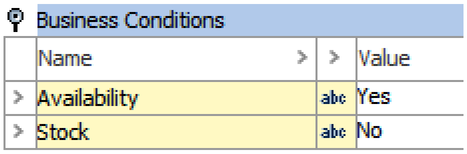
Return to the 'Global Business Rules' in the **System Setup** tab. Right-click on the business condition, and select 'Test Business Rule.' On the 'Test & Time Business Rule' dialog, select the test object where the Availability and Stock values are the same. Select 'Test.' The results will show that these attributes are the same.



Test Object: 18212 L B (18212)
Timing: 3.193950 ms
Result: Evaluated OK
Message:
Log:
□ Execute in Approved Workspace

Notice that context-specific JavaScript binds do not work in Business Rule Test.

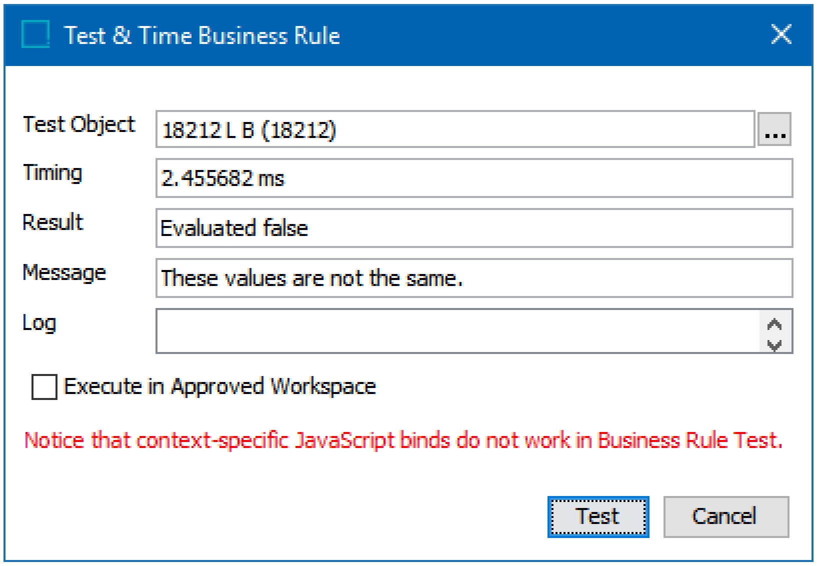
Now, return back to the object and change one of the values. In this example, Stock is set to 'No.'



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.



Test & Time Business Rule

Test Object	18212 L B (18212)
Timing	2.455682 ms
Result	Evaluated false
Message	These values are not the same.
Log	

☐ Execute in Approved Workspace

Notice that context-specific JavaScript binds do not work in Business Rule Test.

Business Condition: No Potential Duplicates

The No Potential Duplicates condition uses rank scores and the threshold from an identifying matching algorithm to determine if the current node has any potential matches. If no matches are found, it will return 'true' or otherwise 'false'.

For instance, the condition is used in a Find Similar solution to prevent transition in workflows before any potential duplicates have been resolved.

For more information, refer to the **Find Similar in Workflows** topic in the **Matching, Linking, and Merging** documentation.

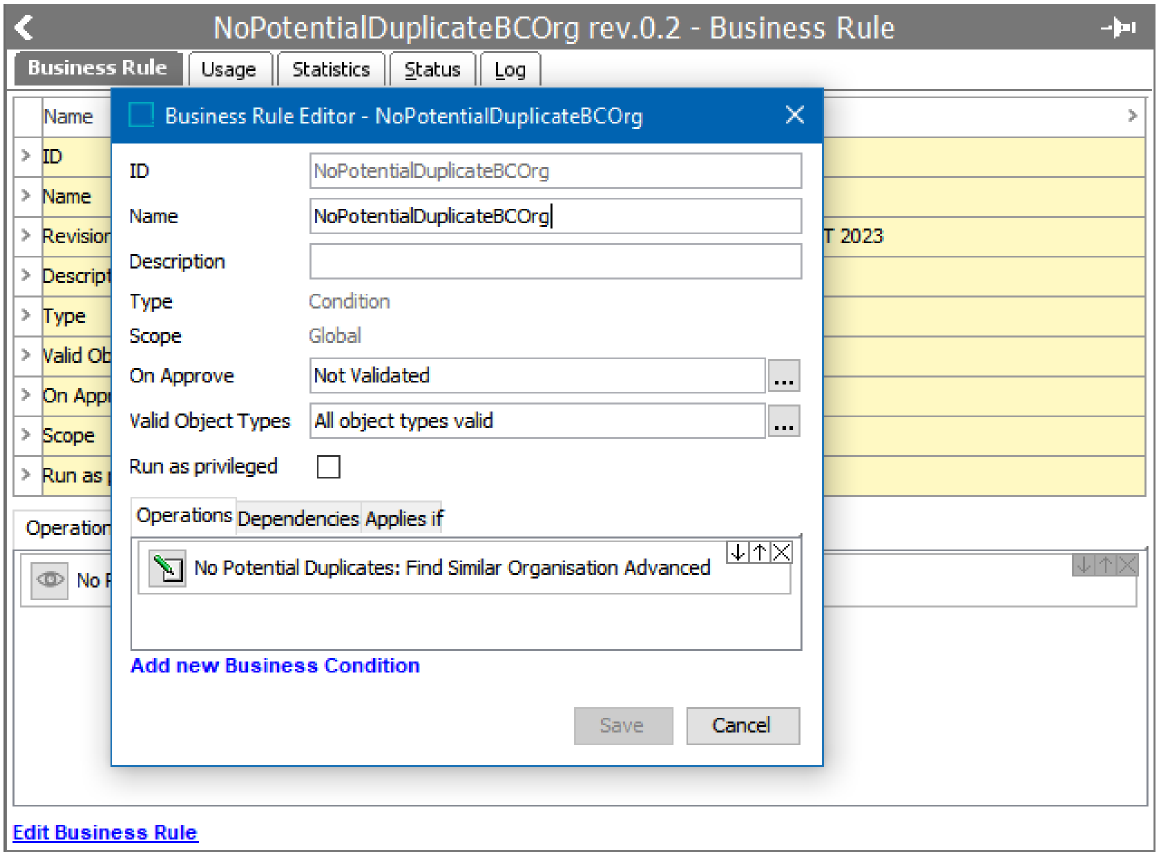
Prerequisites

Before using this business condition:

1. Create a business rule as defined in the **Creating a Business Rule, Function, or Library** topic.
2. Edit the business rule to configure the operation as defined in **Editing a Business Rule or Function** topic.

Configuration

1. On the ‘Business Rule’ tab, select ‘Edit Business Rule’.
2. On the ‘Business Rule Editor’, select ‘Add new Business Condition’.
3. Click the edit operation button.


NoPotentialDuplicateBCOrg rev.0.2 - Business Rule

Business Rule Editor - NoPotentialDuplicateBCOrg

ID: NoPotentialDuplicateBCOrg

Name: NoPotentialDuplicateBCOrg

Description:

Type: Condition

Scope: Global

On Approve: Not Validated

Valid Object Types: All object types valid

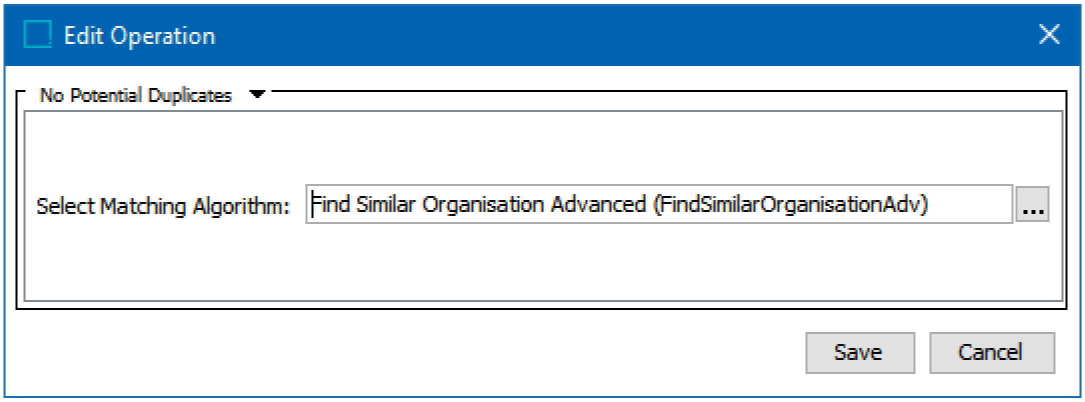
Run as privileged: □

Operations: No Potential Duplicates: Find Similar Organisation Advanced

Add new Business Condition

4. On the ‘Edit Operation’ dialog, select ‘No Potential Duplicates’ from the dropdown.

5. In the ‘Select Matching Algorithm’ field, select the corresponding matching algorithm.


Edit Operation

No Potential Duplicates

Select Matching Algorithm: Find Similar Organisation Advanced (FindSimilarOrganisationAdv)

Business Condition: 'NOT' Condition

The business condition type "NOT references another condition and inverts the results of that condition so that 'true' becomes 'false' and 'false' becomes true'.

For example, using a 'NOT' business condition enables users to invert the result of a 'No Potential Duplicates' business condition used on a workflow transition in a Find Similar process to only allow transitions if there are potential duplicates.

It is possible to override the return messages from the inverted business condition.

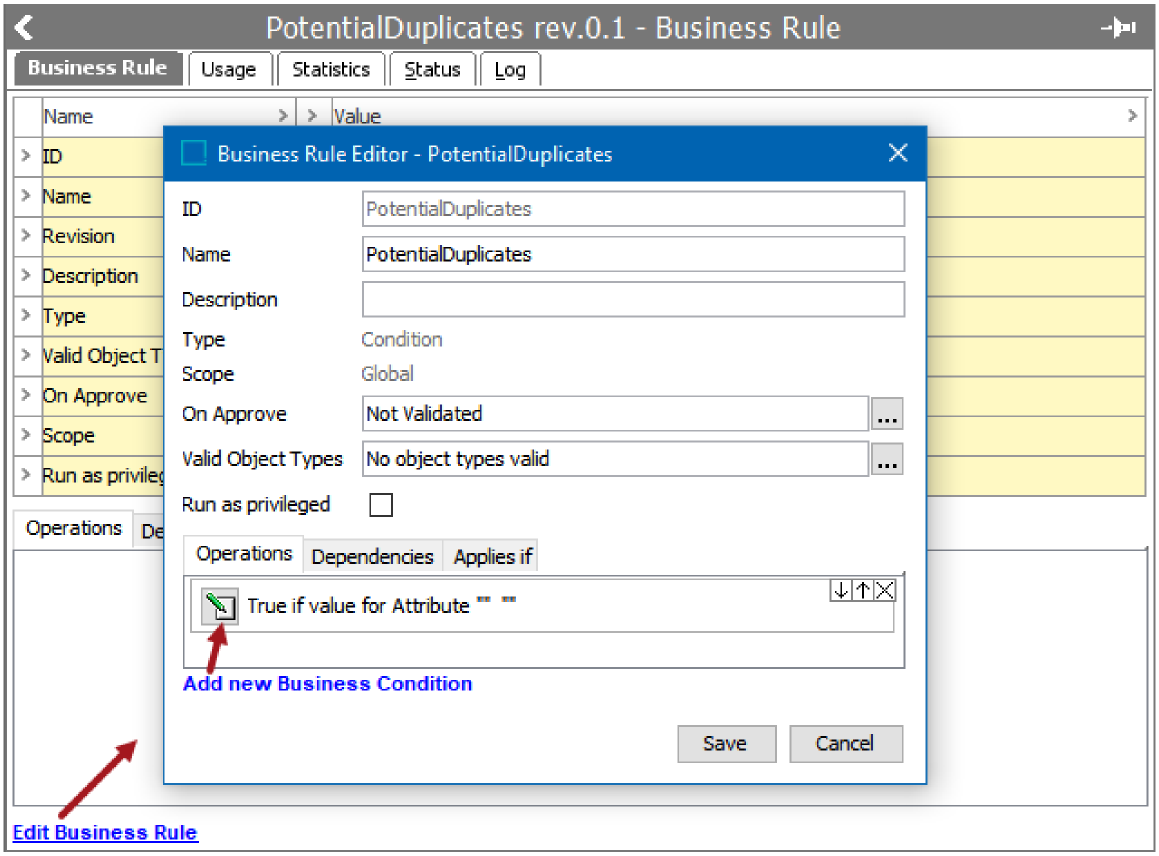
Prerequisites

Before using this business condition:

1. Create a business rule as defined in **Creating a Business Rule, Function, or Library** topic.
2. Edit the business rule to configure the operation as defined in **Editing a Business Rule or Function** topic.

Configuration

1. Create a new business condition, e.g., 'PotentialDuplicates' as in this example.
2. On the 'Business Rule' tab, click on 'Edit Business Rule'.
3. On the 'Business Rule Editor', select 'Add new Business Condition'.
4. Click the 'True if value for Attribute' edit operation icon.
5. On the 'Edit Operation' dialog, from the 'Attribute Value Comparison' dropdown, select the 'NOT Condition'.



ID: PotentialDuplicates

Name: PotentialDuplicates

Description:

Type: Condition

Scope: Global

On Approve: Not Validated

Valid Object Types: No object types valid

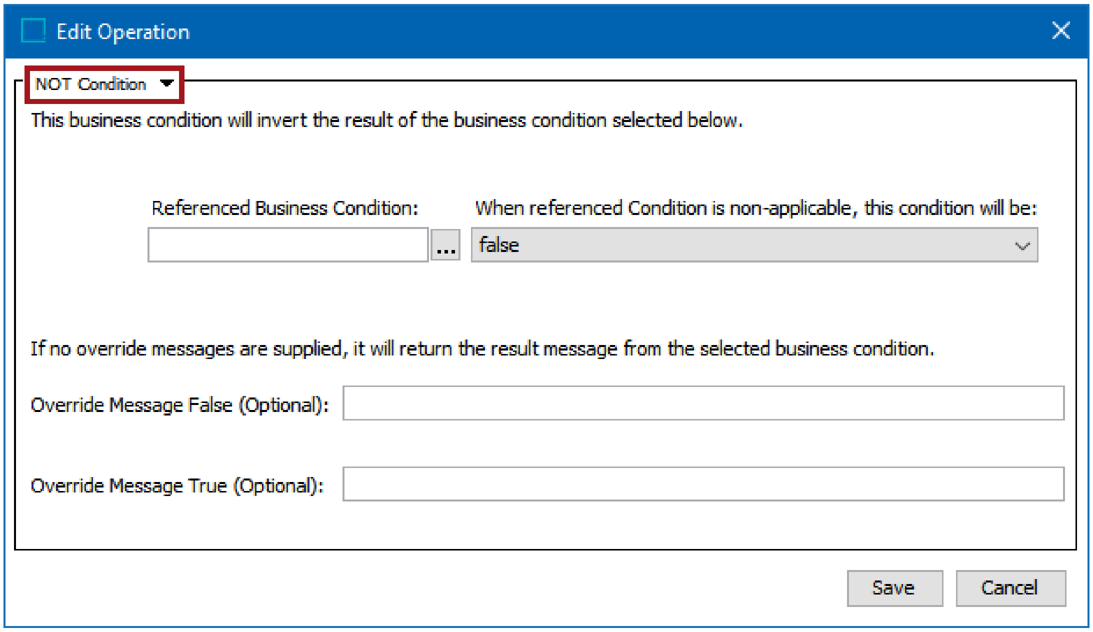
Run as privileged: □

True if value for Attribute "" ""

6. The 'Edit Operation' dialog opens. In the field 'Referenced Business Condition', select the business condition you wish to invert.

7. In the field ‘Referenced Business Condition’, select the business condition you wish to invert.

> **Note:** The 'NOT' condition allows business rule writers to override message(s) generated by the inverted business condition using the 'Override Message' fields.



This business condition will invert the result of the business condition selected below.

Referenced Business Condition:

When referenced Condition is non-applicable, this condition will be: false

If no override messages are supplied, it will return the result message from the selected business condition.

Override Message False (Optional):

Override Message True (Optional):

Business Condition: OR Condition

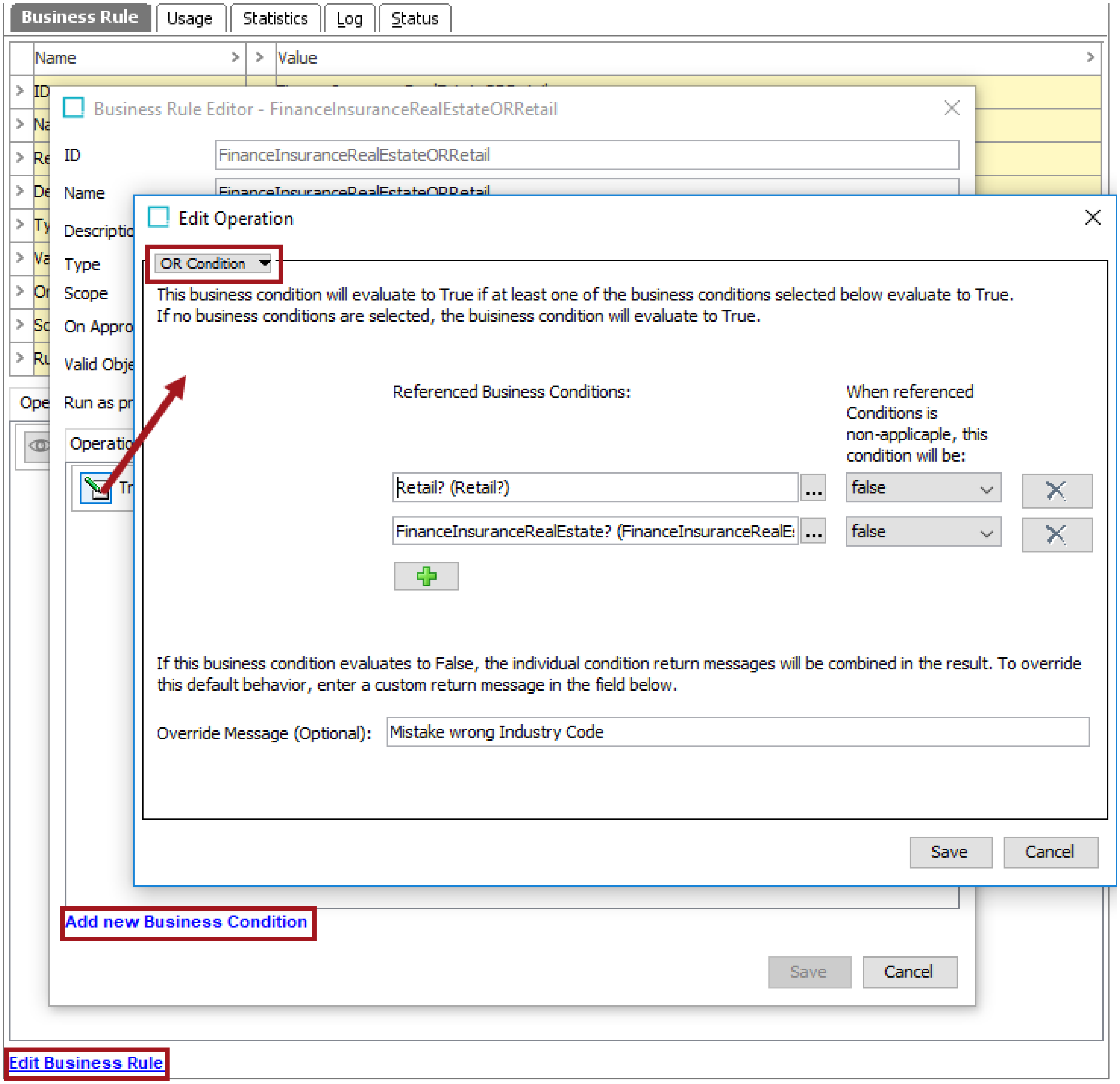
This operation references one or more other business conditions and returns True if any of the referenced conditions return True.

For example, if this operation references three other business conditions, but only two of them return true, this operation still returns true.

Prerequisites

Before using this business condition:

1. Create a business rule as defined in the **Creating a Business Rule, Function, or Library** topic.
2. Edit the business rule to configure the operation as defined in **Editing a Business Rule or Function** topic.



ID: FinanceInsuranceRealEstateORRetail

Name: FinanceInsuranceRealEstateORRetail

This business condition will evaluate to True if at least one of the business conditions selected below evaluate to True. If no business conditions are selected, the buisiness condition will evaluate to True.

Referenced Business Conditions:

When referenced Conditions is non-applicable, this condition will be:

Referenced Business Conditions	When referenced Conditions is non-applicable, this condition will be:
Retail? (Retail?)	false
FinanceInsuranceRealEstate? (FinanceInsuranceRealE	false

If this business condition evaluates to False, the individual condition return messages will be combined in the result. To override this default behavior, enter a custom return message in the field below.

Override Message (Optional): Mistake wrong Industry Code

Configuration

1. On the Edit Operation dialog, select **OR Condition** from the dropdown.
2. For the **Referenced Business Condition** parameter, click the ellipsis button (...) to display the 'Select Business Condition' dialog, and browse or search for the desired business condition.

   Press the plus sign to add additional business conditions to the operation.

3. For the **When referenced Condition is non-applicable, this Condition will be** parameter, select **True** or **False** in the dropdown. This selection determines the return value of the referenced condition in cases where the referenced condition is not applicable.
4. For the **Override Message** parameter, enter a custom message to display if this operation evaluates to false. This is an optional parameter that overrides the default messaging behavior. If this parameter is not configured, in the event that all conditions evaluate to false, the individual messages will be combined into a single message.
5. Click the **Save** button to add the operation to the business rule editor.

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 is named ‘List Colors’ with values ‘Blue,’ ‘Black,’ and ‘Red.’ This is the 'Defining' attribute. Another LOV attribute is named ‘List_Size’ with values 'M,' ‘L’, ‘XL,’ or ‘XXL.' This is the 'Dependent' attribute. 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.

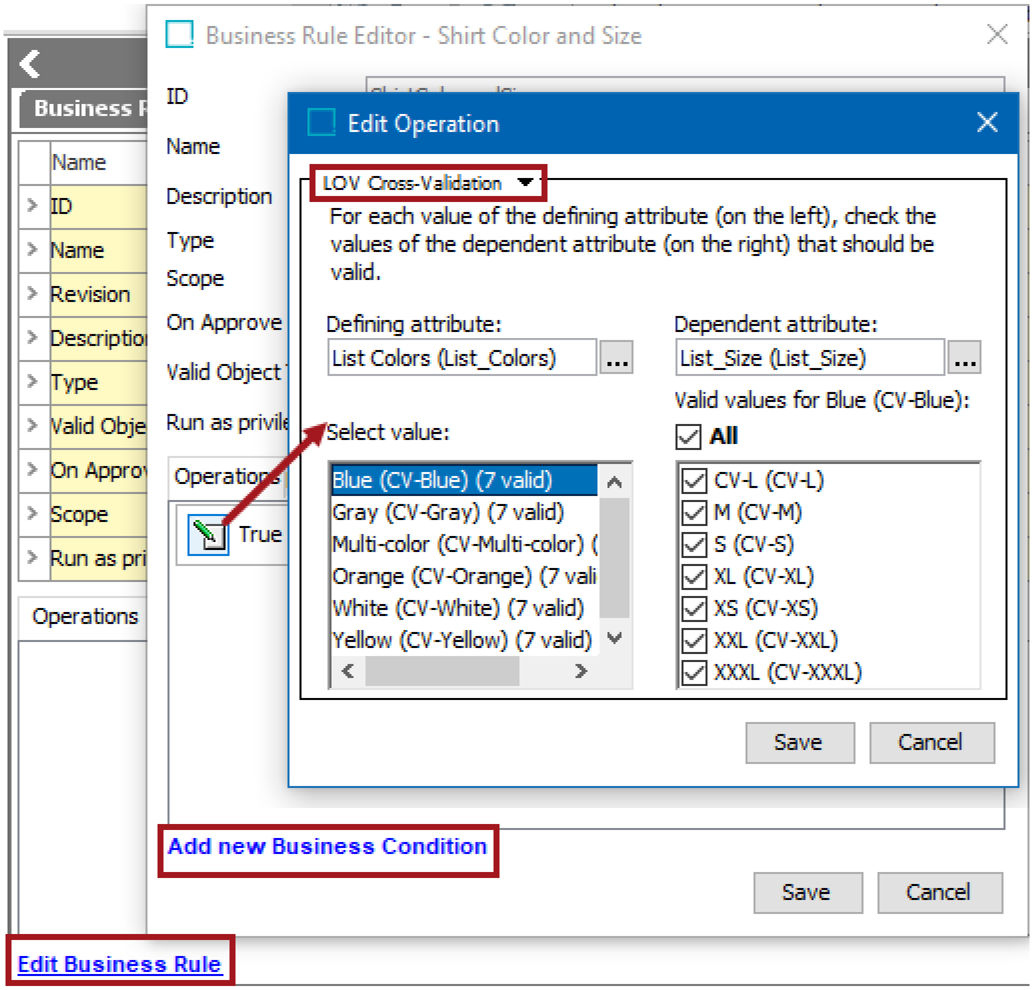
Important: It is not recommended to use LOVs with duplicate values in cross-validation in both dependent and defining attributes.

Prerequisites

Before using this business condition:

1. Verify the 'Use IDs on values' option for both the 'Defining attribute' and the 'Dependent attribute' LOVs is set to **Yes**.
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. Create a business rule as defined in the **Creating a Business Rule, Function, or Library** topic.
4. Edit the business rule to configure the operation as defined in **Editing a Business Rule or Function** topic.

Configuration



Business Rule Editor - Shirt Color and Size

Edit Operation

LOV Cross-Validation

For each value of the defining attribute (on the left), check the values of the dependent attribute (on the right) that should be valid.

Defining attribute: List Colors (List_Colors)

Dependent attribute: List_Size (List_Size)

Select value:

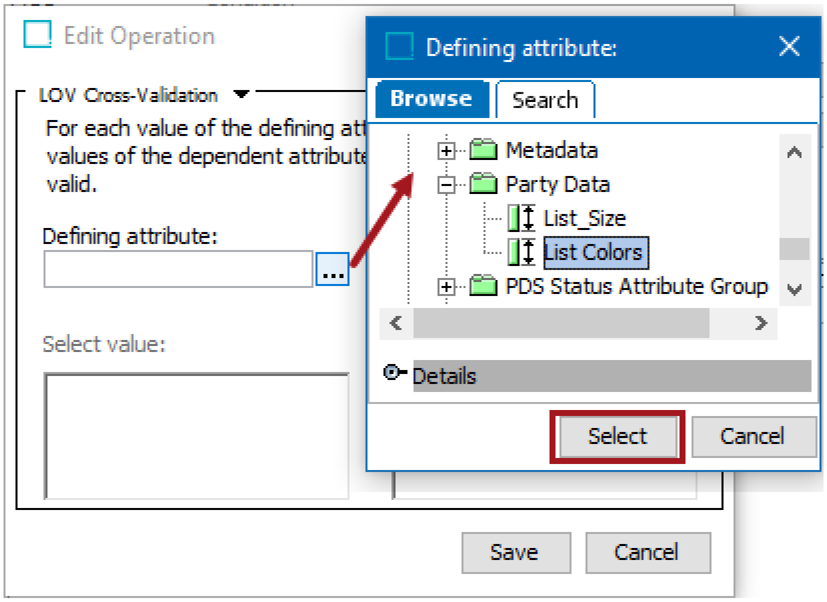
- Blue (CV-Blue) (7 valid)
- Gray (CV-Gray) (7 valid)
- Multi-color (CV-Multi-color) (
- Orange (CV-Orange) (7 vali
- White (CV-White) (7 valid)
- Yellow (CV-Yellow) (7 valid)

Valid values for Blue (CV-Blue):

☑ **All**

- ☑ CV-L (CV-L)
- ☑ M (CV-M)
- ☑ S (CV-S)
- ☑ XL (CV-XL)
- ☑ XS (CV-XS)
- ☑ XXL (CV-XXL)
- ☑ XXXL (CV-XXXL)

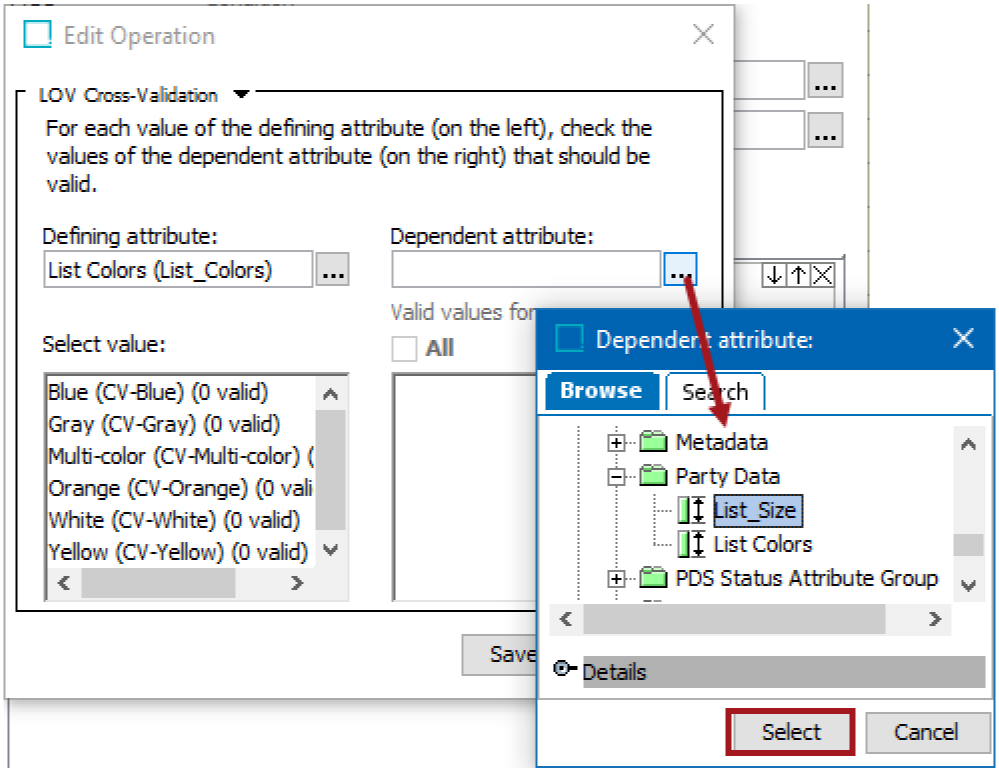
1. On the 'Edit Operation' dialog, select **LOV Cross-Validation** from the dropdown.
2. For the 'Defining attribute' field, click the ellipsis button (**...**) to display the Defining Attributes dialog. Use the 'Browse' or 'Search' tab to identify the defining list of values, then select the desired single-valued LOV attribute. Click **Select** to display this attribute as the defining attribute.



For each value of the defining att
values of the dependent attribute
valid.

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.

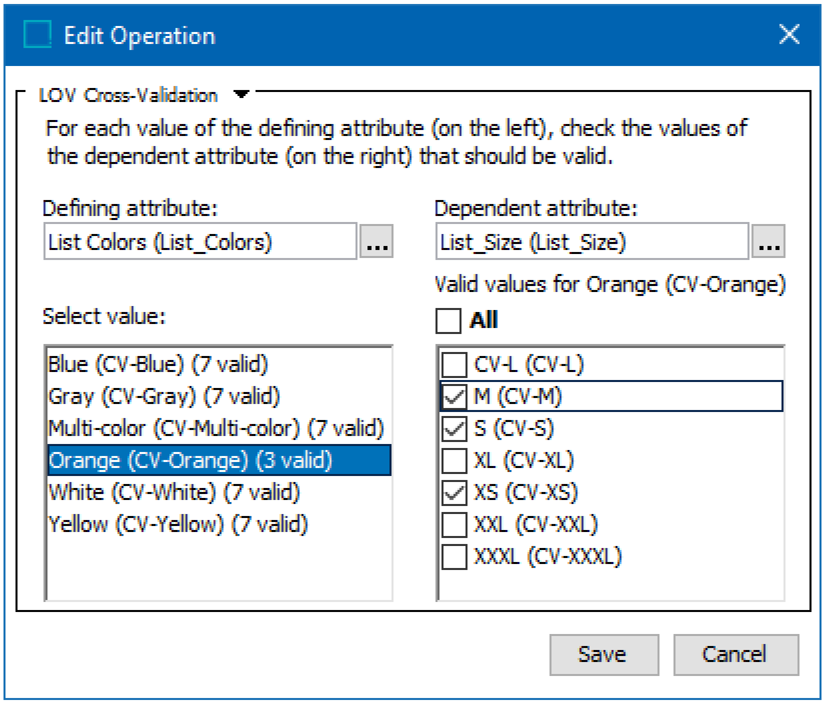


For each value of the defining attribute (on the left), check the values of the dependent attribute (on the right) that should be valid.

Defining attribute: List Colors (List_Colors)

Blue (CV-Blue) (0 valid)
Gray (CV-Gray) (0 valid)
Multi-color (CV-Multi-color) (
Orange (CV-Orange) (0 vali
White (CV-White) (0 valid)
Yellow (CV-Yellow) (0 valid)

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.



Edit Operation

LOV Cross-Validation

For each value of the defining attribute (on the left), check the values of the dependent attribute (on the right) that should be valid.

Defining attribute: List Colors (List_Colors)

Dependent attribute: List_Size (List_Size)

Valid values for Orange (CV-Orange)

□ **All**

Select value:

- Blue (CV-Blue) (7 valid)
- Gray (CV-Gray) (7 valid)
- Multi-color (CV-Multi-color) (7 valid)
- Orange (CV-Orange) (3 valid)
- White (CV-White) (7 valid)
- Yellow (CV-Yellow) (7 valid)

- □ CV-L (CV-L)
- ☑ M (CV-M)
- ☑ S (CV-S)
- □ XL (CV-XL)
- ☑ XS (CV-XS)
- □ XXL (CV-XXL)
- □ XXXL (CV-XXXL)

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.

Note: The cross validation functionality will also work in the Web UI when the LOV Cross-Validation business condition has been configured for two editable LOV attributes inside a data container.

Important: If one or more attributes are configured in multiple business conditions, the business conditions that share attributes will not work when applied. For instance, if two LOV Cross-Validation business conditions are set and one attribute is configured in both, the cross validation functionality will not work as expected.

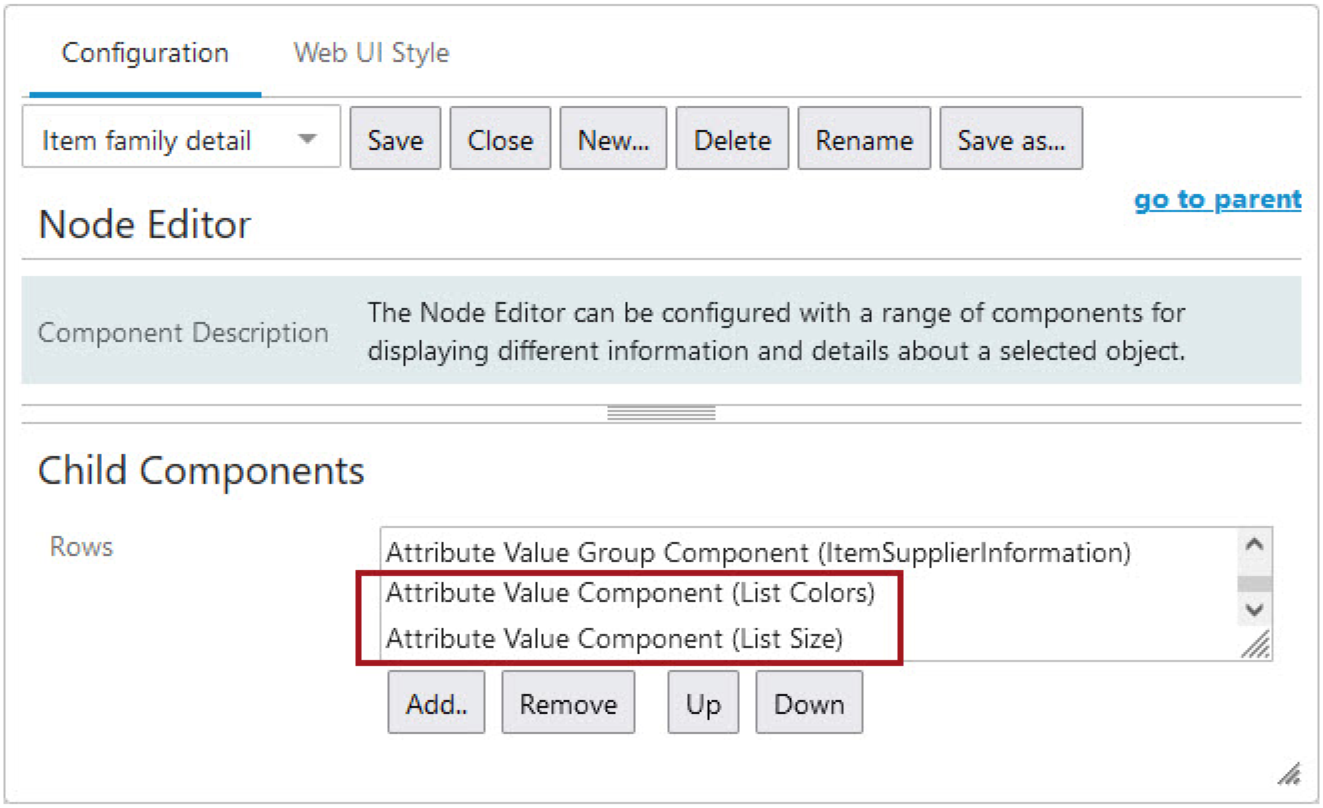
Setup of LOV Cross-Validation in the Web UI

To enable this cross-validation business condition in the Web UI, additional steps must be taken in the Web UI designer.

Navigate to the screen or component the cross-validation will be applied to. In this example, the cross-validation business condition is applied to two 'Attribute Value Components' added to a Node Editor screen.

In the Node Editor, add two Attribute Value Components, one for the 'Defining' attribute, and one for the 'Dependent' attribute.

In this example, an Attribute Value Component has been added for 'List Colors', the 'Defining' attribute, and one for 'List_Size', the 'Dependent' attribute.



Node Editor

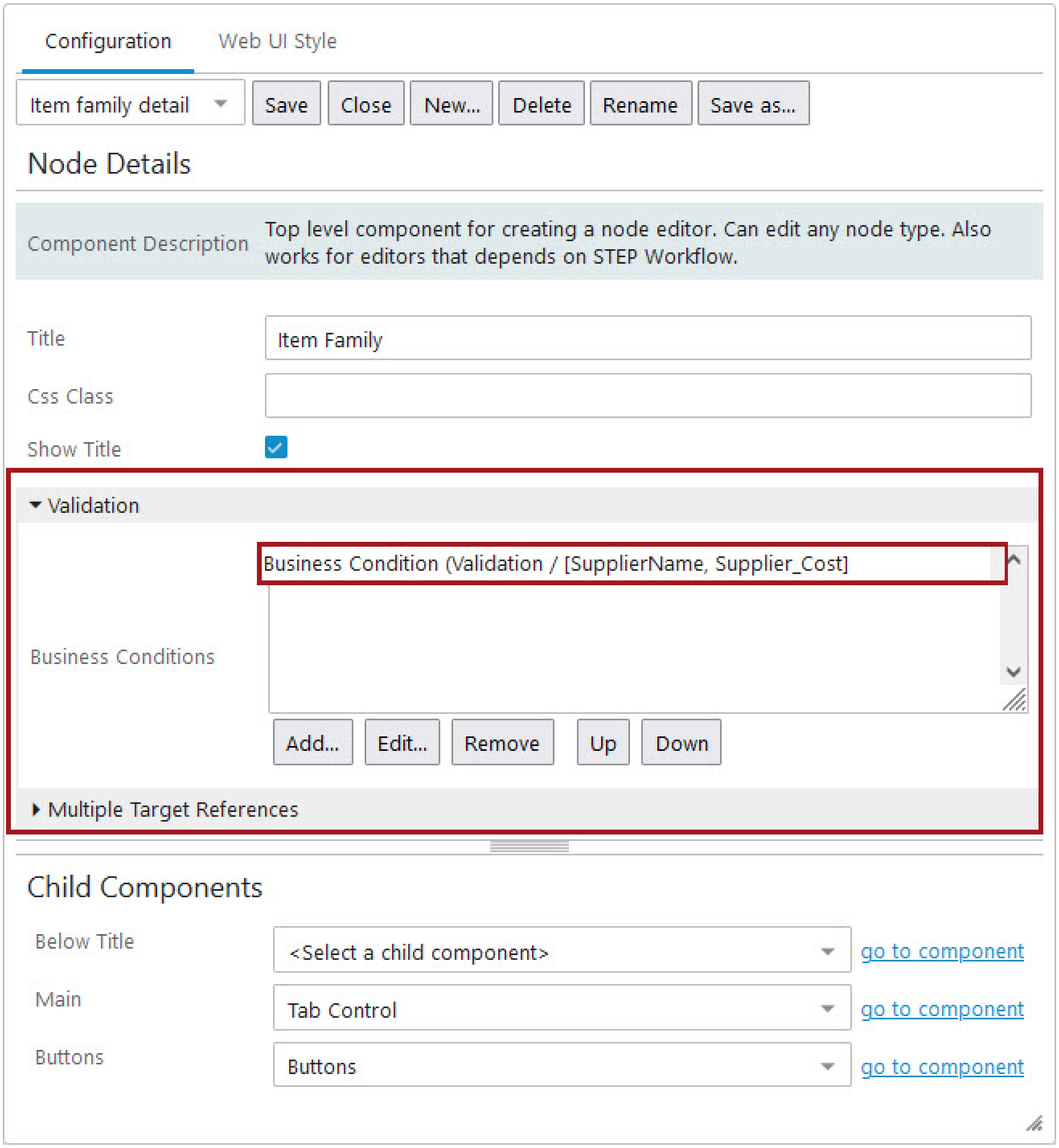
Component Description: The Node Editor can be configured with a range of components for displaying different information and details about a selected object.

Child Components

Rows:
Attribute Value Group Component (ItemSupplierInformation)
Attribute Value Component (List Colors)
Attribute Value Component (List Size)

For more information on setting up an Attribute Value Component, refer to the **Attribute Value Component** topic in the **Web User Interfaces** documentation.

On the Node Details properties screen in the designer (click the 'go to parent' textual button to navigate to this page until 'go to parent' no longer displays), expand the 'Validation' section, and add the business condition that details the cross-validation.



Node Details

Component Description: Top level component for creating a node editor. Can edit any node type. Also works for editors that depends on STEP Workflow.

Title: Item Family

Css Class:

Show Title: ☑

▾ Validation

Business Conditions: Business Condition (Validation / [SupplierName, Supplier_Cost]

Child Components

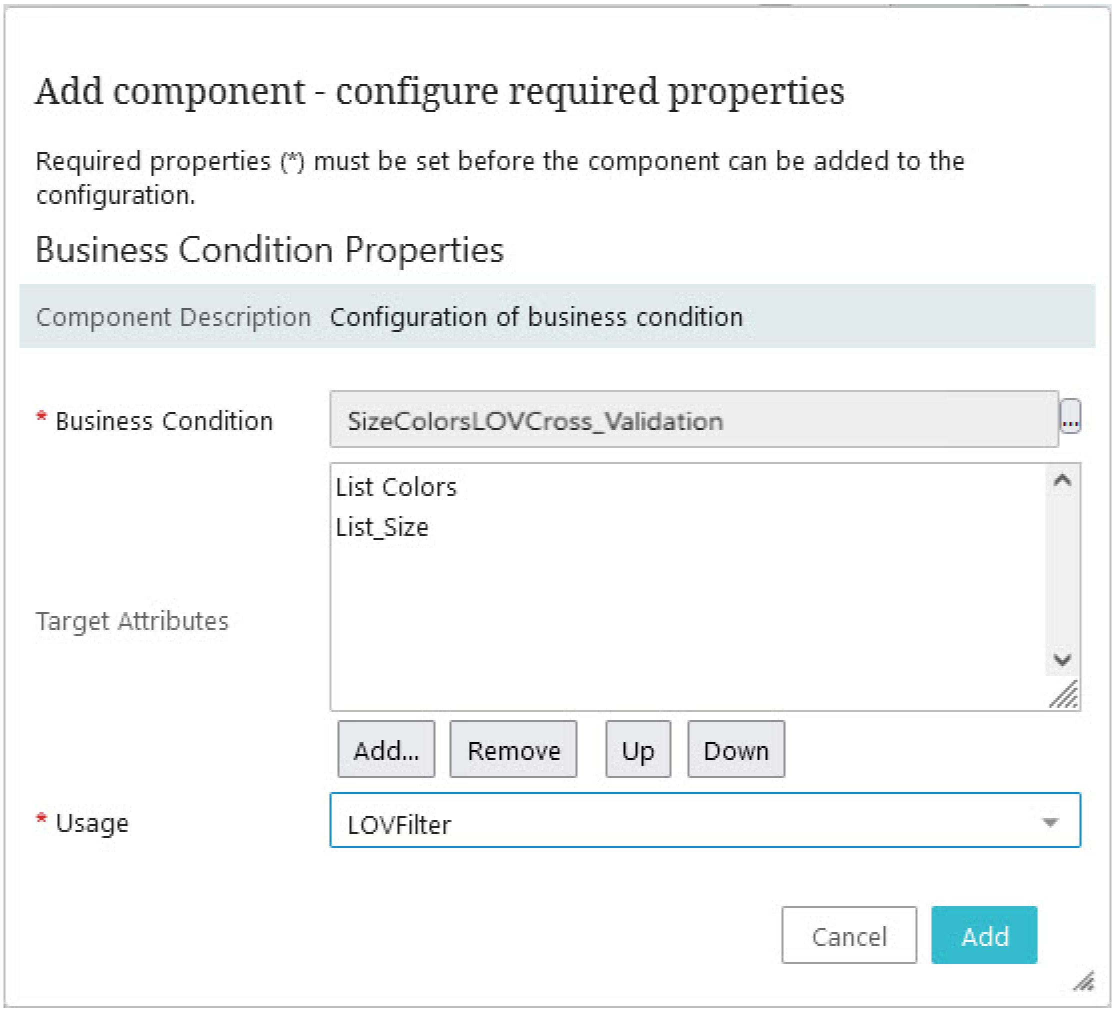
Below Title: <Select a child component> go to component

Main: Tab Control go to component

Buttons: Buttons go to component

Double-click the added business condition. In the 'Add component - configure required properties' dialog that displays, under the 'Target Attributes' parameter, click the 'Add...' button. In the Node Picker that displays, find and select the two attributes—the 'Defining' and 'Dependent' attributes—referenced in the business condition.

For 'Usage', select 'LOVFilter'.



Add component - configure required properties

Required properties (*) must be set before the component can be added to the configuration.

Business Condition Properties

Component Description: Configuration of business condition

* Business Condition: SizeColorsLOVCross_Validation

Target Attributes: List Colors, List_Size

* Usage: LOVFilter

To setup business conditions in the Web UI, refer to the **Business Rules in Web UI** topic in the **Business Rules** documentation.

Bidirectional and Omnidirectional LOV Cross-Validation

If a business condition is configured with both the 'Defining' and 'Dependent' attributes, and those same attributes are added as 'Target Attributes' in the Web UI configuration, the business condition enables a bidirectional LOV cross-validation. This means that the list of LOV options shown in dropdown for the 'Defining' attribute is filtered based on the selected value of the 'Dependent' attribute, and vice versa.

In a business condition where both the 'Defining' and 'Dependent' attributes are configured but only the 'Dependent' attribute is added as a 'Target Attribute' in the Web UI configuration, only the 'Dependent' attribute's LOV options will be filtered and the 'Defining' attribute's LOV option dropdown will show all values. This scenario is an omnidirectional LOV cross-validation. In this case, selecting an LOV value from the 'Defining' attribute's LOV dropdown that is disallowed based on the configuration of the 'Dependent' attribute will cause a warning to display for both the 'Defining' and 'Dependent' attributes.

The LOV cross-validation functionality also works for the following components:

- 'Grouping' (Node Editor)
- 'Initiate Item Screen'
- Globally Configured Data Container Component – Title with Dialog View
- Globally Configured Multi Edit Data Container Component

Business Condition: Reference other Business Condition

This operation allows the current business condition to reference another business condition.

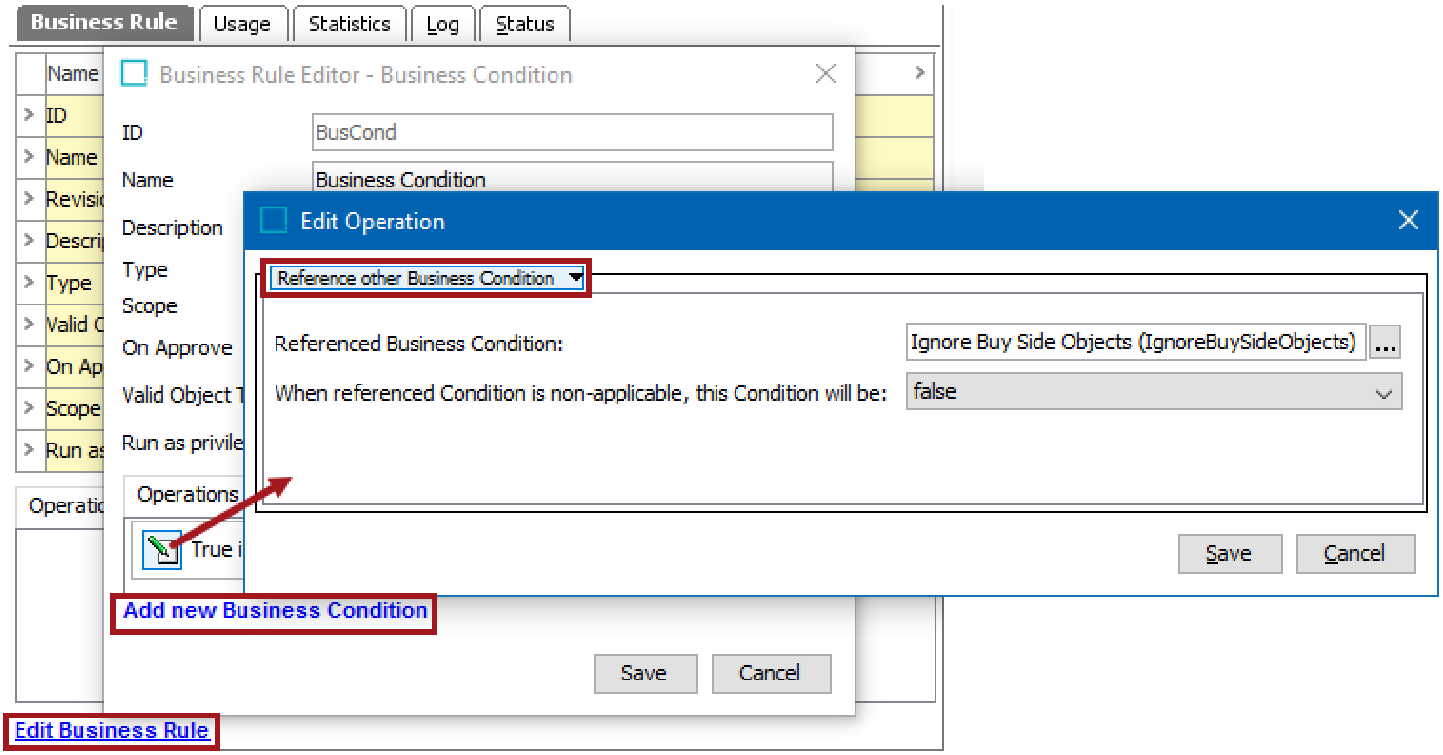
For example, this allows you to use the same business condition in multiple places.

Prerequisites

Before using this business condition:

1. Create a business rule as defined in the **Creating a Business Rule, Function, or Library** topic.
2. Edit the business rule to configure the operation as defined in **Editing a Business Rule or Function** topic.

Configuration



ID: BusCond

Name: Business Condition

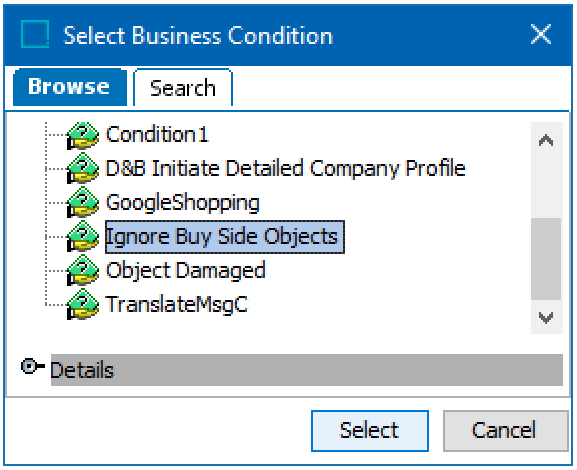
Reference other Business Condition

Referenced Business Condition: Ignore Buy Side Objects (IgnoreBuySideObjects)

When referenced Condition is non-applicable, this Condition will be: false

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.

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.

This condition works on the Product Variant level and returns 'False' if the product variant is a duplicate.

> **Note:** This condition can affect performance based on the number of siblings and variant attributes to be checked.

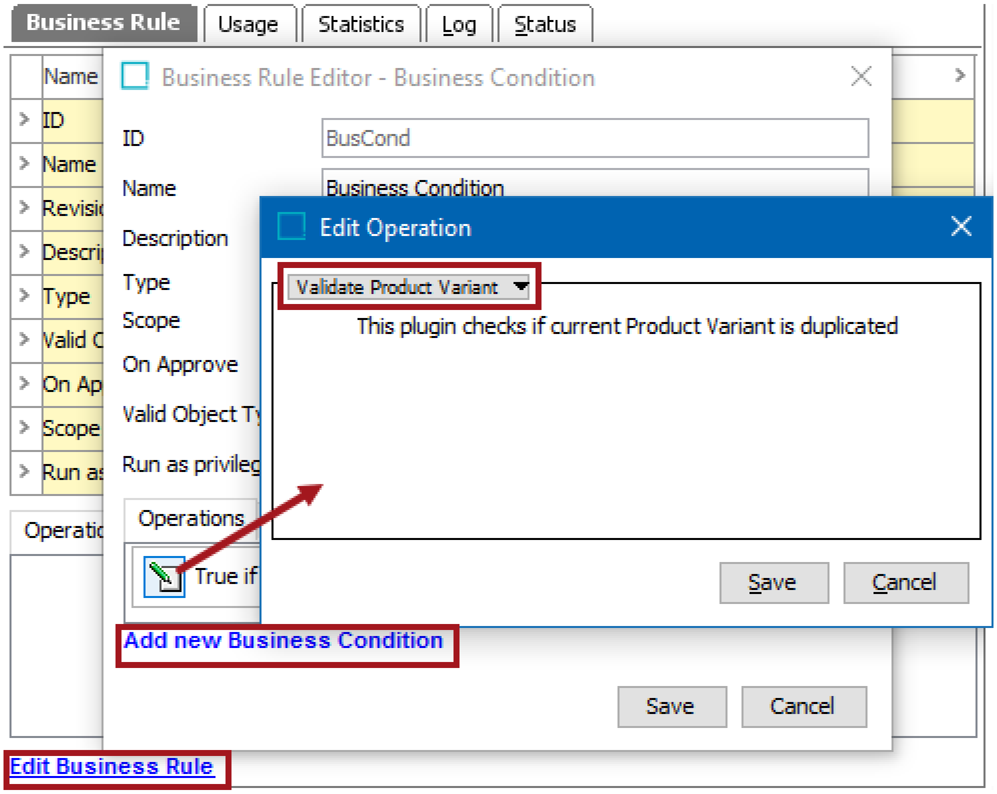
For more information and required configuration, refer to the **Product Variants** topic in the **System Setup** documentation.

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, Function, or Library** topic.
3. Edit the business rule to configure the operation as defined in **Editing a Business Rule or Function** topic.

Configuration



ID: BusCond

Name: Business Condition

Validate Product Variant

This plugin checks if current Product Variant is duplicated

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**.

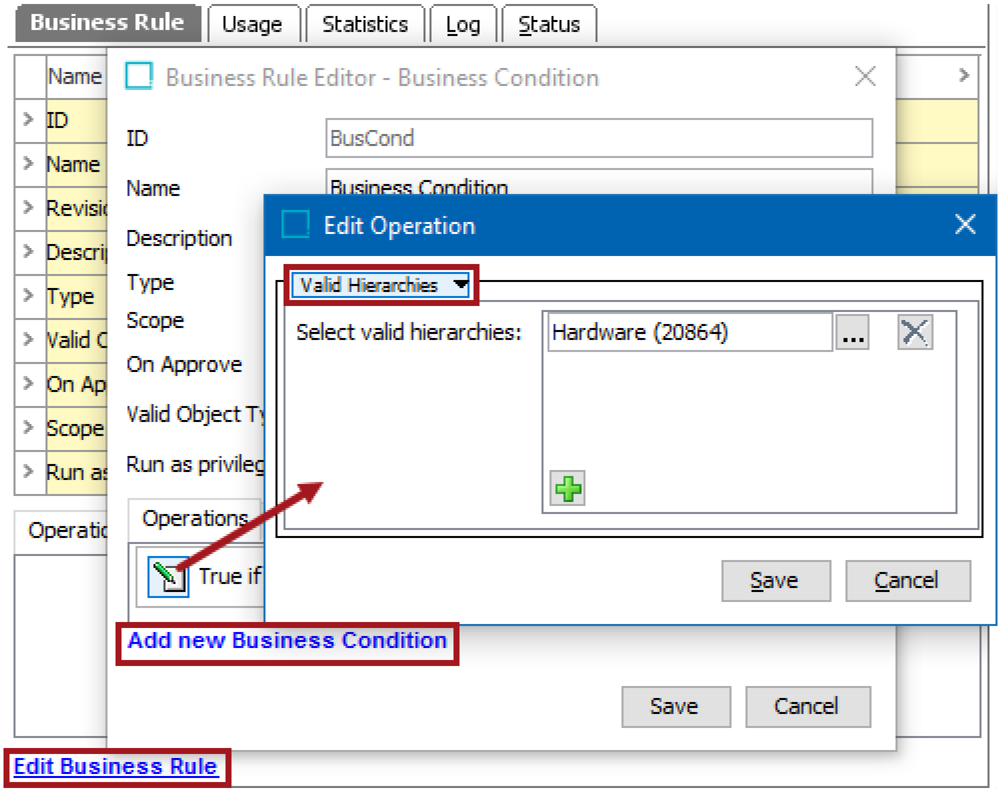
Commonly, this condition is used on the 'Applies If' tab and serves as a precondition for testing the main condition or applying an action.

Prerequisites

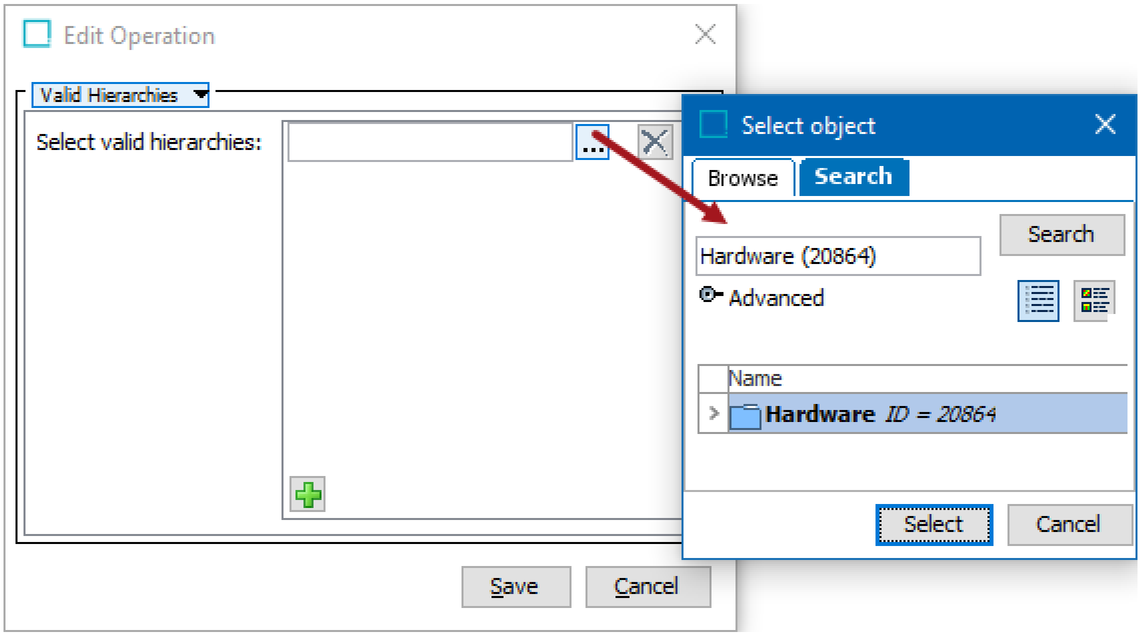
Before using this business condition:

1. Create a business rule as defined in the **Creating a Business Rule, Function, or Library** topic.
2. Edit the business rule to configure the operation as defined in **Editing a Business Rule or Function** 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 the data. Business functions typically serve as helpers, allowing other elements to delegate a part of their logic to reusable business functions. Business functions are valid on all object types.

As with 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, refer to the **Business Conditions** topic or the **Business Actions** topic.

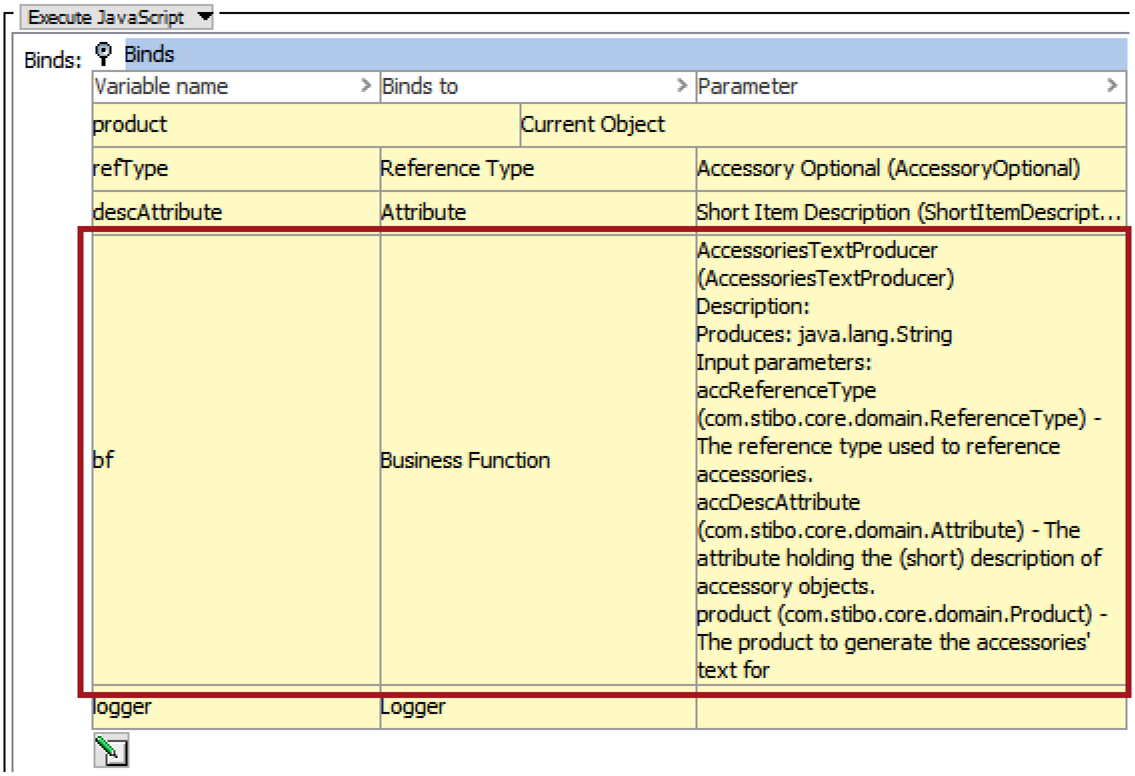
To create a business function:

1. Follow the setup included in the **Creating a Business Rule, Function, or Library** topic.
2. Select a method:
    - Use the Query Template Builder operation to create a Web UI search query, as described in the **Query Template Builder Operation** topic.
    - Use the JavaScript Function operation to write in JavaScript, as described in the **JavaScript Function Operation** topic.
    - Use the Extension API to create a Java business plugin, as described in the **User-Defined Functions** topic.

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, refer to 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.



Variable name	Binds to	Parameter
product	Current Object	
refType	Reference Type	Accessory Optional (AccessoryOptional)
descAttribute	Attribute	Short Item Description (ShortItemDescript...
bf	Business Function	AccessoriesTextProducer (AccessoriesTextProducer) Description: Produces: java.lang.String Input parameters: accReferenceType (com.stibo.core.domain.ReferenceType) - The reference type used to reference accessories. accDescAttribute (com.stibo.core.domain.Attribute) - The attribute holding the (short) description of accessory objects. product (com.stibo.core.domain.Product) - The product to generate the accessories' text for
logger	Logger	

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)

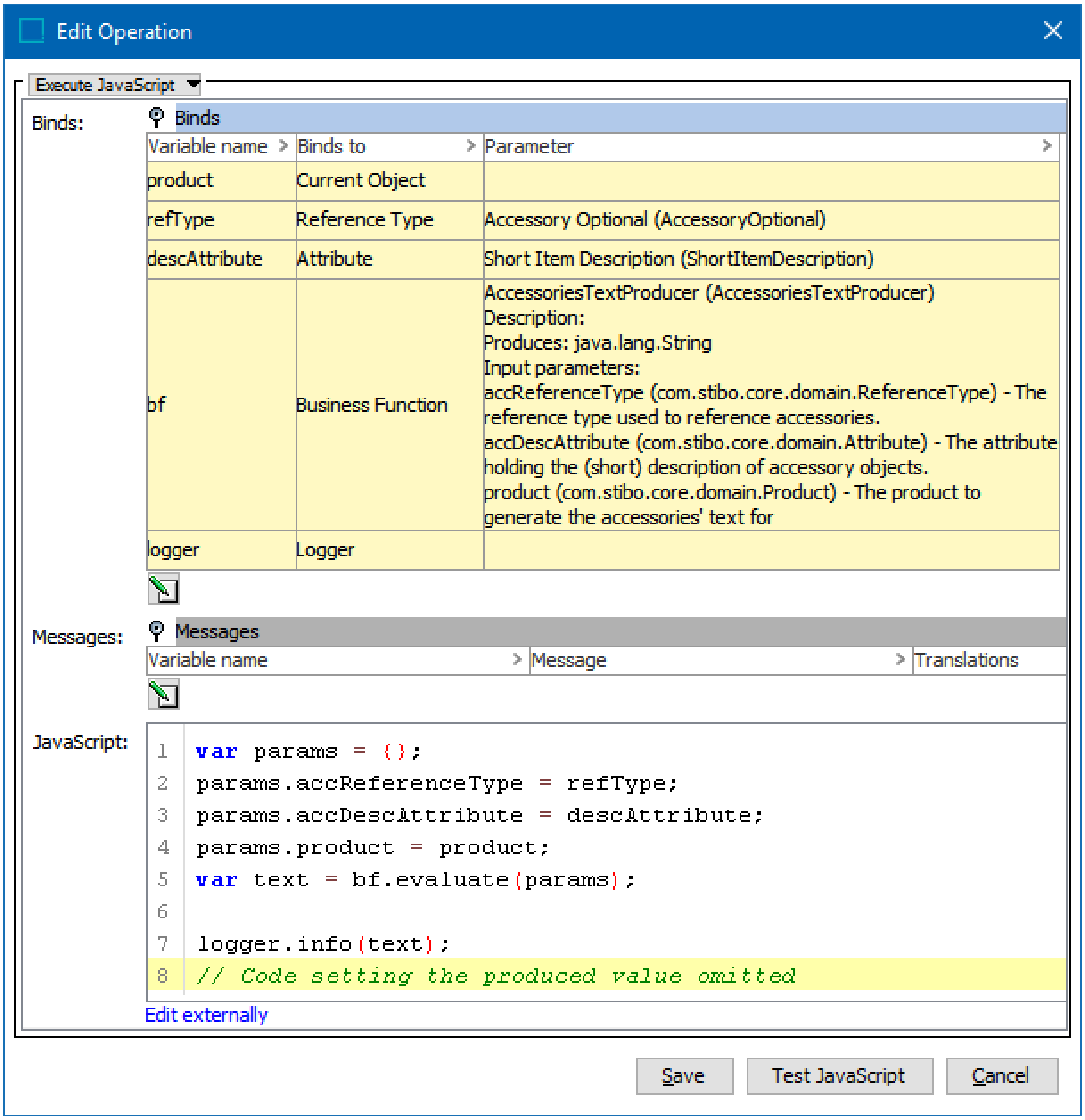
Refer to the scripting API Javadoc in the STEP API Documentation for more information.

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:



Binds:

Variable name	Binds to	Parameter
product	Current Object	
refType	Reference Type	Accessory Optional (AccessoryOptional)
descAttribute	Attribute	Short Item Description (ShortItemDescription)
bf	Business Function	AccessoriesTextProducer (AccessoriesTextProducer) Description: Produces: java.lang.String Input parameters: accReferenceType (com.stibo.core.domain.ReferenceType) - The reference type used to reference accessories. accDescAttribute (com.stibo.core.domain.Attribute) - The attribute holding the (short) description of accessory objects. product (com.stibo.core.domain.Product) - The product to generate the accessories' text for
logger	Logger	

Messages:

Variable name	Message	Translations

JavaScript:

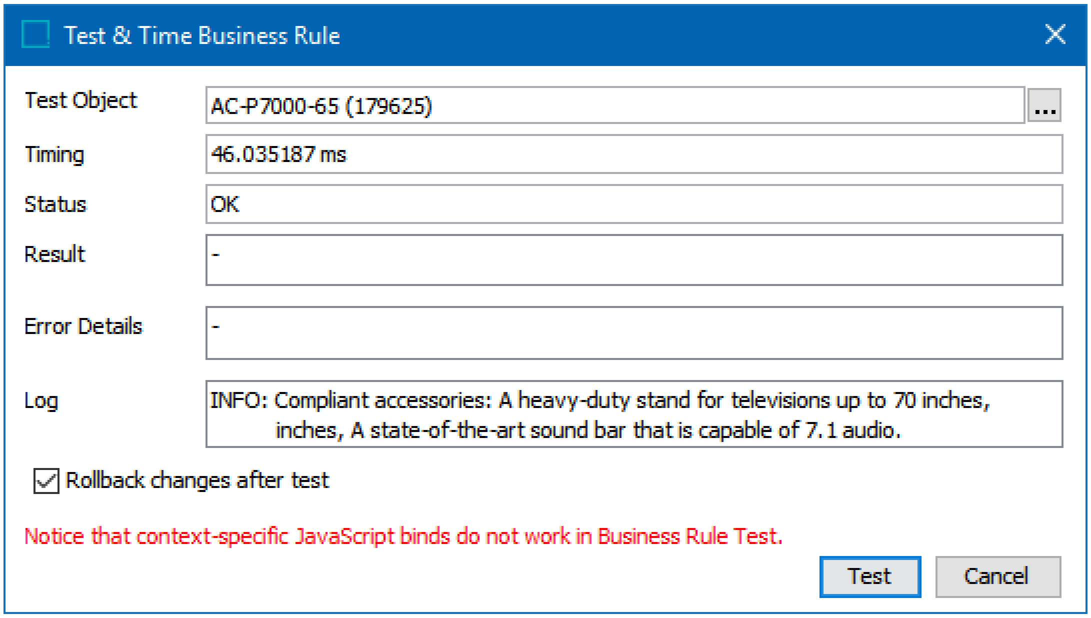
```
var params = {};
params.accReferenceType = refType;
params.accDescAttribute = descAttribute;
params.product = product;
var text = bf.evaluate(params);

logger.info(text);
// Code setting the produced value omitted
```

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
```

Testing this business action shows the proper results:



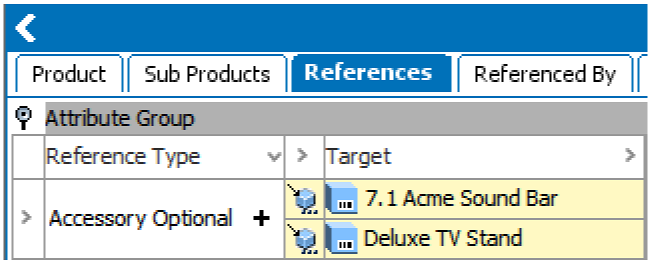
Test & Time Business Rule

Test Object	AC-P7000-65 (179625)
Timing	46.035187 ms
Status	OK
Result	-
Error Details	-
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.

☑ Rollback changes after test

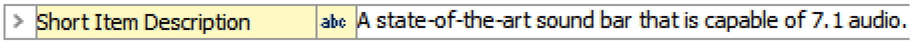
Notice that context-specific JavaScript binds do not work in Business Rule Test.

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:



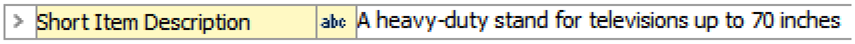
Reference Type	Target
Accessory Optional	7.1 Acme Sound Bar Deluxe TV Stand

For the 7.1 Acme Sound Bar product, the short item description is:



Short Item Description	A state-of-the-art sound bar that is capable of 7.1 audio.

For the Deluxe TV Stand, the short item description is:

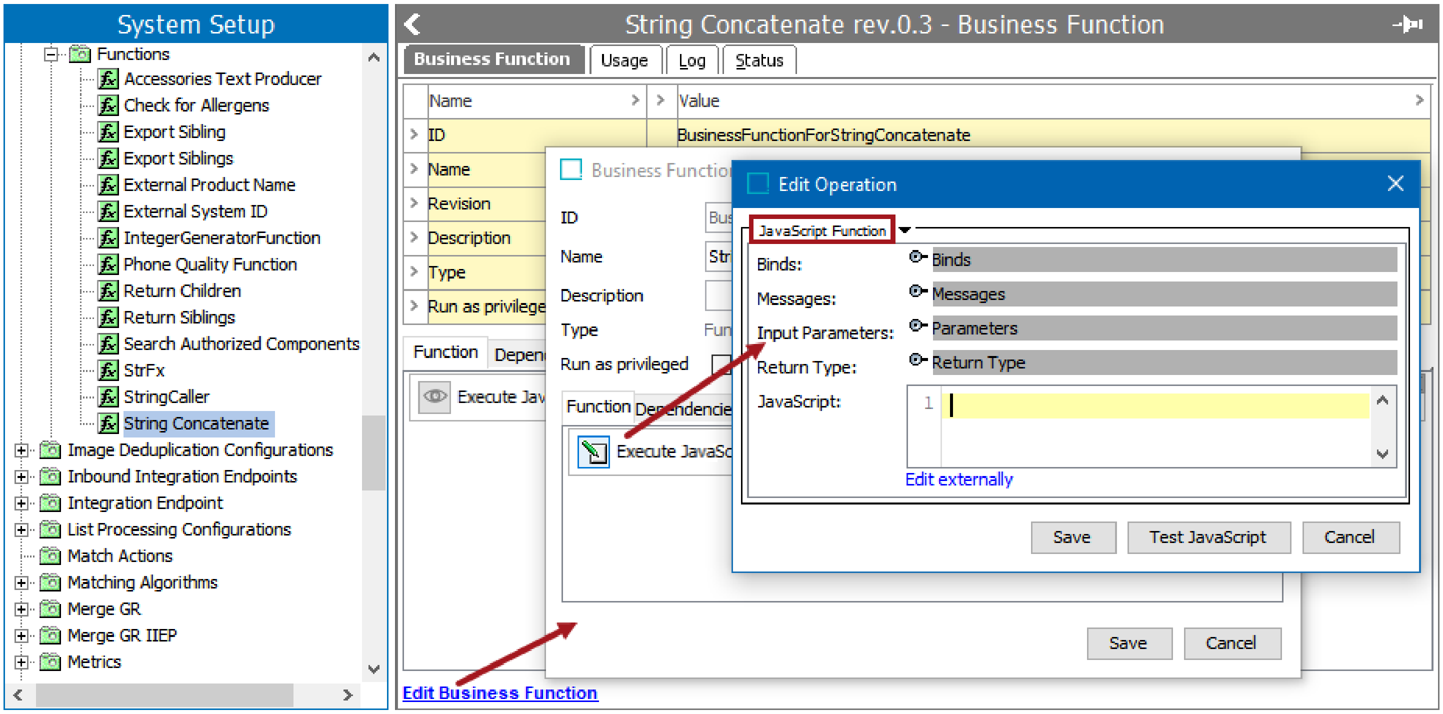


Short Item Description | abc | A heavy-duty stand for televisions up to 70 inches

In the test, both of these attributes are returned per the JavaScript function as shown in the Log parameter on the Test & Time Business Rule image above.

JavaScript Function Operation

The default Business Function operation is the 'JavaScript Function' and displays on the Function tab. This operation allows users to define their own functions using JavaScript while binding various data for use.



System Setup

- Functions
  - Accessories Text Producer
  - Check for Allergens
  - Export Sibling
  - Export Siblings
  - External Product Name
  - External System ID
  - IntegerGeneratorFunction
  - Phone Quality Function
  - Return Children
  - Return Siblings
  - Search Authorized Components
  - StrFx
  - StringCaller
  - String Concatenate
- Image Deduplication Configurations
- Inbound Integration Endpoints
- Integration Endpoint
- List Processing Configurations
- Match Actions
- Matching Algorithms
- Merge GR
- Merge GR IIEP
- Metrics

String Concatenate rev.0.3 - Business Function

Name	Value
ID	BusinessFunctionForStringConcatenate
Name	
Revision	
Description	
Type	
Run as privilege	

Edit Operation

JavaScript Function

Binds: Binds

Messages: Messages

Input Parameters: Parameters

Return Type: Return Type

JavaScript: 1

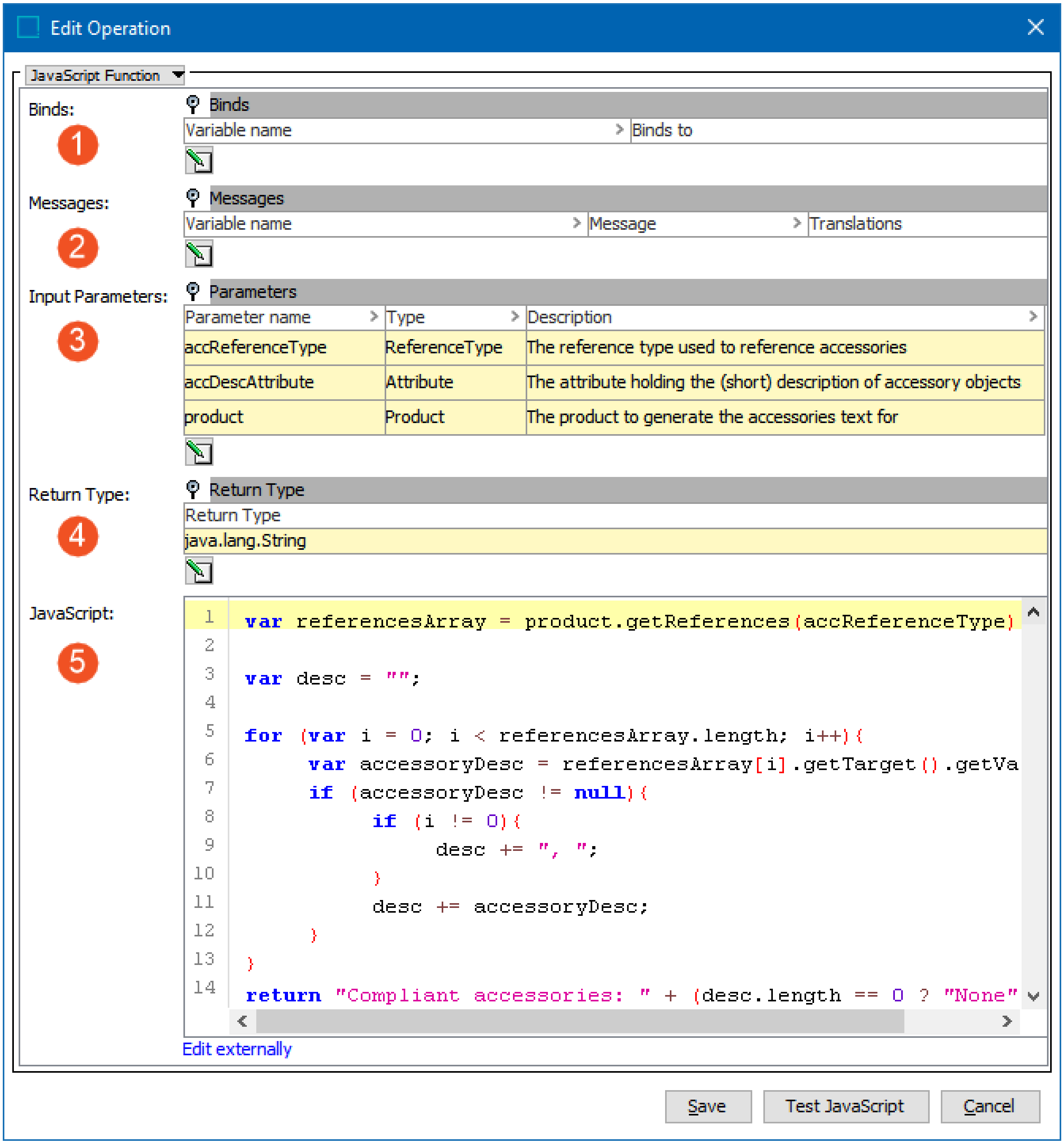
To add the JavaScript to this operation, click the **Edit Business Function** link.

For more information on editing a business function, refer to the **Editing a Business Rule or Function** topic in the **Business Rules** documentation.

Refer to the **Custom Reference Target Search** topic in the **Web User Interfaces** documentation for more information on creating reference editors that target objects the user is allowed to select based on the object type validity configured on the reference type.

Configuration

The example in this section configures a JavaScript Function to display the text of a referenced object's attribute value description.



Binds

Variable name	Binds to

Messages

Variable name	Message	Translations

Parameters

Parameter name	Type	Description
accReferenceType	ReferenceType	The reference type used to reference accessories
accDescAttribute	Attribute	The attribute holding the (short) description of accessory objects
product	Product	The product to generate the accessories text for

Return Type

Return Type
java.lang.String

```
var referencesArray = product.getReferences(accReferenceType)

var desc = "";

for (var i = 0; i < referencesArray.length; i++){
     var accessoryDesc = referencesArray[i].getTarget().getVa
     if (accessoryDesc != null){
          if (i != 0){
               desc += ", ";
          }
          desc += accessoryDesc;
     }
}
return "Compliant accessories: " + (desc.length == 0 ? "None"
```

1. **Binds -** bind STEP objects to the JavaScript, including the 'STEP Manager' and 'Logger' dynamic binds available for business functions. For more information on JavaScript Binds, refer to the **JavaScript Binds** topic as well as the **Adding a Bind** topic in the **Resource Materials** online help.

2. **Messages -** 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, refer to the **Localized Messages for JavaScript Business Rules** topic in the **Execute JavaScript** section.

3. **Input Parameter -** define parameters that can be passed to the function to produce an output. A number of STEP and Java types are available.

   For a custom reference target search, use one of these:

   - **String**: Populated with the search string the user has entered. Most cases require a String input.
   - **Node**: Populated with any node selected. When creating a reference, this will be the source node of the reference. Most cases require a Node input.
   - **Data Container Object**: If the user is within a data container editor, the source of the reference will be a data container instance, and this source will be available to the business function as a Data Container Object.

   > **Note:** Use caution when changing an input parameter type for an active JavaScript Function business rule. Business rules that call the modified function may also need updating.

   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. **Return Type -** define the type of data returned by the business function.

   > **Note:** Use caution when changing a return type for an active JavaScript Function business rule. Business rules that call the modified function may also need updating.
   > Also, it is important to note that JavaScript Business Functions configured to return either a 'Double' or an 'Integer' or lists of these types should be specified in the return command rather than relying on the JavaScript to Java type conversion. For example, to return an integer value of 3 use:
   >
   > ```
   > return new java.lang.Integer(3);
   > ```
   >
   > instead of
   >
   > ```
   > return 3;
   > ```
   >
   > In the latter case, return values that are in fact integers may not be recognized as such and will produce errors.

   For a custom reference target search, the business function must return a 'Query Specification.' Otherwise, the business function cannot be selected in the Web UI for Target Search Function in the Search Table Tab Properties (for the References component Node Picker Dialog) as described later in this topic.

5. **JavaScript -** add function logic. 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.

For a custom reference target search, the business function must return a 'Query Specification.' Otherwise, the business function cannot be selected in the Web UI for Target Search Function in the Search Table Tab Properties (for the References component Node Picker Dialog) as described in the **Custom Reference Target Search Configuration** topic of the **Web User Interfaces** documentation.

Refer to the **Calling a Business Function from a JavaScript Business Rule** topic for information on executing and testing business functions.

Query Template Builder Operation

The 'Query Template Builder' is a way to create a business function that returns a query result. Users can create a search query business function without knowing JavaScript.

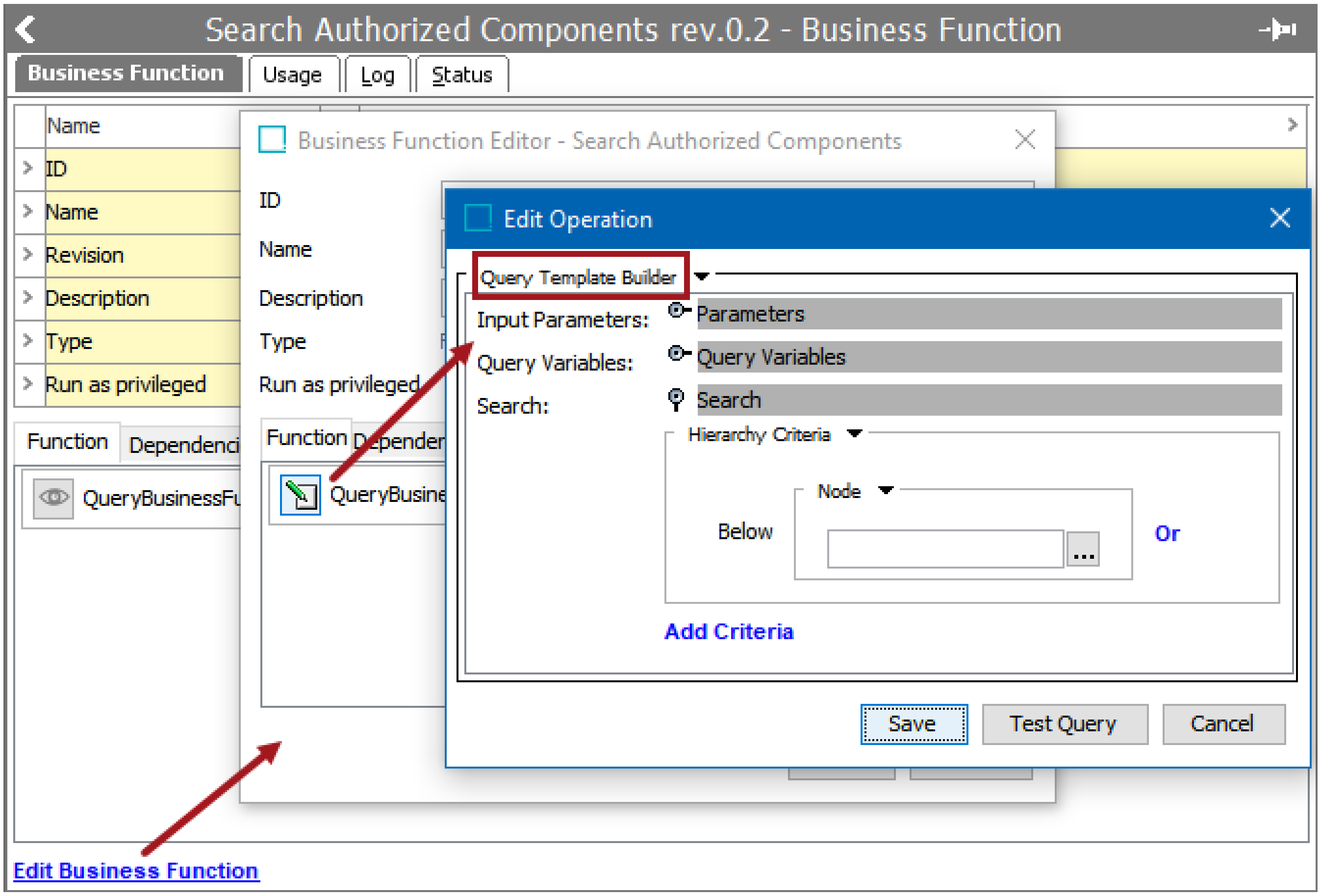
For example, in Web UI, a reference target search benefits from a query by restricting the reference type options available to only those that are appropriate for the use case. The Query Template Builder supports combining criteria to declare precisely which targets should be available for a given reference. For configuration steps, refer to the **Custom Reference Target Search** topic in the **Web User Interfaces** documentation.

Sample uses for the Query Template Builder include limiting reference type search results to targets, such as:

- below a specified hierarchy node to limit reference targets to only products under a specified classification.
- of a certain object type, like only contact person.
- with a reference of a certain type, and/or points to the same node as the reference on the current object, like limiting the supplier location selection to only locations belonging to an already selected supplier.
- with an attribute value that matches the attribute value on the current object or data container, like selecting only remote controls that supports the protocol of the current TV object.
- where the entity owned by the data container already has at least one of the specific type of data container being referenced.

Refer to the **Use Cases** section at the end of this topic for other common scenarios and setup.

To add the query to this operation, click the **Edit Business Function** link.

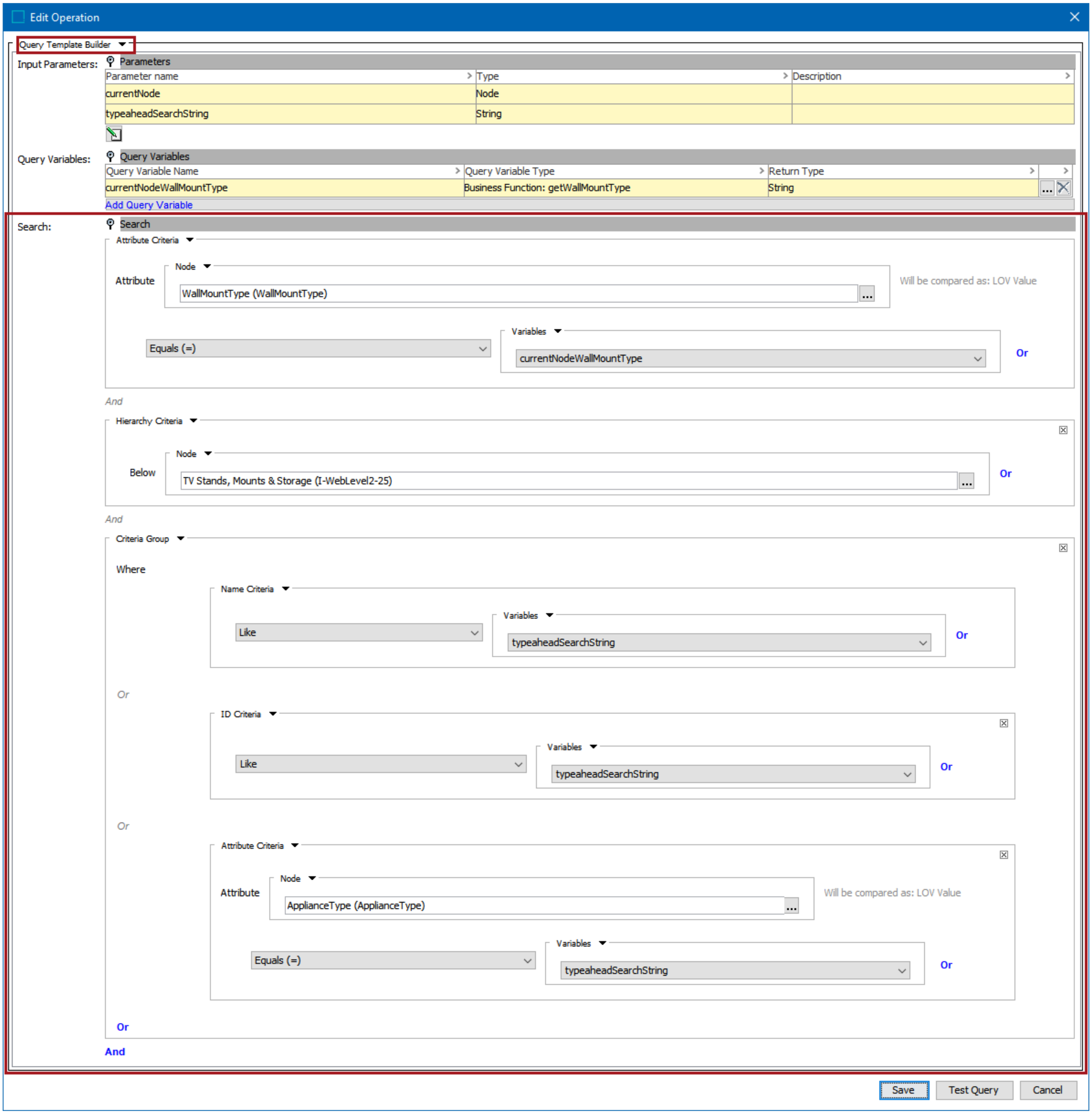


For more information on editing a business function, refer to the **Editing a Business Rule or Function** topic in the **Business Rules** documentation.

Configuration Example

The example in this section configures a Business Function so the reference target options are limited to:

- products where the attribute (WallMountType) matches the setting on the current node (via the 'getWallMountType' business function provided by the Query Variable)
- AND that are below the identified classification hierarchy (I-WebLevel2-25)
- AND where the specified search string text matches the name, ID, or the value found in the specified attribute.



Input Parameters: Parameters

Parameter name	Type	Description
currentNode	Node	
typeaheadSearchString	String	

Query Variables: Query Variables

Query Variable Name	Query Variable Type	Return Type
currentNodeWallMountType	Business Function: getWallMountType	String

Search:

Attribute Criteria
Attribute — Node: WallMountType (WallMountType) — Will be compared as: LOV Value
Equals (=) — Variables: currentNodeWallMountType — Or

And

Hierarchy Criteria
Below — Node: TV Stands, Mounts & Storage (I-WebLevel2-25) — Or

And

Criteria Group
Where
Name Criteria
Like — Variables: typeaheadSearchString — Or

Or

ID Criteria
Like — Variables: typeaheadSearchString — Or

Or

Attribute Criteria
Attribute — Node: ApplianceType (ApplianceType) — Will be compared as: LOV Value
Equals (=) — Variables: typeaheadSearchString — Or

Or
And

Note: When using variables, if the variable cannot be resolved, the Query Template Builder search ignores the criteria to build the search. Continuing with the example above, if after creating the search, you delete the getWallMountType business function, then the top-most Attribute value criteria cannot be evaluated and as a result, the Query Template Builder returns any product linked under the 'TV Stands, Mounts & Storage'

> hierarchy, where either product name, product ID or the ApplianceType attribute matches the search string the user entered in the typeahead.

1. **Input Parameter** - define the parameters that can be passed to the function from the outside. STEP and Java types are available. For the business function to be used in Web UI for Custom Reference Target Search, in input parameters must be of type Node and String.

   > **Note:** Use caution when changing an input parameter type for an active business rule. Business rules that call the modified function may also need to be updated.

2. **Query Variables** - define business function(s) to be used within the search query. Click the **Add Query Variable** link, click the **Business Function** option, and add a Variable Name. Click the ellipsis button (...) to choose the desired business function and make it available for selection in the Search flipper Variables dropdowns. Query Variables allow the user to retrieve values to be used in the search criteria from the current object. For example, to search wall mounts that match the current node can be a business function that get a current node object type or the parent of the current node.

3. **Search** - use the available search operations to build a query. Only objects that are true for the entire criteria in the search are included in the result set. At least one criteria must exist and the Hierarchy Criteria is displayed by default. Click the dropdown to change the first criteria if necessary since it cannot be deleted.

   - **Hierarchy Criteria** - defines the location in the hierarchy to search below. Select either Variables or Node from the dropdown and select the Query Variable or click the ellipsis button (...) to select the node.

   - **Object Type Criteria** - defines the object type to search for. Select either Variables or Node from the dropdown and select the Query Variable or click the ellipsis button (...) to select the node.

   - **ID Criteria** - uses an Equals, Does Not Equal, or Like (preferred) operator as a search criteria. Select either Variables or Node from the dropdown and select the Query Variable or click the ellipsis button (...) to select the node. Using the 'Like' operator allows searching on parts of the ID. 'Like' is necessary in custom reference target search to match the user's partially written search string towards the STEP ID.

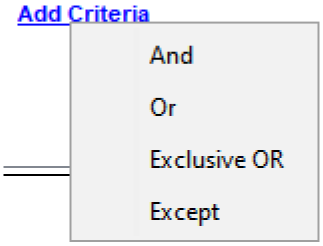
   - **Name Criteria** - uses an Equals, Does Not Equal, or Like (preferred) operator to define the name of the target. Select either Variables or Node from the dropdown and select the Query Variable or click the ellipsis button (...) to select the node. Using the 'Like' operator allows searching on parts of the name. 'Like' is necessary in custom reference target search to match the user's partially written search string towards the STEP name.

   - **Attribute Criteria** - search a specific attribute. Both the value and the attribute can be defined by variables or input parameters. Select either Variables or Node from the dropdown and select the Query Variable or click the ellipsis button (...) to select the node.

   - **Reference Type Criteria** - search nodes containing a specific reference type. Defines the reference type allowed for the target. Select either Variables or Node from the dropdown and select the Query Variable or click the ellipsis button (...) to select the node.

- **Criteria Group** - defines how multiple criteria combine. One criteria group combines the included criteria using a selectable operator. As defined in the next section, some criteria options have limits: AND and OR allow more criteria; while EXCLUSIVE OR and EXCEPT allow only two criteria.

Click the **Add Criteria** link to display the operator types and choose one when adding a new criteria.



- **And** - All criteria must be true for the whole Criteria Group is considered true.
- **Or** - If at least one criteria is true, the whole Criteria Group is considered true.
- **Exclusive OR** - combines only two criteria; when one is true and the other is false, the whole Criteria Group is considered true. For example, to query for packages that are either large and light or small and heavy, create the criteria as: <package_size larger than XL> EXCLUSIVE OR <package_weight larger than Heavy>.
- **Except** - combines only two criteria; when the first is true and the second is false, the whole Criteria Group is considered true.

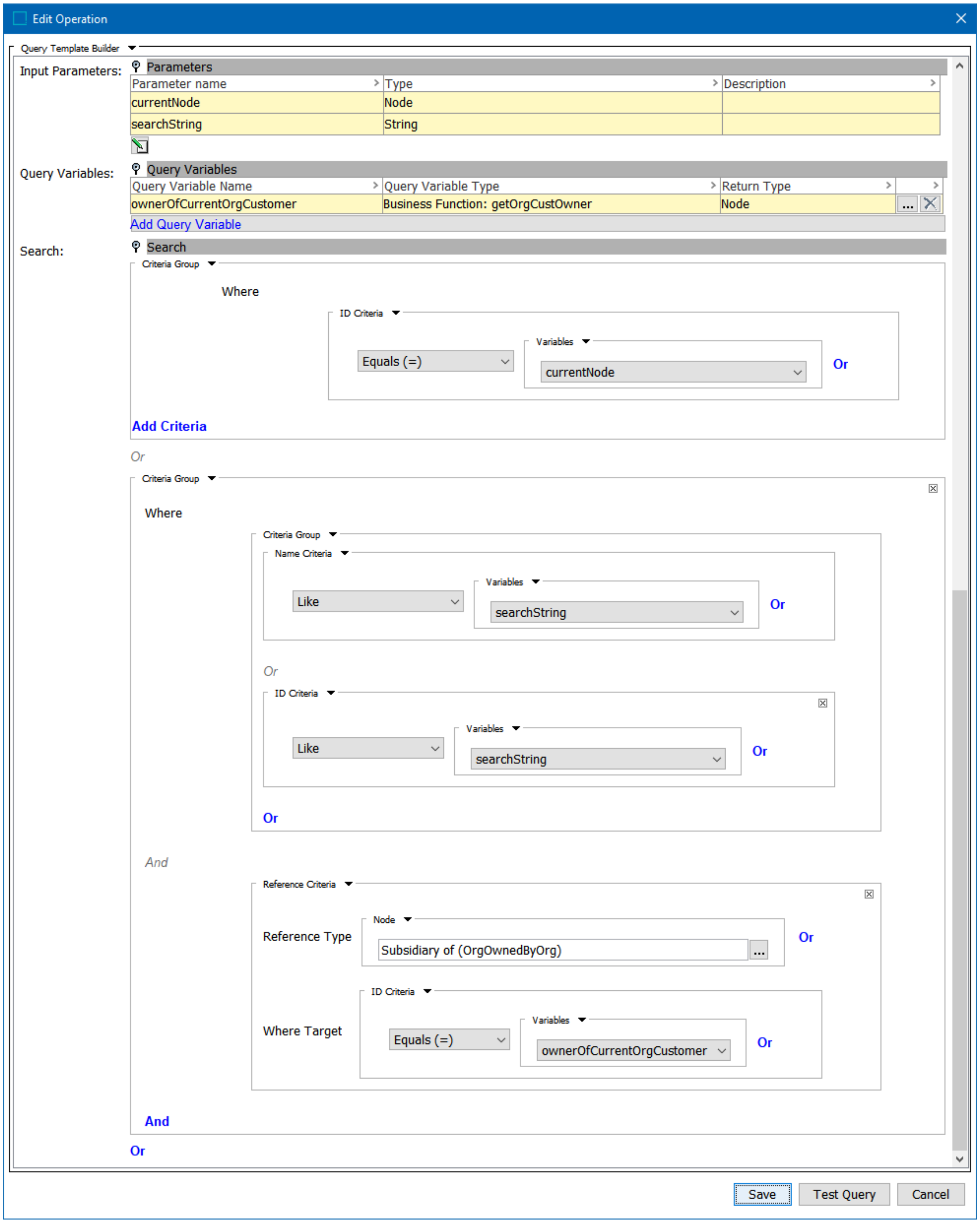
To remove criteria, on a single criteria, click the delete button (☒); on a multi-criteria item, click the delete button (X) to remove only a part of the criteria.

Click the **Or** link (Or) or the **And** link (And) to create a relationship between criteria.

Use Cases

The image above shows a use case mapping applicable wall mounts for televisions where the appliance type of the target of the reference must match the wall mount type on the current object.

A use case in customer data is when selecting a 'ship to' location for a sales area. The user is allowed to select a 'ship to' location and the options will include the customer itself (to indicate direct shipments). Additionally, 'ship to' locations must belong to the 'ship to' account group and match the search characters the user entered. In the image below, the 'ship to' customer itself continues to be available regardless of the search characters entered by the user.



Edit Operation

Query Template Builder

Input Parameters:

Parameters

Parameter name	Type	Description
currentNode	Node	
searchString	String	

Query Variables:

Query Variables

Query Variable Name	Query Variable Type	Return Type
ownerOfCurrentOrgCustomer	Business Function: getOrgCustOwner	Node

Add Query Variable

Search:

Search

Criteria Group

Where

ID Criteria: Equals (=) — Variables: currentNode — Or

Add Criteria

Or

Criteria Group

Where

Criteria Group

Name Criteria: Like — Variables: searchString — Or

Or

ID Criteria: Like — Variables: searchString — Or

Or

And

Reference Criteria

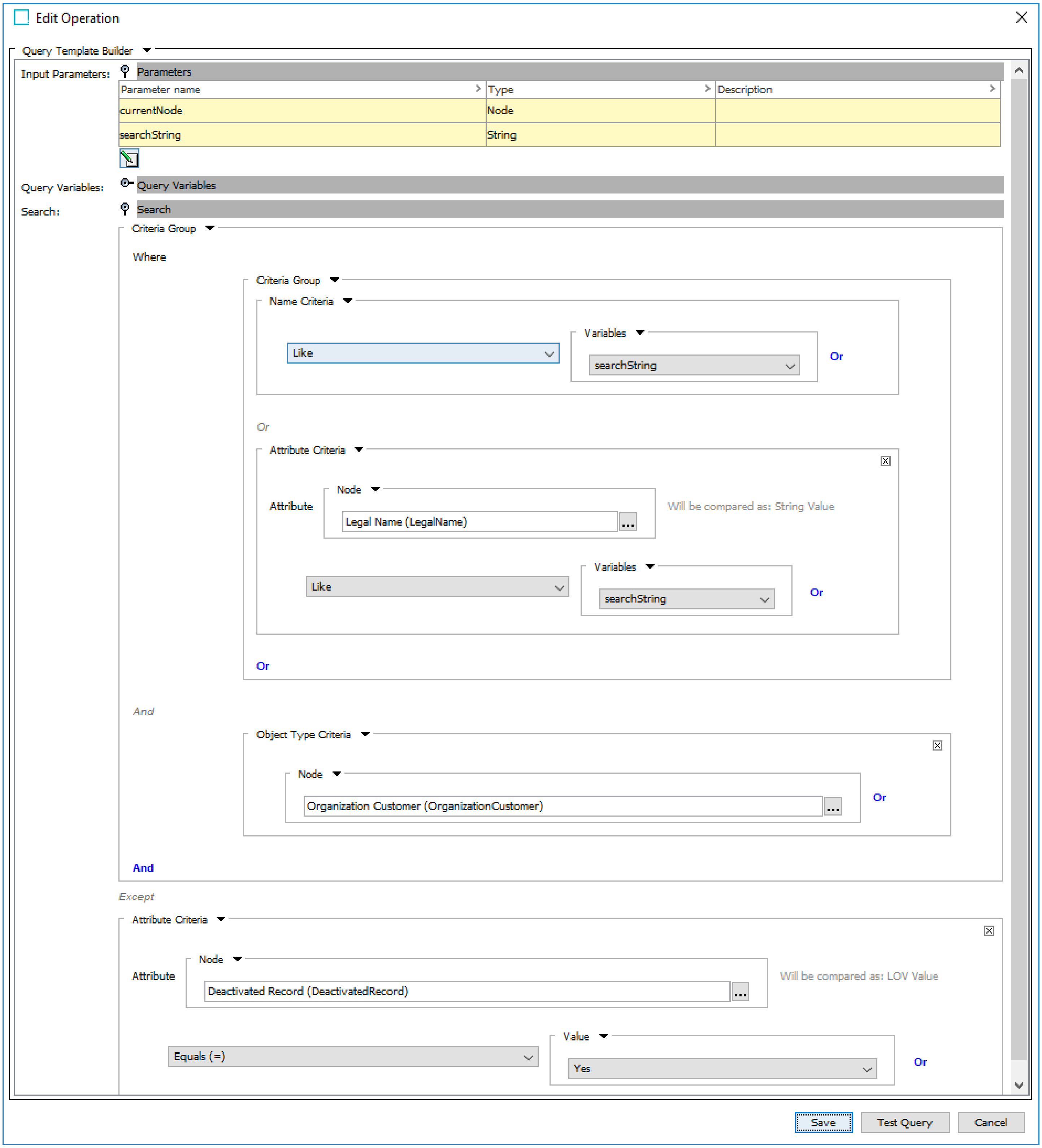
Reference Type — Node: Subsidiary of (OrgOwnedByOrg) — Or

Where Target — ID Criteria: Equals (=) — Variables: ownerOfCurrentOrgCustomer — Or

And

Or

While yet another customer data use case is onboarding a contact person, where a contact-to-organization reference is mandatory. The reference target search finds organizations based on the entered string compared to Legal Name or STEP name and also filters out deactivated organization records.



Edit Operation

Query Template Builder

Input Parameters: Parameters

Parameter name	Type	Description
currentNode	Node	
searchString	String	

Query Variables: Query Variables

Search: Search

Criteria Group

Where

Criteria Group

Name Criteria

Like

Variables: searchString

Or

Or

Attribute Criteria

Attribute — Node: Legal Name (LegalName)

Will be compared as: String Value

Like

Variables: searchString

Or

Or

And

Object Type Criteria

Node: Organization Customer (OrganizationCustomer)

Or

And

Except

Attribute Criteria

Attribute — Node: Deactivated Record (DeactivatedRecord)

Will be compared as: LOV Value

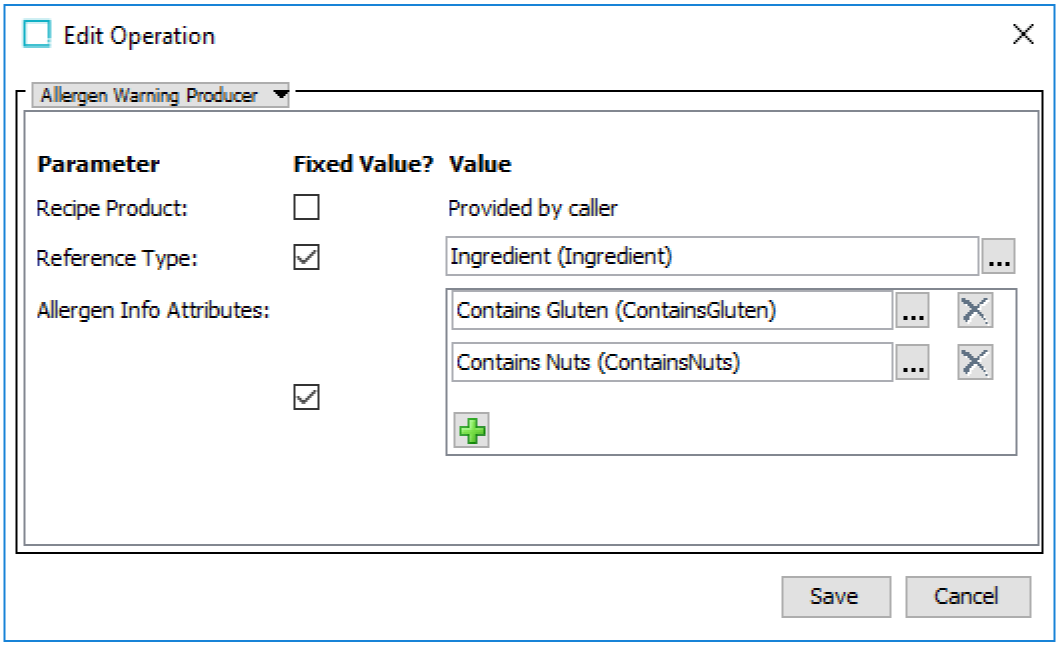
Equals (=)

Value: Yes

Or

User-Defined Functions

In addition to the out-of-the-box business function plugins, it is possible to develop configurable plugins via the Extension API which then display 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:



Parameter	Fixed Value?	Value
Recipe Product:	□	Provided by caller
Reference Type:	☑	Ingredient (Ingredient)
Allergen Info Attributes:	☑	Contains Gluten (ContainsGluten) Contains Nuts (ContainsNuts)

As displayed 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, refer to the **STEP Extension API Guide** available via the **STEP API Documentation**, at [system]/sdk or from the Start 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, refer to **Creating a Business Rule, Function, 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, refer to the online version of this topic.

Business Rules Recommended Practices

This document describes the recommended guidelines for writing business rules. The recommendations are compiled from other Stibo Systems documents and experiences from the Stibo Systems Professional Services team.

Considerations

- Writing complex business rules is best done by a software developer.
- Business rules can be maintained without involving Stibo Systems.
- Business rules provide a way to extend the behavior of your system, however, these rules may have a big impact on functionality and performance.
- Business rules can be used in imports, approval processes, workflows, Web UI screens, etc. and provide a flexible way to tailor the core functionality in a very precise manner.
- The alternative to business rules are extensions, either as written using the extension API or custom extensions developed by Stibo Systems. The advantages of extensions include their ability to run faster, as they do not have to be compiled at run-time and that they can be developed using a Java IDE, which offers code completion and syntax checking while writing the code. Extensions should be considered for complex solutions.

Recommendations

When developing business rules:

- The system supports these types of business rules: business conditions, business actions, and business functions, as well as business rule libraries, which are a collection of JavaScript methods. For more information, refer to the **Business Rules** overview topic in this documentation.
- The core concept to improving business rules is to write clear, well-documented code. Refer to the **Writing Clear Business Rules** topic in this documentation for suggested considerations.
- Business libraries are useful elements for reusing JavaScript code and across business rules. However, business libraries can cause performance issues, and in these cases, business functions may be more useful. Refer to the **Using Business Libraries** topic in this documentation.
- Should issues arise through using business rules, a good logging regimen as well as proper exception handling is critical. Refer to the **Logging** topic in this documentation. For a deep exploration of exception handling, refer to the **Recommended Error Handling Practices** topic in the **Resource Material** documentation.
- Relying on business rules is critical for many types of use cases, but business rules can impact performance if they are not maintained properly. Refer to the **Performance Considerations** topic for some practices to ensure that business rules are not hindering the system.
- To export, maintain, and test business rules, refer to the **Working with Business Rules Externally** topic.
- The **Appendix for Recommended Practices for Business Rules** contains tricks and code examples that can help you write business rules.

Writing Clear Business Rules

Clearly written code is easy to maintain and support. Readability improves when using self-explaining names and proper indentation. It is advised to keep all names and descriptions in a commonly used language within the organization so that relevant audience can understand what is being accomplished. This topic covers items to aid in clarity.

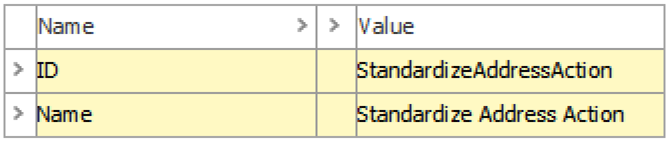
Suggested Practices

These guidelines are suggestions to aid in writing clear business rules.

Business Rules Objects

When naming business rules in the system, the following limitations should be implemented:

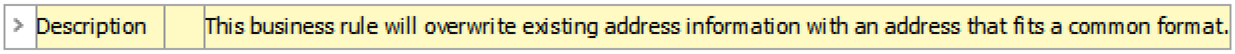
- ID: Title case, no spaces or special characters
- Name: Title case, spaces allowed



Name	Value
ID	StandardizeAddressAction
Name	Standardize Address Action

Description Field

As with most objects in the system, every business rule has a description field. Creating a short but detailed description of what the business rule is expected to do and in which context it is used will aid future users and set expectations.



Description	This business rule will overwrite existing address information with an address that fits a common format.

Bind Variables

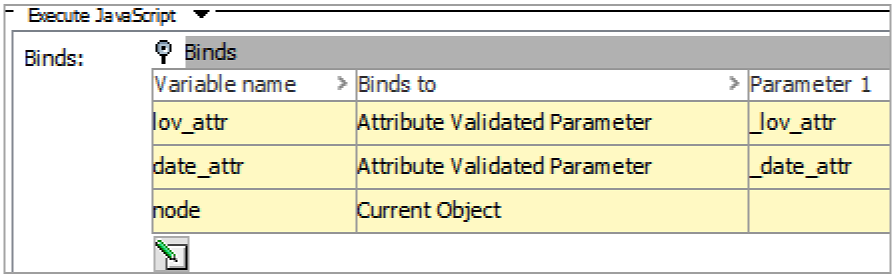
Bind variables should be named in camel case and starting with a letter. These binds should be named so that the name of the bind variable clearly identifies which bind is being used.

Bind Variables Name to Use	For Binding to Type
<attributeID>Value (unless auto-ID is used)	Attribute Value
<derivedEventTypeID>EventType, e.g. webLinkEventType	Event Type
<OIEPID>EventQueue, e.g. eCommEventQueue	Event Queue

Bind Variables Name to Use	For Binding to Type
approveContext	Approve context
currentEventQueue	Current event queue
currentObjectID	ID
currentWorkflow	Current workflow
gdsnDataMap	GDSN Data Map
logger	Logger
manager	STEP Manager

Use bind instead of hard coding IDs

When creating business rules, a bind allows for supporting a myriad of inbound data rather than a specific hardcoded ID. This is the preferred approach when a business rule does not need context-specific bind, which renders the business rule untestable using the standard business rule test functionality.



Variable name	Binds to	Parameter 1
lov_attr	Attribute Validated Parameter	_lov_attr
date_attr	Attribute Validated Parameter	_date_attr
node	Current Object	

For business rules that potentially have context specific binds, such as current workflow, it can, however, be a better approach to retrieve the workflow using the manager and a hardcoded workflow ID. This method will allow the business rule to still be testable.

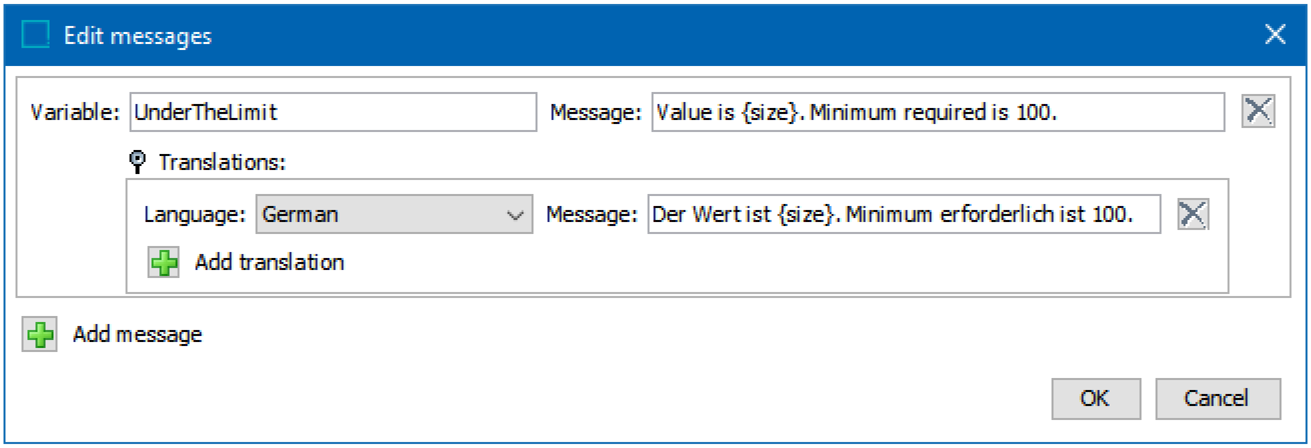
Variable and Function Names

Variable and function names should be in camel case and start with a letter. Underscore characters may be used.

Although the Rhino engine used for executing JavaScript is reinitialized prior to the execution of each JavaScript plugin script fragment, it is considered recommended practice to declare all script variables using the 'var' keyword.

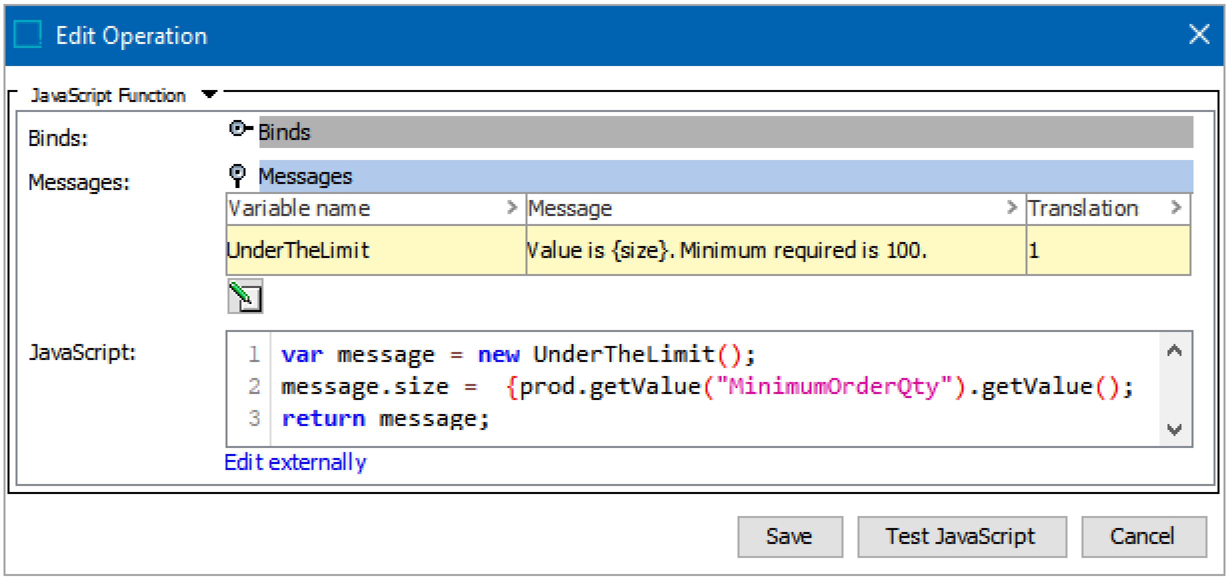
Error Messages

When defining error messages for business rules, always make sure that one of the messages is in English to make them easier to understand for a broader audience. Note that the use of the '{size}' variable allows the error message to have more meaningful contextual use.



Variable: UnderTheLimit
Message: Value is {size}. Minimum required is 100.

Language: German
Message: Der Wert ist {size}. Minimum erforderlich ist 100.



Variable name	Message	Translation
UnderTheLimit	Value is {size}. Minimum required is 100.	1

```
var message = new UnderTheLimit();
message.size =  {prod.getValue("MinimumOrderQty").getValue();
return message;
```

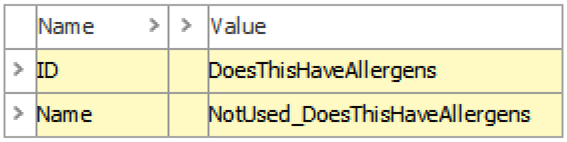
Code Layout

Business rules code lines should be 80 characters or fewer. To clearly structure the nested code values, indentations with the tab button are recommended.

```
function toCelsius(fahrenheit) {
   return (5 / 9) * (fahrenheit - 32);
}
for (i = 0; i < 5; i++) {
   sum += i;
}
if (time < 20) {
   greeting = "Good day";
} else {
   greeting = "Good evening";
}
```

General House Keeping

It is recommended to clearly mark unused business rules as being obsolete, such as by adding a 'NotUsed' prefix to the name, if they cannot be deleted. It is equally advised to mark temporary business rules with a prefix like 'Temp.'



	Name >	>	Value
>	ID		DoesThisHaveAllergens
>	Name		NotUsed_DoesThisHaveAllergens

Using Business Libraries

Business libraries are useful for writing JavaScript business rules once and referencing these rules again later even across multiple business rules. For more information, refer to the **Business Library** topic in this documentation.

Structuring Business Libraries

Use libraries to encapsulate common code which allows it to be reused by multiple business rules. Split libraries according to their overall theme:

- WorkflowUtilities
- PackagingUtilities
- NumberFormattingUtilities
- ApprovalUtilities

Limiting Business Libraries

A business rule is compiled each time it is executed. Generally, it takes about 500 milliseconds to compile about 8,500 lines of code at each business rule execution. If a business rule depends on a library, the library is compiled as well. To expand it further, if a business rule only uses a single function within a library, the whole library is still compiled.

If a library depends on another library, the other library needs to be compiled as well.

Important: To reiterate, libraries are compiled every time the business rule is executed, which is especially burdensome to performance when libraries depend on each other. Dividing a large library into multiple libraries, but keeping the dependencies, does not resolve the issue.

The system caches the compiled scripts instead of recompiling them before each execution. By default, 100 business rules are cached (Script.CacheSize=100). When the cache is filled up, the least-used business rules are evicted from the cache.

The cache reduces performance impact, but it is still recommended to keep business rules libraries small to reliably improve performance.

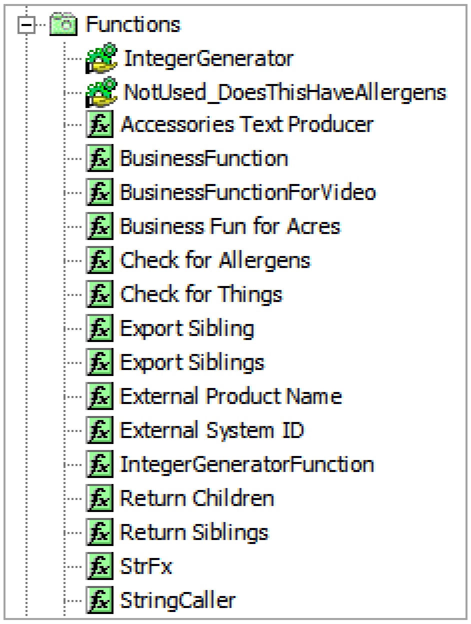
An alternative to business rule libraries is business functions. A business function is one specific function, so unlike a library, there is not any additional unused functions included and compiled.

Business Functions

Business functions, introduced in the 9.0 release, were designed to be used for certain areas of the system where business actions or conditions was insufficient due to their lack of returning a result.

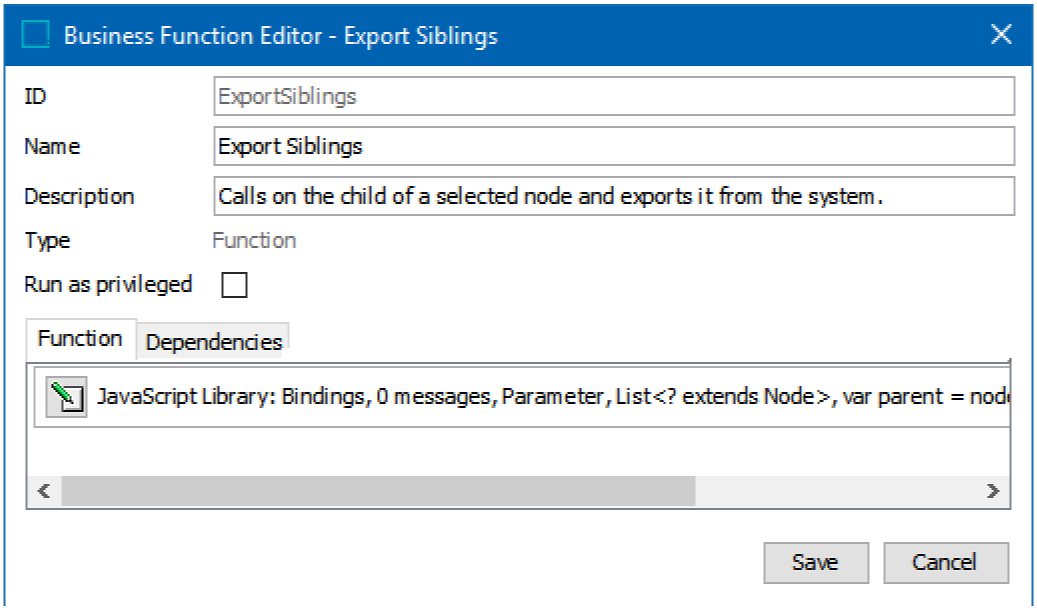
Since business functions can be used as part of a business action or condition, they also prove quite useful as a single function library. This use may be preferred to library since the function itself can be named to make sense in what it does and that the entire function is used each time. This approach opposes a library where often only parts of it is used at a given situation. One downside to a business function is that it cannot be used to alter data stored in the database.

A business function can be thought of as a JavaScript method with an input and an output. To that respect, the naming of the business function should be carefully considered so that the business function name reflects the functionality it offers.



- Functions
  - IntegerGenerator
  - NotUsed_DoesThisHaveAllergens
  - Accessories Text Producer
  - BusinessFunction
  - BusinessFunctionForVideo
  - Business Fun for Acres
  - Check for Allergens
  - Check for Things
  - Export Sibling
  - Export Siblings
  - External Product Name
  - External System ID
  - IntegerGeneratorFunction
  - Return Children
  - Return Siblings
  - StrFx
  - StringCaller

Once a proper naming convention has been established, the description field of the business function should also be filled out. It is also advised to establish any limitations to the business function, if there are any, as well as indicating what the returned element is.

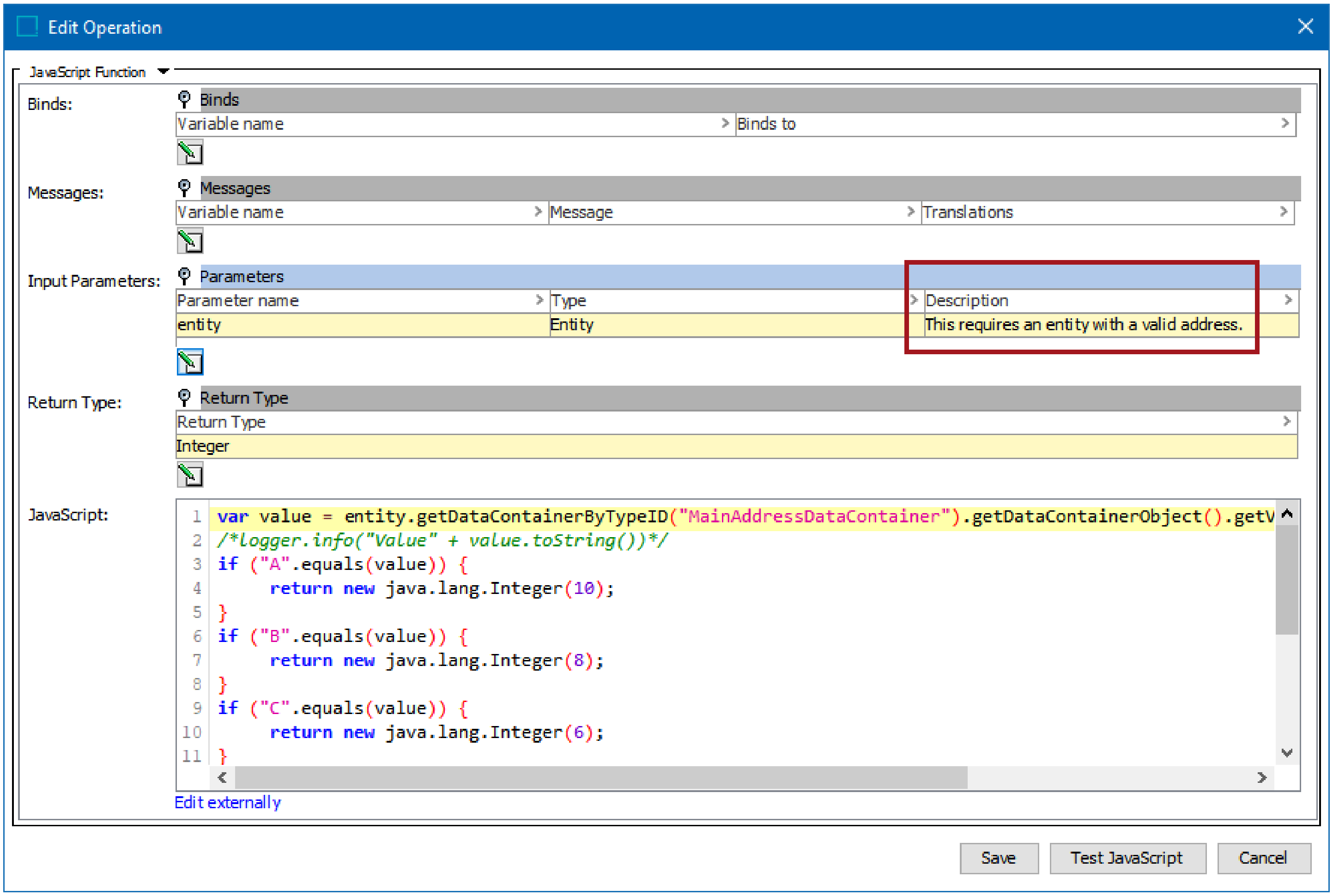


Business Function Editor - Export Siblings

ID	ExportSiblings
Name	Export Siblings
Description	Calls on the child of a selected node and exports it from the system.
Type	Function
Run as privileged	□

JavaScript Library: Bindings, 0 messages, Parameter, List<? extends Node>, var parent = nod

If there are input parameters to the business function, these should be properly named and described using the proper **Description** field when adding the parameter. Remember that this information is what the user of the business function must rely on to be able to choose the correct input parameter when calling the business function. It is important to also state assumptions / limitations if there are any, like 'parameter must not be null' or similar.



Variable name	Binds to

Variable name	Message	Translations

Parameter name	Type	Description
entity	Entity	This requires an entity with a valid address.

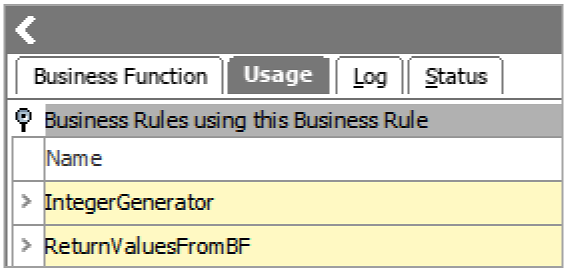
Return Type
Integer

```
var value = entity.getDataContainerByTypeID("MainAddressDataContainer").getDataContainerObject().getV
/*logger.info("Value" + value.toString())*/
if ("A".equals(value)) {
     return new java.lang.Integer(10);
}
if ("B".equals(value)) {
     return new java.lang.Integer(8);
}
if ("C".equals(value)) {
     return new java.lang.Integer(6);
}
```

The business function code should adhere to good coding practices with comments where necessary. It is recommended to use JavaScript functions or binding other business functions to structure code. Calling the business function is done by parsing a JSON object containing the input parameters to the evaluate function.

Business functions are recommended for obfuscating code, as the business function can be used as a black box, like when using libraries. Unlike libraries, a business function's purpose is more obvious. On the downside, the business function runs in non-transactional scope, and thus, cannot alter the database in any way.

One additional advantage of using the business function over the library is that the business function offers 'Usage' information about which business rules have binds to it.



Business Rules using this Business Rule

Name
IntegerGenerator
ReturnValuesFromBF

Logging and Exception Handling

Logs provide a method to figure out what is wrong with a process of the MDM system and may offer paths to solve these issues. This topic considers some of the recommended practices to log errors in the system and how to deal with exceptions.

Logging

The MDM system provides the option to set the detail level of business rules warnings and errors that should be logged in the log file. Logging many details may have a negative impact on performance, simply because the system will be busy logging these details.

It is therefore recommended to configure the business rule logging to avoid logging unnecessary details.

The amount of logging can be controlled globally (for all business rules) using the `Log.Level.com.stibo.scripting.StepScriptEngineManager` configuration property in the sharedconfig.properties file.

The values are `ALL|FINEST|FINER|FINE|CONFIG|INFO|WARNING|SEVERE|OFF` and use the appropriate level for each server environment consciously. For example:

- Set the log level details on DEV and TEST to FINE to trace errors.
- Set the log level details on QA to INFO or WARNING.
- Set the log level details on PROD to SEVERE.

It is also possible to implement a 'log level' local to a specific business rule. For the logging of business rules, it is recommended to log the result of the business rule during development on the development server but remove the logging when development of the business rule is successfully finished and deployed to the test, quality, and production servers.

The use of business rule logging can be analyzed by examining the log file. In case the log file contains business rule remarks and results, then the business rule logs to the log file.

An easy and transparent way to turn logging on and off, is to set a Debug Flag in the business rule code.

For example:

```
//Debug 'flag' REMEMBER to turn 'false' when you are done
var isDebug = false;
//Function to handle whatever logging of debug information should occur or not
function logDebug(message) {
   if(isDebug) {logger.info(message)}
}
...
logDebug("This is a message for the log file")
...
```

Exception handling

Good exception handling practices will allow developers the opportunity to prevent negative side effects to the system. For more information of error handling practices in the system, refer to the **Recommended Error Handling Practices** topic in the **Resource Material** documentation.

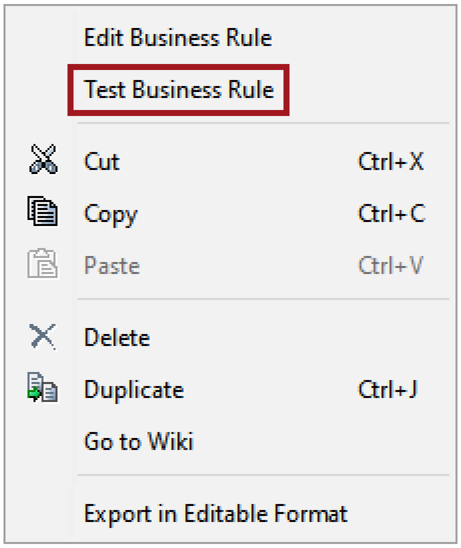
Performance Considerations

Some business rules are invoked frequently, like rules running during approval and import. To preserve the performance of the MDM system, it is important to make sure that the business rules perform well. A business rule runs in a single transaction, so it is also important that it finishes within a short time to avoid optimistic locking errors.

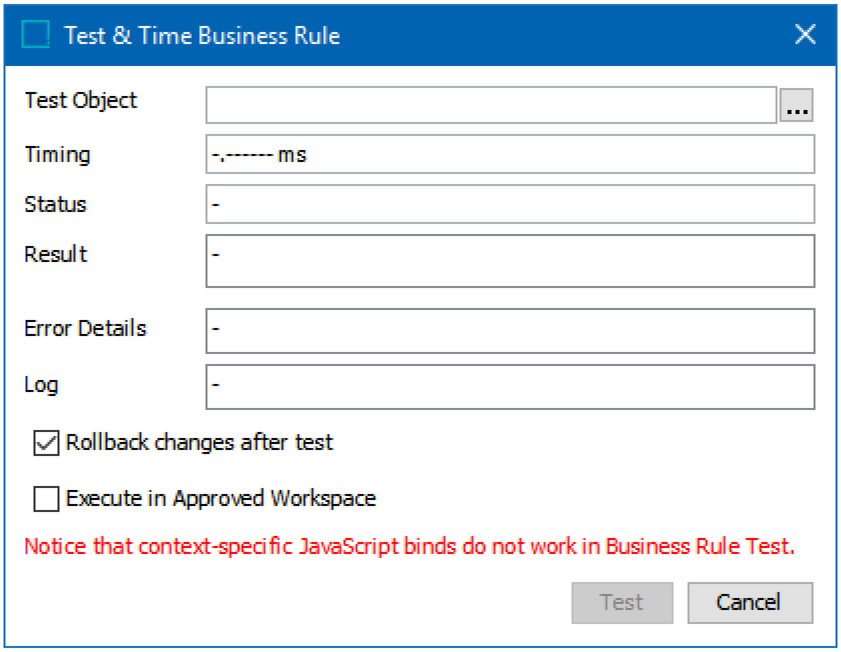
Investigating Business Rule Performance

There are several ways to analyze and monitor business rules.

1. In the workbench, there is a business rule test option that is typically used during development. To test, right-click on the business rule then select 'Test Business Rule.'



Once this option is selected, the 'Test & Time Business Rule' dialog will display. The 'Test Object' field needs to be populated with any objects that are valid for the business rule.



Test Object

Timing: -.------ ms

Status: -

Result: -

Error Details: -

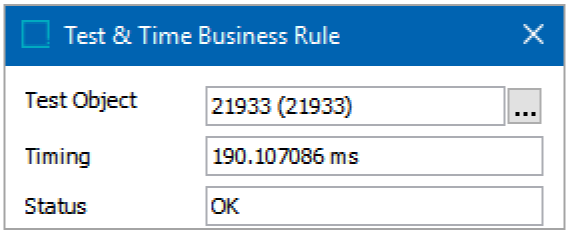
Log: -

☑ Rollback changes after test

□ Execute in Approved Workspace

Notice that context-specific JavaScript binds do not work in Business Rule Test.

Select 'Test' with an object selected. Note the timing field. Run a few different items to generate different timings, and analyze the results. The following screenshot shows that the business rule took about 190 milliseconds to complete the business rule for item 21933. Be aware that it might take longer or shorter for other items running the same rule.



Test Object: 21933 (21933)

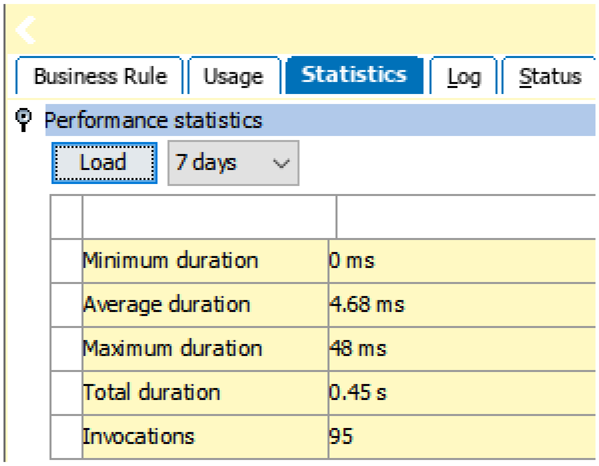
Timing: 190.107086 ms

Status: OK

2. Another way to test a business rule is in the Statistics tab of the business rule in the workbench.

   The business rule statistics tab displays minimum, maximum, average, and total duration of the business rule as well as the number of invocations per selected period. The period can be configured from a period of an hour to a week.

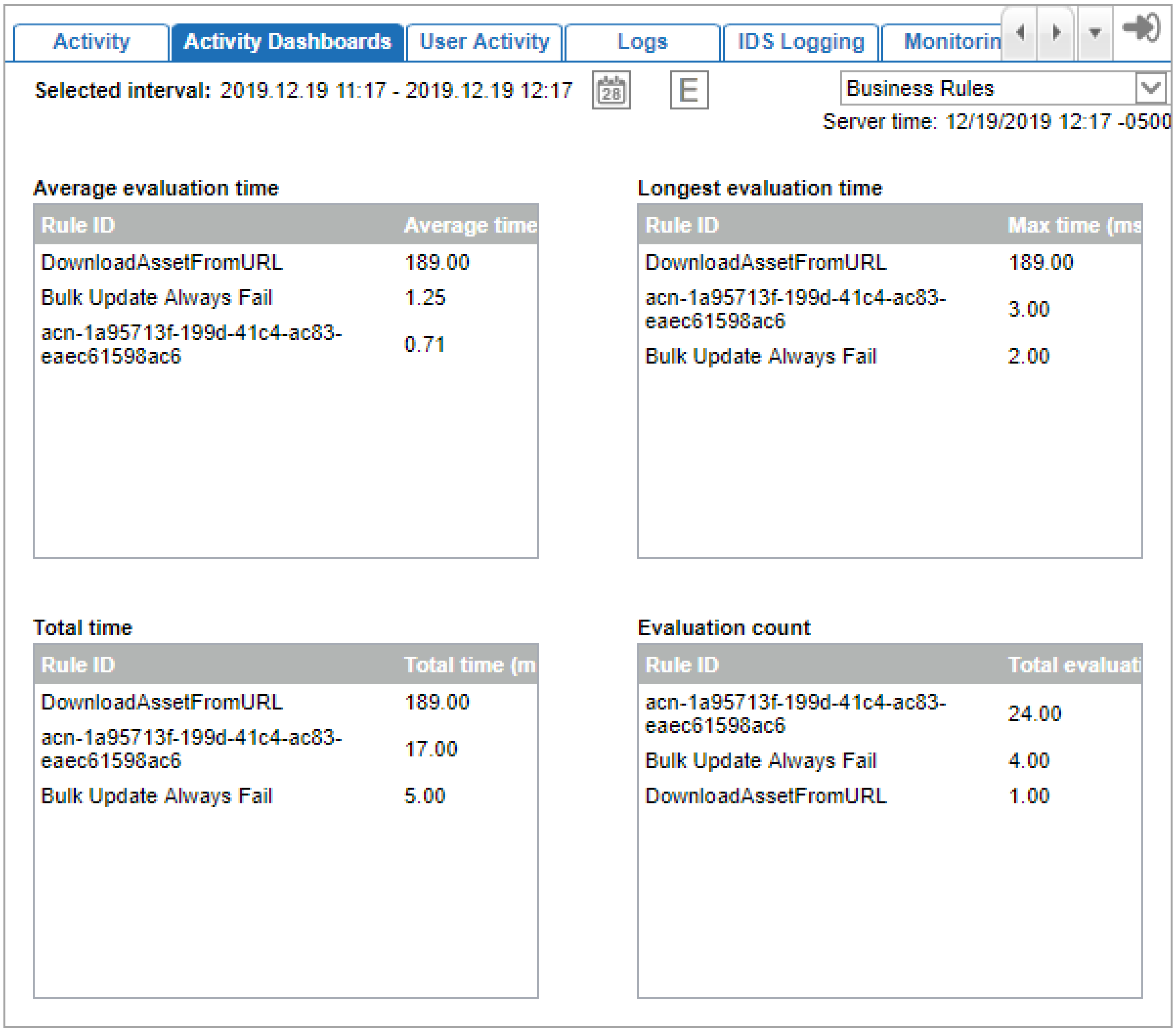
   The following screenshot shows the same business rule which was invoked 95 times during the last seven days. Through these invocations, an average duration of about 4.6 ms was calculated. If the maximum duration value of 48 ms is selected, the workbench will navigate to the record that caused the slowest time to complete.



Minimum duration	0 ms
Average duration	4.68 ms
Maximum duration	48 ms
Total duration	0.45 s
Invocations	95

This method of business rule analysis gives an indication of the business rule performance over a period of time.

3. The Admin Portal provides the possibility to track and trace business rule performance over a given period. From the Admin Portal, under Activity Dashboard, select Business Rules from the top right dropdown.



Selected interval: 2019.12.19 11:17 - 2019.12.19 12:17

Business Rules

Server time: 12/19/2019 12:17 -0500

Average evaluation time

Rule ID	Average time
DownloadAssetFromURL	189.00
Bulk Update Always Fail	1.25
acn-1a95713f-199d-41c4-ac83-eaec61598ac6	0.71

Longest evaluation time

Rule ID	Max time (ms
DownloadAssetFromURL	189.00
acn-1a95713f-199d-41c4-ac83-eaec61598ac6	3.00
Bulk Update Always Fail	2.00

Total time

Rule ID	Total time (m
DownloadAssetFromURL	189.00
acn-1a95713f-199d-41c4-ac83-eaec61598ac6	17.00
Bulk Update Always Fail	5.00

Evaluation count

Rule ID	Total evaluati
acn-1a95713f-199d-41c4-ac83-eaec61598ac6	24.00
Bulk Update Always Fail	4.00
DownloadAssetFromURL	1.00

The period over which the statistics are gathered can be configured. The dashboard shows the top business rules over the configured period, with:

- The longest average evaluation time
- The longest maximum evaluation time
- The longest total time
- The number of invocations

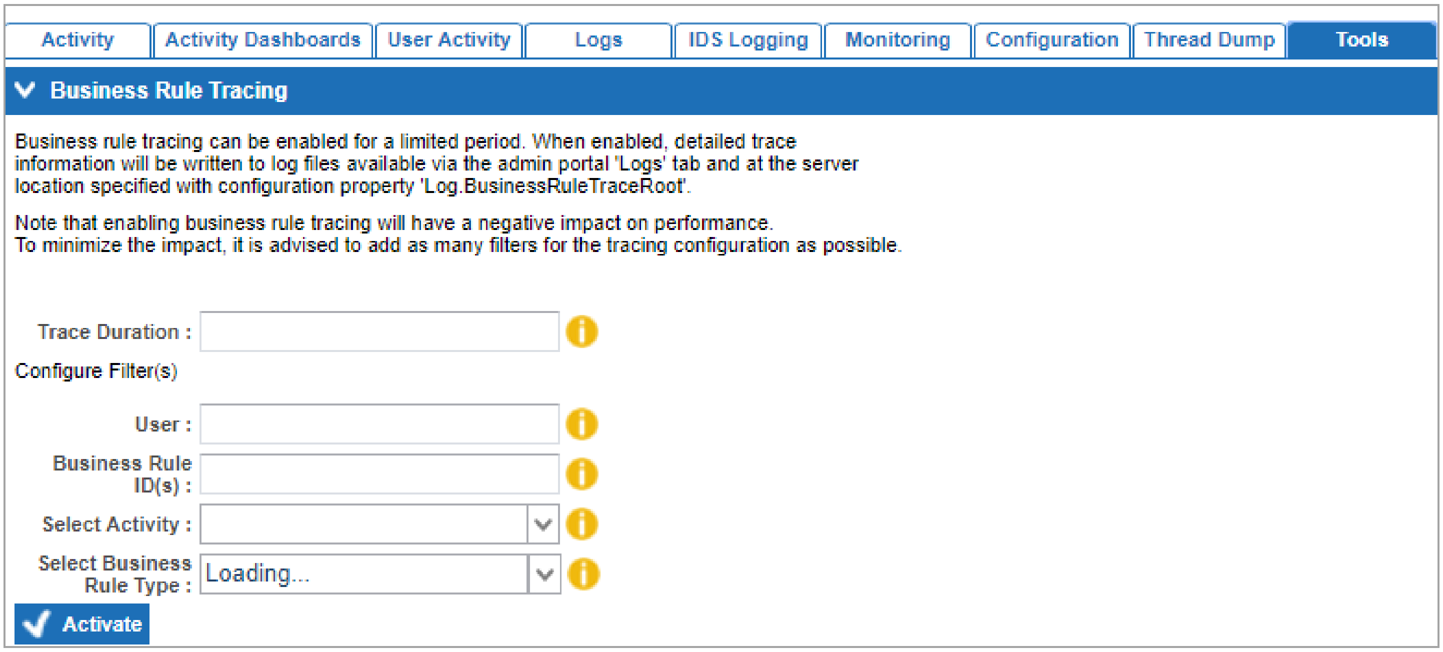
This method of business rule analysis gives an indication of the performance of the most demanding business rule over a period of time. It is very important to analyze the business rules stated under 'Total time' since these are the business rules with the longest average evaluation time and the most number of invocations.

4. There is also an option in the Admin Portal version 8.1 and above to trace business rules. The functionality of the Business Rule Tracing section of the Tools tab is described within the interface itself.

   Business rule tracing can be enabled for a limited period. When enabled, detailed trace information will be written to log files available via the Admin Portal 'Logs' tab.

> **Note:** Enabling business rule tracing will have a negative impact on performance. To minimize the impact, it is advised to add as many filters for the tracing configuration as possible.

Click the yellow information icon next to each parameter for a complete description of the parameter / filter and any relevant information for populating it.



Business Rule Tracing

Business rule tracing can be enabled for a limited period. When enabled, detailed trace information will be written to log files available via the admin portal 'Logs' tab and at the server location specified with configuration property 'Log.BusinessRuleTraceRoot'.

Note that enabling business rule tracing will have a negative impact on performance.
To minimize the impact, it is advised to add as many filters for the tracing configuration as possible.

Trace Duration :

Configure Filter(s)

User :

Business Rule ID(s) :

Select Activity :

Select Business Rule Type : Loading...

When the necessary information has been added, click the 'Activate' button to begin tracing.

> **Note:** Once tracing has been activated, the relevant business rule(s) must be triggered in the system within the time frame defined in the Trace Duration parameter so that the rule is active for tracing. Furthermore, if the system is stopped or restarted, any tracing that was in progress will also be stopped.

Tracing will stop automatically when the specified duration has expired. Alternatively, users can click the 'Stop' button, which is available only when tracing is in progress, at any time to kill the trace prior to completion of the duration.

Performance Recommendations

- Keep business rule transactions small

  Business rules have a transaction, which allows you to write data to the system. However, business rules with long transactions will degrade the performance. Furthermore, the system runs with optimistic locking policy. The longer the transaction, the larger is the probability of introducing an optimistic locking failure when running the business action simultaneously.

- Avoid the function GetChildren with many nodes

Business rules using calls 'getChildren' on a huge number of children may cause memory problems. The problem is that the 'getChildren' uses an unsafe call that will read all children. It should be changed into using 'queryChildren.'

```
public List getChildren(final String internalId, final Class wantedChildType, final
Manager manager) {
...
   } else if (obj instanceof Product) {
      Product product = (Product) obj;
      children = product. getChildren(); //also works for product overrides, but
this action is unsafe.
```

It is recommended to analyze the business rules and determine if the 'getChildren' function is not used on a selection with more than 10,000 children. If over 10,000 children are expected, change it to the 'queryChildren' function.

- Use arrays instead of multiple read calls

  Business rules repeatedly using calls to the database for large sets of data significantly degrades performance. Instead, use one call to get the data, and push it into arrays and work from there. Minimizing the number of calls to the database aids performance.

  When multiple business rules are executed sequentially (e.g., as part of an approval process), and these business rules fetch the same data from the database multiple times, it is beneficial to rewrite the business rules to fetch the data once, and push the data into (multi-dimensional) arrays or local data structures.

- Consider In-Memory for business rules

  In-Memory can improve performance of the business rules. In-Memory provides faster operations on complex data models where business rules navigate references.

  Consider using In-Memory when performance improvement on business rules is still required and all previous recommendations on business rules are implemented.

Working with Business Rules Externally

Business rules can be created, maintained, and tested outside of the system. This allowance enables users to govern the life cycle of business rules in a standard source code control system such as Git, and from there, be able to deploy appropriate versions of the business rules to the various systems that are part of a Development, Testing, Acceptance and Production (DTAP) environment.

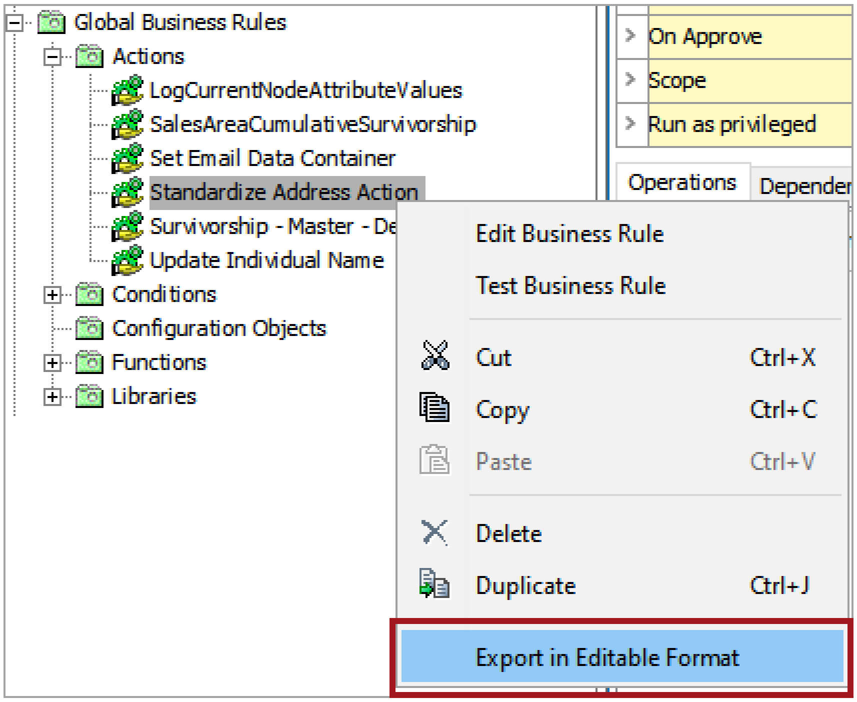
For more information about the STEP GIT integration, refer to the **Version Control System Integration** topic in the **Configuration Management** documentation.

> **Note:** While it is possible to setup and import business rules into the system, it is easier to create dependencies, binds, valid object types, and any other required components of business rules inside of the MDM system where users can browse for the information needed rather than creating these connections.

Configuration

For this use case, a preexisting business rule will be exported from the system, augmented locally, and then, imported back into the system. Experienced developers may use this template to generate completely business rules outside of the system.

From the desired business rule in the workbench, right-click then select the 'Export in Editable Format' option.



Once exported, the business rule may be edited as desired. The following code snippet is an example of a business rule that will standardize addresses.

```
// Business rule metadata omitted
/*===== business rule plugin definition =====
{
   "pluginId" : "JavaScriptBusinessActionWithBinds",
   "binds" : [ {
   "contract" : "CurrentObjectBindContract",
   "alias" : "node",
   "parameterClass" : "null",
   "value" : null,
   "description" : null
}, {
   "contract" : "ReferenceTypeBindContract",
   "alias" : "refType",
   "parameterClass" : "com.stibo.core.domain.impl.ReferenceTypeImpl",
   "value" : "PrimaryProductImage",
   "description" : null
}, {
   "contract" : "AssetBindContract",
   "alias" : "asset",
   "parameterClass" : "com.stibo.core.domain.impl.FrontAssetImpl",
   "value" : "100300",
   "description" : null
}, {
   "contract" : "LoggerBindContract",
   "alias" : "logger",
   "parameterClass" : "null",
   "value" : null,
   "description" : null
} ],
   "messages" : [ ],
   "pluginType" : "Operation"
}
*/
exports.operation0 = function (node,refType,asset,logger) {
// "Current Object" bound to "node"
// A reference type bound to "refType"
// An asset bound to "asset"
var existingRefs = node.getReferences(refType).toArray();
if (existingRefs.length == 0) {
   logger.info("Creating reference");
   node.createReference(asset, refType);
} else {
   logger.info("Asset " + asset.getID() + " already has a Primary Product Image");
}
}
```

Since the system uses binds to connect data to JavaScript, those are best maintained after the edited JavaScript file is imported back into the system. Once completed, it is recommended that developers test the business rule to ensure it functions as expected. For more information, refer to the **Testing a Business Rule** topic in this documentation.

Writing User Notification Messages for Business Rules

Business rules enable admin users to apply a wide range of automation to master data processes. Included in business rules' range of features is the ability to display messages to the user at specific moments, and under specific circumstances. Some messages may convey a bit of information relevant to an action that was just taken, or an acknowledgment that the desired outcome has been achieved, but most often, a configured message to the user will display when something has gone wrong.

Because so much flexibility is supported for creating business rules, a lot of latitude is granted to admin users configuring business rules, specifically as it relates to drafting text notifications. The information contained in this topic aims to provide admin users with a primer on recommended practices to consider when writing business rule notifications, so the end user may be provided with information that is useful, clear, and timely.

General Guidelines

The aim for notifications should be to provide the user with concise, informative messages that, depending on the kind of notification, inform the user a process has completed, prompt the user to take a specific next step, or confirm whether an action is successful or has failed. Automated system notifications that only provide jargon-rich technical information are not as strong as those that offer plain-language messages designed to help the user take the next step. Notifications can be displayed via popup messages (as in the 'Information' and 'Acknowledgment' message type screenshots below), or as text shown below the affected field in a Node Editor (as in the 'Warning' and 'Error' message type screenshots, also below), or as small pop-ups that display near the affected cell(s) in Node Lists.

There are four types of notifications that can be presented to users in business rules: Information, Acknowledgment, Warning, and Error. What follows is a description of each of these, with an emphasis on Warning and Error as these are the notification types that will most commonly be used with business rules.

Information

An information notification informs the user of an event that may be of interest to the user. These notifications are always system-initiated.

Message Template

Information messages inform the user of a positive event that is likely to be of interest to the user. Unlike error messages, information notifications are not associated with a problem, so the triggering event is the most relevant piece of information, whereas with an error, the problem itself is the most relevant piece of information.

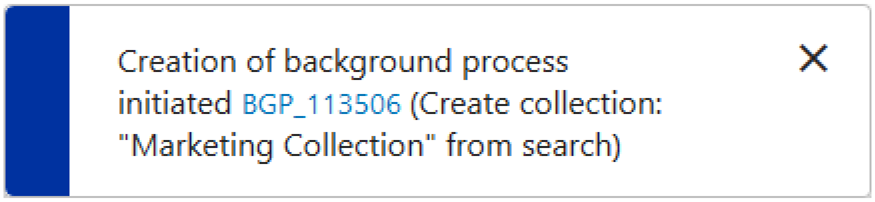
What follows is the recommended template for an informational message:

{{Event}}. {{Further Information}}. refer to {{Reference to all information}}.

A good example of this kind of message might be, 'Creation of background process BGP_113506 initiated. (Create collection: “Marketing Collection” from search)'

In this example, the relevant event is described. Following this, the admin-named event that has initiated the background process, which is ‘Create collection’, the collection itself is named ‘Marketing Collection’, and the collection is being created from the results of a search.

When properly configured in a business rule using the Web UI Context bind, information messages will be coded blue.



Creation of background process initiated BGP_113506 (Create collection: "Marketing Collection" from search)

Acknowledgment

An Acknowledgment message informs the user that their goal was fulfilled. These messages might best be described as the opposite of error messages. Acknowledgment messages are always generated by the successful completion of user-initiated tasks.

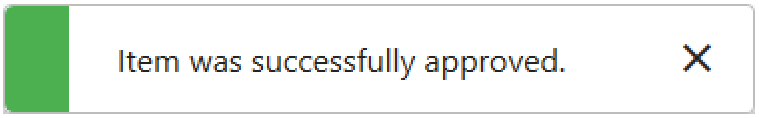
What follows is the recommended template for an acknowledgment message:

{{Object(s)}} {{action}}

A good example of this kind of message would be: 'Item was successfully approved.'

In this case, 'Item' is the object, and 'successfully approved' is the action.

When properly configured in a business rule using the Web UI Context bind, acknowledgment messages will be coded green.



Item was successfully approved.

Warning

A Warning message warns users of a condition that might cause a problem in the future. Conditions that generate warning messages can stem from either user- or system-initiated actions. The fundamental purpose of a warning message is to help the user lessen the risk of a potential negative consequence. A warning message might display, for example, when a user is at risk of deleting an asset, losing system access, or executing an action that will take significant time to correct.

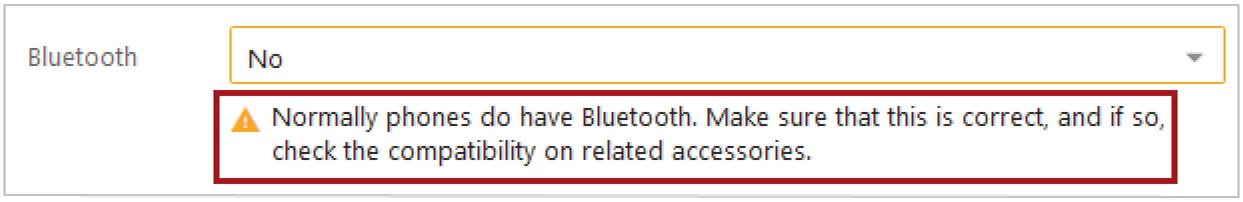
What follows is the recommended template for a warning message:

{{Point of concern}}, because {{cause}}. {{Solution}}

A good example of this kind of message is in the Node Editor warning notification shown below.

In this instance, the stated point of concern is 'Normally phones do have Bluetooth.' The solution statement is, 'Make sure that this is correct, and if so, check the compatibility on related accessories.' This message is clear, written in plain language, and alerts the user to a potential problem, and provides steps they can take to avoid an error going forward.

When properly configured in a business rule, popup warning messages will be coded amber, and warning messages for Node Editor fields (using the Data Issues Report bind) will be preceded with an amber triangle, and turn the outline of the affected field amber.



Bluetooth: No

⚠ Normally phones do have Bluetooth. Make sure that this is correct, and if so, check the compatibility on related accessories.

Error

An Error message lets the user know two things: one, a problem has occurred and two, the user cannot proceed until the problem has been cleared. The issue being raised can stem from either a user- or system-initiated action. Effective error messages inform users not only what the problem is, but also provide a brief explanation of its cause, as well as information that can point users towards a fix. The error message should prompt users to either perform an action or change their behavior.

Some examples of events that can trigger error messages are the following: input problems, Create, Read, Update, and Delete issues (CRUD), violations of business rules, business-relevant system to system problems, server connection problems, database problems, and network problems.

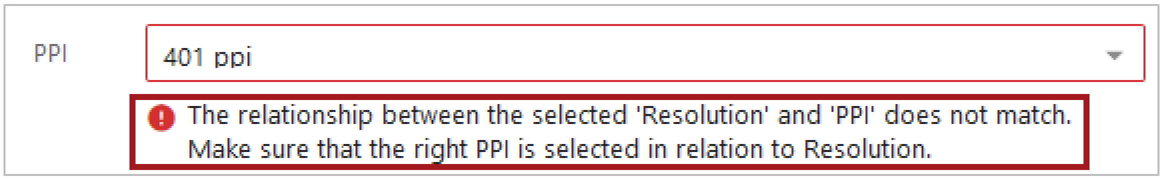
What follows is the recommended template for an error message:

{{Problem}}, because {{cause}}. {{Solution}}

A good example of this kind of message is in the Node Editor error notification shown below.

In this instance, the stated problem is 'The relationship between the selected 'Resolution' and 'PPI' does not match.' The solution statement is, 'Make sure that the right PPI is selected in relation to Resolution.' This message is clear, written in plain language, and gives the user a clear course of action they can follow to clear the error and proceed.

When properly configured in a business rule, popup error messages will be coded red, and error messages for Node Editor fields (using the Data Issues Report bind) will be preceded with a red circle, and turn the outline of the affected field red.



PPI: 401 ppi

❗ The relationship between the selected 'Resolution' and 'PPI' does not match. Make sure that the right PPI is selected in relation to Resolution.

Appendix for Recommended Practices for Business Rules

This appendix contains some guidelines and code examples that can help users to write business rules.

Comparison of Objects

The comparison operators are used to compare two values in a Boolean fashion. The standard available comparators in JavaScript are:

Operator	Name	Example	Result
==	Equal	x == y	True, if X is equal to Y
===	Identical	x === y	True, if X is equal to Y, and they are of the same type
!=	Not Equal	x != y	True, if X is not equal to Y
!==	Not Identical	x !== y	True, if X is not equal to Y, or they are not of the type
<	Less Than	x < y	True, if X is less than Y
>	Greater Than	x > y	True, if X is greater than Y
<=	Less Than or Equal To	x <= y	True, if X is less than or equal to Y
>=	Greater Than or Equal To	x >= y	True, if X is greater than or equal to Y

Note: Since business rules are running on a Java runtime environment, there can be differences in what is otherwise perceived as being the same.

The following example uses String objects. Two strings are compared, with one being in Java while the other is JavaScript.

```
var someJSString = “xyz”;
var comparison = someJSString == product.getValue(”<ID>”).getSimpleValue();
```

This comparison will always produce 'False' no matter if the value on the product was 'xyz.' In instances when comparing a String, it is therefore recommended to always use the `.equals()` method available on both JavaScript and Java:

```
var someJSString = “xyz”;
var comparison = someJSString.equals(product.getValue(”<ID>”).getSimpleValue());
```

Looping Children of an Object

When looping through the children of an object, the user should refrain from calling the `getChildren()` method as it consumes memory, and in cases when not all children need to be evaluated, the `getChildren()` is ineffective. Instead the `queryChildren()` method should be used since this method only picks the next object when the previous object has been processed.

This code snippet demonstrates a process of looping through children using an anonymous inline implementation of the `queryChildren()` consume method.

```
var childrenQuery = node.queryChildren();
childrenQuery.forEach(function(child) {
    logger.info(child.getTitle());
    return false; // break the "forEach" on the query
});
```

Conversely, this example shows a process of looping through children using an explicit implementation of the `queryChildren()` consume method, which in this use case is `printTitle`.

```
function printTitle(child) {
    logger.info(child.getTitle());
    return true; // continue the "forEach" on the query
}
var childrenQuery = node.queryChildren();
childrenQuery.forEach(printTitle);
```

Searches

While it is possible to do searches as part of the Scripting API, one should be careful about using this approach as they can easily lead to very long running times of the scripts.

Important: The following examples are not valid with all platform release versions.

For example, in a system that is running on a version earlier than 9.0, the following code snippet will perform a single attribute search on Product object types.

```
function singleAttributeSearchProduct(manager, attribute, value, maxResult)
{
var config = new
com.stibo.core.domain.singleattributequery.SingleAttributeQueryHome.SingleAttributeQ
uerySpecification(com.stibo.core.domain.Product, attribute,value);
    var home = manager.getHome
(com.stibo.core.domain.singleattributequery.SingleAttributeQueryHome);
    return home.querySingleAttribute(config).asList(maxResult);
}
```

> **Note:** No matter what is chosen as `maxResult`, the result list cannot be more than 100 values as the query will simply be cut of at this time.

In this example using a 9.0 or greater system, a single attribute is returned for all Product object types.

```
function breakingQueryConsumer(node) {
  logger.info(node.getTitle());
  logger.info("Breaking...");
  return false; // break the "forEach" on the query
}

function continuingQueryConsumer(node) {
  logger.info(node.getTitle());
  logger.info("Continuing...");
  return true; // continue the "forEach" on the query
}

var singleAttributeQueryHome =
manager.getHome
(com.stibo.core.domain.singleattributequery.SingleAttributeQueryHome);
var conditions = new
com.stibo.core.domain.singleattributequery.SingleAttributeQueryHome.SingleAttributeQ
uerySpecification(com.stibo.core.domain.Product,descriptionAttribute, "test");
var query = singleAttributeQueryHome.querySingleAttribute(conditions);

query.forEach(breakingQueryConsumer);
query.forEach(continuingQueryConsumer);
```

In a 9.0 or greater system, users may use the queryAPI to search.

> **Note:** To use the queryAPI, the query add-on component needs to be installed. For on-premise systems, instructions for installing components can be found in the **SPOT Program** topic in the **System Administration Guide** found in **Downloadable Documentation**. For SaaS systems, contact your Stibo Systems account manager.

```
var conditions = com.stibo.query.condition.Conditions;

// create a below condition
var isBelowCondition = conditions.hierarchy().simpleBelow(productsRoot);

// create an attribute value condition
var hasValueTestCondition = conditions.valueOf(descriptionAttribute).eq("test");

var queryHome = manager.getHome(com.stibo.query.home.QueryHome);

// query where both conditions are met (and).
```

```
var querySpecification = queryHome.queryFor(com.stibo.core.domain.Product).where
(isBelowCondition.and(hasValueTestCondition));

var result = querySpecification.execute();
result.forEach(showTitle);

function showTitle(node) {
  logger.info(node.getTitle());
  return true;
}
```

The following example also leverages the queryAPI on 9.0 or newer systems.

> **Note:** To use the queryAPI, the query add-on component needs to be installed. For on-premise systems, instructions for installing components can be found in the **SPOT Program** topic in the **System Administration Guide** found in **Downloadable Documentation**. For SaaS systems, contact your Stibo Systems account manager.

```
function searchProductByAttributeValueAndObjectType(manager, objectTypeID, attrID,
value) {
  var conditions = com.stibo.query.condition.Conditions;
  var hasValueTestCondition = conditions.valueOf(manager.getAttributeHome
().getAttributeByID("" + attrID)).eq("" + value);
  var hasObjectTypeCondition = conditions.objectType(manager.getObjectTypeHome
().getObjectTypeByID(objectTypeID));
  var queryHome = manager.getHome(com.stibo.query.home.QueryHome);
  var querySpecification = queryHome.queryFor(com.stibo.core.domain.Product).where
(hasValueTestCondition.and(hasObjectTypeCondition));
  var result = querySpecification.execute();
  return result;
}
```

Date Handling

While comparing dates can introduce complications, Java is offers some tools to compare dates against each other. This application allows users, for instance, to determined if a date is before or after another date.

In the following example, the code snippet determines if a date has passed already.

```
function hasISODateBeenExceeded(dateString) {
  return hasDateBeenExceeded(dateString, "yyyy-MM-dd");
}
function hasDateBeenExceeded(dateString, pattern) {
  var now = java.time.LocalDate.now();
  var parsed = java.time.LocalDate.parse(dateString,
java.time.format.DateTimeFormatter.ofPattern(pattern));
  return now.isAfter(parsed);
}
// hasISODateBeenExceeded("2017-08-17") will return true as the date has been
exceeded (compared to now)
```

Additional Use Cases

The following JavaScript examples are some of the scenarios that business rules may leverage JavaScript. Where applicable, assume that the 'node' is a bind to the 'Current Object' while 'workflow' is a bind to the 'Current Workflow.'

Setting a Static Value

```
node.getValue("AttributeID").setSimpleValue("No");
//OR do the same thing by using the LOV value ID
node.getValue("AttributeID").setLOVValueByID("N");
```

Creating a Child of Current Object

```
var newChild = node.createProduct(""/*ID (optional for object types with auto ID)*/,
"ObjectTypeID");
```

Creating a Reference that is intended for Reference Types that Allow Multiple References

```
//An error will occur if you try to create a reference between two objects if a
reference of that reference type already exists
//This function returns the existing reference if it already exists instead of
producing the error
function createReferenceOrGetExisting(source, target, referenceType) {
  for (var references = source.getReferences(referenceType).iterator();
references.hasNext(); ) {
    var reference = references.next();
    if (reference.getTarget().equals(target)) {
      return reference;
    }
  }
  return source.createReference(target, referenceType);
}

//A similar function to the previous one except it deletes the existing reference
before creating a new reference (namely for reference types that only allow one
reference)
function replaceSingleReference(source, target, referenceType) {
  for (var references = source.getReferences(referenceType).iterator();
references.hasNext(); ) {
    var reference = references.next();
    if (reference.getTarget().equals(target)) {
      return reference;//Do nothing if it exists
    } else {
      reference.delete();
      break;
    }
  }
  return source.createReference(target, referenceType);
}
var anAsset = node.getManager().getAssetHome().getAssetByID("AssetID");
var referenceType = node.getManager().getReferenceTypeHome().getReferenceTypeByID
("ReferenceTypeID");
var reference = createReferenceOrGetExisting(node, anAsset, referenceType);
```

Auto-submit Node to Next State

```
//A rule like this would be configured On Entry for a state
var task = node.getTaskByID(currentWorkflowBind.getID(), "StateID");
task.triggerLaterByID("TransitionID", "A message viewable in the state log");
```

Partial Approval

```
//This rule approves this object's location in the hierarchy.
//If this object has never been approved, your set of part objects would have to
contain a ParentPartObject to run any partial approval successfully.
var changesToApprove = new java.util.HashSet();
for (var partObjects = node.getNonApprovedObjects().iterator(); partObjects.hasNext
(); ) {
  var partObject = partObjects.next();
  if (partObject instanceof com.stibo.core.domain.partobject.ParentPartObject) {
    changesToApprove.add(partObject);
    break;
  }
}
node.approve(changesToApprove);
```

Starting a Workflow and Setting a Variable

```
var workflowInstance = node.startWorkflowByID("WorkflowID", "An optional message to
show in the state log");
if (workflowInstance) {//It may have failed to start the workflow due to a start
condition
  workflowInstance.setSimpleVariable("WorkflowVariableID", "A text value");
}
```

Units with LOVs

Values in the system are stored as both the value and the unit. For example, if viewing an attribute with a value of '10g' in Web UI, the following example code will give access to that value, that value without the unit, and only the unit.

```
logger.info(node.getValue("SingleValueWithUnit").getSimpleValue());//10 g
logger.info(node.getValue("SingleValueWithUnit").getValue());//10
logger.info(node.getValue("SingleValueWithUnit").getUnit().getTitle());//g
logger.info(node.getValue("YesNoAttribute").getSimpleValue());//Yes
logger.info(node.getValue("YesNoAttribute").getID());//Y
```

Reading a Multi-value Attribute and Adding a Value

```
function getMultiValueAsStrings(valueObject) {
  var simpleValue = valueObject.getSimpleValue();
  return simpleValue ? simpleValue.split("<multisep/>") : [];
}
var value = node.getValue("MultiValuedAttribute");
var strings = getMultiValueAsStrings(value);
if (strings.indexOf("New Value") == -1) {//Ordinarily, a duplicate value does not
make sense
  value.addValue("New Value");
}
```

Setting a Deadline Relative to Current Date

This example is run on a workflow set the deadline that is not a precise one but related to the current date.

```
var calendar = java.util.Calendar.getInstance();
calendar.add(java.util.Calendar.DATE, 1);//Set to tomorrow
workflow.getTaskByID("WorkflowID", "StateID").setDeadline(calendar.getTime());
```

Setting a Deadline to a Particular Date

This example is also run on a workflow and set an specific date.

```
var dateFormat = new java.text.SimpleDateFormat("dd/MM/yyyy");
workflow.getTaskByID("WorkflowID", "StateID").setDeadline(dateFormat.parse
("20/01/2021"));
```

Reading Asset Content

```
if (asset.hasContent()) {
  try {
    var outputStream = new java.io.ByteArrayOutputStream();//it is worth considering
writing to a file to save memory for large assets
```

```
    asset.download(outputStream);
    var inputStream = new java.io.ByteArrayInputStream(outputStream.toByteArray());
    var bufferedReader = new java.io.BufferedReader(new java.io.InputStreamReader
(inputStream));
    for (var line = bufferedReader.readLine(); line; line = bufferedReader.readLine
()) {
      logger.info(line);
    }
} finally {
    if (bufferedReader) {
      bufferedReader.close();//If we were using something other than ByteArray
streams, closing becomes important
    }
  }
}
```

Writing an E-mail

```
mailHome.mail()
  .addTo("toAddress@acme.com", "Optional name")
  .from("fromNoReplyAddress@acme.com", "Optional name")//Uses
Mail.DefaultFromMailAddress property by default
  .subject("Subject of Email")
  .htmlMessage("<div>The body of the email</div>")//As opposed to plainMessage
  .attachment()//This returns an Attachment object, not the Mail
    .fromAsset(asset)
    .name("TheAttachment.jpg")//Optional
    .attach()//Calling attach finishes the creation of the attachment and returns
the mail object again
.send();
```

Creating a Classification Product Link

```
var classification = manager.getClassificationHome().getClassificationByID
("AClassificationID");
var linkType = manager.getHome
(com.stibo.core.domain.classificationproductlinktype.ClassificationProductLinkTypeHo
me).getLinkTypeByID("LinkTypeID");
try {
  var classificationProductLink = node.createClassificationProductLink
(classification, linkType);//classification.createClassificationProductLink(node,
linkType); works too
  //Do things with new link, like write metadata
} catch (e) {
  if (e.javaException instanceof com.stibo.core.domain.UniqueConstraintException) {
    logger.info("Link already exists.");
  } else if (e.javaException instanceof
com.stibo.core.domain.LinkTypeNotValidException) {
    logger.info("Link type not valid for this product and classification");
} else {
    throw(e);
  }
}
```