BUSINESS RULES
USER GUIDE

StiboSystems

STEP Trailblazer 8.1

Table of Contents

Business Rules

Business rules are pieces of business logic that are stored as objects in System Setup. You can use business rules, for example, in workflows, during object approval, in imports, in bulk updates, in data profiling, and in the STEP Web UI.

There are two main types of business rules:

- A **business condition** is logic that should only evaluate to either true or false. Side effects in a condition are not supported (such as performing an approval, setting a value, or adding links / references). For more information on business conditions, see **Business Conditions**.
- A **business action** is logic that when executed, typically performs an action and does have side effects. For more information on business conditions, see **Business Actions**.

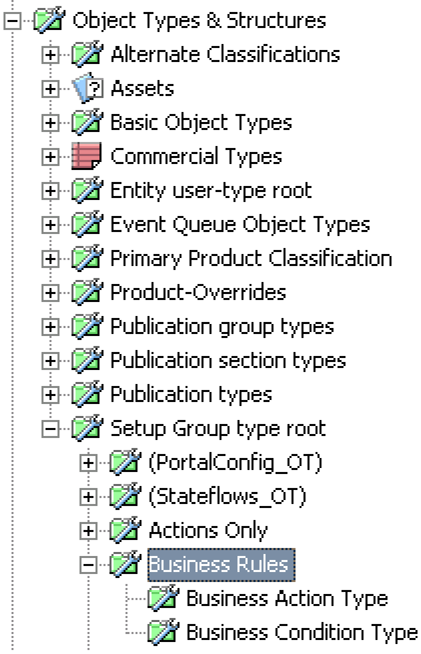
Setting up Business Rules

If your system does not already have a setup group for business rules, use the following steps to create one. The business rule setup group allows business actions and business conditions as child nodes.

To Set up Business Rules

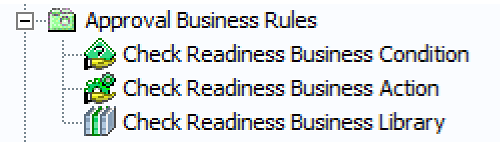
1. In **System Setup**, expand **Object Types & Structures**.
2. Right-click **Setup Group type root**, choose **New Object Type**.
3. Enter an **ID** and a **Name** such as Business Rules and then click **Create**.
4. Right-click the object type you just created, and add two object types. You can, for example, name them Business Condition Type and Business Action Type.
5. In **Object Types & Structures**, expand **Basic Object Types**, and then select the **Business Action Type** you just created.
6. On the **References** tab, click **Add Parent**.
7. In the **Select New Parent** dialog, select the setup group you just created and click **Select** to make it a valid parent.
8. From the **Maintain** menu, point to **Insert**, and then choose **Setup Group Root**.
9. Select the setup group object type you just created, enter an **ID** and a **Name**, and then click **Create**. An instance of the object type is created in **System Setup**.
10. In **Object Types & Structures**, expand **Basic Object Types**, and then select **Business Condition Type**.
11. Repeat steps 6-9 for business conditions.

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



```
Object Types & Structures
    Alternate Classifications
    Assets
    Basic Object Types
    Commercial Types
    Entity user-type root
    Event Queue Object Types
    Primary Product Classification
    Product-Overrides
    Publication group types
    Publication section types
    Publication types
    Setup Group type root
        (PortalConfig_OT)
        (Stateflows_OT)
        Actions Only
        Business Rules
            Business Action Type
            Business Condition Type
```

The following shows an instance of the business rules setup group object type named Approval Business Rules. It contains a business condition named Check Readiness Business Condition, a business action named Check Readiness Business Action and a business library named Check Readiness Business Library.



- Approval Business Rules
  - Check Readiness Business Condition
  - Check Readiness Business Action
  - Check Readiness Business Library

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

Working With Business Rules

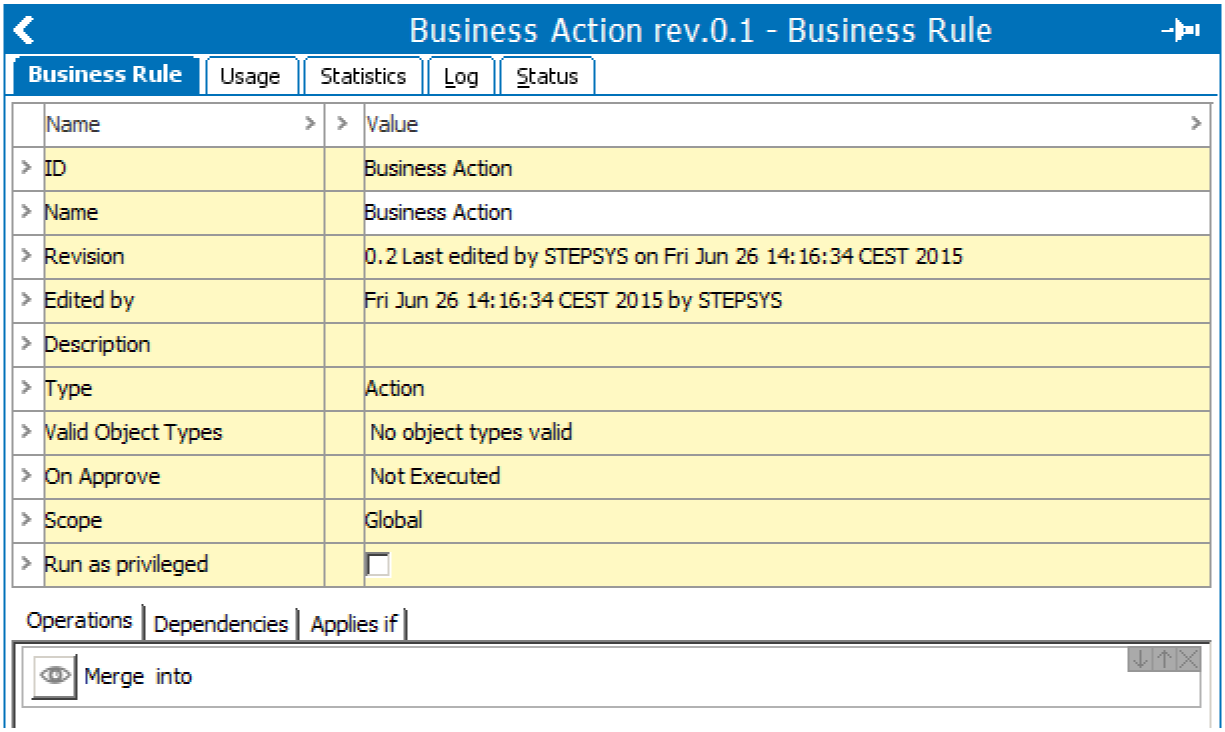
The various settings for business rules are specified in the Business Rule editor. The following describes how to work with global business rules. For information about local business rules, see **Using Business Rules in STEP Workflows** in the **STEP Workflows** documentation.

To View a Business Rule

In **System Setup**, expand **Business Rules**, then select the relevant business rule. This opens the business rule in an editor in a read-only mode. (Business rules viewed in the Approved Workspace are read-only and can only be edited from the Main workspace.)

Business Rule Tab

The **Business Rule** tab displays information about ID, Name, Edited by, Description and Type, as well as settings such as valid object types, scope, and how the action or condition is used on approval (On Approve).



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 contents of the tabs **Operations**, **Dependencies** and **Applies if** are also read-only. To view the details, click the **View Operation** icon.

The Scope of a Business Rule

The scope of a business rule can be local or global. Business rules created via a workflow are scoped as local by default, while business rules created via the System Setup navigator are scoped as global by default.

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 product enters a specific workflow state. Local business rules are not available from a business rule selector or the System Setup navigator. Local business rules are typically located below the workflows where they are used.

Different icons are used for conditions and actions and each exists in two variants. The global business rule icons contain a hand: condition , action . The variant with no hand is used for local business rules.

Usage tab

The **Usage tab** displays information about where the business rule is used. You can see the workflows, business rules, objects, and assets that use the 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.

Log tab

The **Log tab** shows all changes that have been made to the 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. Minor revisions are made automatically when various users interact with the same object.

- Minor revisions are created automatically when multiple users edit and save business rule metadata (Name, Description, Valid Object Types, On Approve, Run as Privileged).
- Minor revisions are created when single and/or multiple users edit and save business rule content (Operations, Dependencies, Applies if).

Users can initiate a major revision by selecting the object and choosing **Make Revision** from the Maintain menu.

Via the **Status tab** it is possible to browse and view the various revisions of a business rule. Previous versions of business rules can be viewed as read-only by clicking the eye icon for the rule.

It is also possible to **Revert to** a previous version, as well as **Purge** revisions.

To Edit a Business Rule

To edit the business rule, click **Edit business rule** in the lower left corner. For more information, see Editing Global Business Rules.

To Revert a Business Rule to a Previous Version

On the **Status** tab, click the version that you want to revert to, then click **Revert to**. **Revert to** can also be reached by selecting the revision on the Status tab and right-clicking on the row indicator (>).

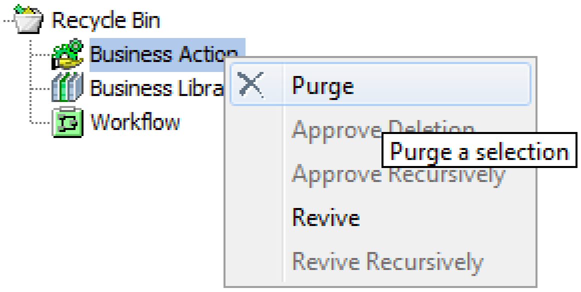
Note: When reverting to a previous version of a workflow, any local business rules owned by the workflow in question will also be reverted to the revisions that were current at the time of the workflow revision in question. For more information, see the Managing Workflows section of the Workflows documentation.

To Purge a Business Rule Revision

On the **Status tab**, click the version that you want to purge, and then click **Purge**.

To Delete a Business Rule

Deleted business rules can be seen in the System Setup Recycle Bin and can be revived from that location. Similarly they can be purged.



For more information, see Recycle Bin.

To Run a Business Rule

When the business rule runs, it first checks if the object type is valid for the business rule. If it is not valid, the result of the business rule is "not applicable". If it is valid, the "applies if" operations is checked. If this is not valid, the business rule is "not applicable". If it is valid, the actual business rule operations are run.

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.

Different icons are used for 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.

Editing Global Business Rules

You specify the various settings for a business rule in the Business Rule editors. This is handled differently for Global and Local business rules.

For more information, see the **Global and Local Business Rules** section of the **Business Rules** documentation.

The editing of global business rules is done via the relevant editor, available via a right-click action and **Edit Business Rule** link on the Business Rule tab.

Editing and saving via a Save button will create minor revisions visible on the Status tab.

Using the Business Rule Editor, you can edit everything except the ID, Type, and Scope.

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

The On Approve and Valid Object Types settings are only displayed when the Business Rule is Global. Local Business Rules following the setting of the governing workflow.

To Run a Business Rule on Approve

In the On Approve field you specify if the business rules should be run during an approval process. This only applies to global business rules.

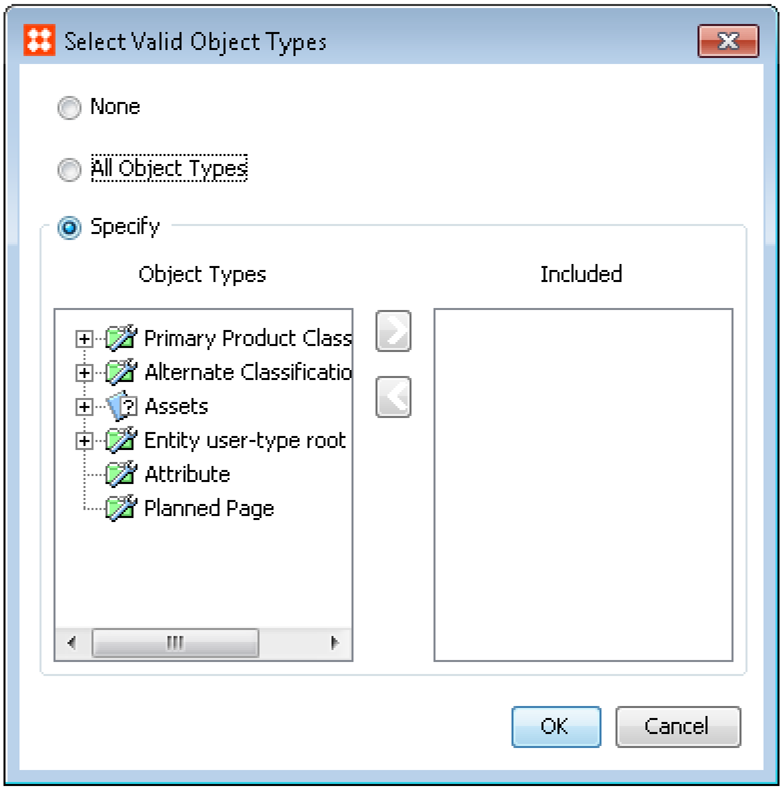
- On conditions, you specify whether to perform the validations before or after approval.
- On actions, you specify whether to execute the action on Approve, and whether to approve changes as well. Execution takes place in the main workspace prior to the actual approval.

For more information, see **Using Business Rules During the Approval Process**.

To Specify Valid Object Types for a Business Rule

- In the **Valid object type** field, click the ellipsis button **(...)** to specify the object type or types that the business rule is valid for.

  You can select **None**, **All object types**, or you can **Specify** specific object types that you want the rule to be valid for.



Select Valid Object Types

○ None

○ All Object Types

● Specify

Run as Privileged

When you select **Run as privileged** the business rule can execute functions that the user is not authorized to execute.

To Add Actions or Conditions to a Business Rule

On the Operations tab on the Business Rule tab, you can see the conditions or actions that make up the business rule. This is also where you can modify conditions or actions or add new conditions or actions.

- To edit an existing condition or action, click the **Edit Operation** icon.

For a description of how to add actions and conditions and for a description of the available actions and conditions, see **Business Actions** or **Business Conditions**.

On the Dependencies tab it is possible to add dependencies to business rules in order to reference business rules to business libraries. This is done by creating an alias on a business rule and linking the alias to a business library that contains JavaScript. For more, see **Using JavaScript Libraries in Business Rules**.

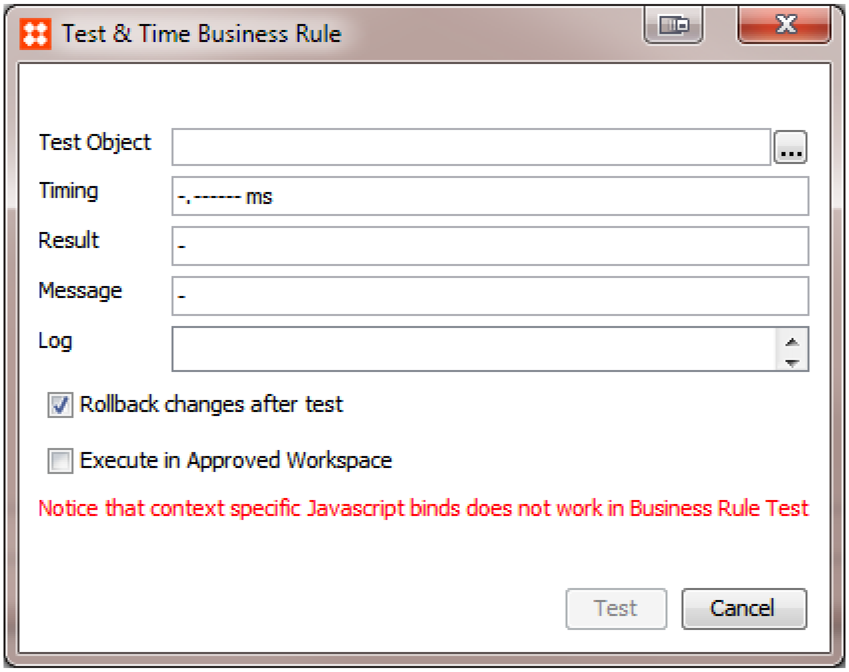
To Add Preconditions to a Business Rule

On the **Applies if** tab, you specify any preconditions to be evaluated before the business rule is applied. You do this by adding preconditions. Preconditions are defined using conditions.

Testing Business Rules

When you have created a business rule, you can test it before you start using it.

1. In **System Setup**, expand **Business Rules**, and then select the business condition or action that you want to test.
2. Right-click the condition or action, and then choose **Test Business Rule**.
3. In the **Test Object** field, click the ellipsis button **(…)**, and then select the object that you want to test the business rule on.
4. If you are testing an **Action**, check **Rollback changes after test** if you want the changes caused by the action to be rolled back after the action has been executed.
5. By default **Execute in Approved Workspace** is unchecked and the test is performed in the **Main Workspace**. Check **Execute in Approved Workspace** if you want the action to be performed on the selected object in the **Approved Workspace**.
6. Click **Test**.



Test Object

Timing: -,------ ms

Result: -

Message: -

Log

☑ Rollback changes after test

□ Execute in Approved Workspace

Notice that context specific Javascript binds does not work in Business Rule Test

- If you are testing an action, the result displays either **executed without errors** or **failed**. If the action fails, an error message is displayed.
- If the business rule is a condition, the result displays either true, false or failed. If a condition evaluates to false, the reason is displayed. If a condition fails, an error message is displayed.

Important: Context specific JavaScript binds do not work when you test business rules. Therefore, you cannot test JavaScript that uses the import change info or approve context binds. Similarly one cannot test business actions where the Execute JavaScript function is used to create parametrized business rules.

Monitoring Business Rules

When you use the **BusinessRule.Warning.Threshold** configuration property, you can 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.

Business Actions

The STEP system provides a range of business actions. 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.

Important: Business actions can be used to modify data and/or take action, while business conditions should be used to validate data only (not to modify data). For more information, see **Business Conditions**.

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

Menu	Action	Description
Attribute values	**Merge Attribute Values**	Merges the value of one attribute into another attribute. For more information, see **Business Action: Merge Attribute Values**.
Data quality	**Generate Match Codes**	Generates match code values as a part of a business action. For more information, see Business Action: Generate Match Codes.
	Standardize address	Overwrites the existing standardized address. For more information, see **Business Action: Standardize Address**.
	Match Duplicates (apply matching algorithm)	Matches a single object or a smaller set of objects for duplicates using a matching algorithm. This action is deprecated. We recommend that you do not use this action. Instead use the "Send Republish Event" to Event Processor configured to apply matching using your matching algorithm.
GDSN - For more information about	**Create CIC object and start workflow**	Creates a CIC entity and starts it in a workflow.

Menu	Action	Description
GDSN Provider, see **Using Business Rules with GDSN**.		
	Execute command	Sends out a message to a data pool.
	Set CIC status for publication	Sets the CIC status for a publication to a recipient.
	Update package hierarchy status	Sets the registration status of a package hierarchy link.
GDSN Receiver	**CICR exception handler**	For more information, see **About GDSN Receiver**.
	Invoke CIC message	
	Send RFCIN message	
	Set status or remove subscription	
	Set subscription status	
References and Links	**Add Attribute Link**	Adds a link to an attribute without additional configuration.
	Add Reference	Adds a reference of a specified type to a target. Choose the target manually, or use the **Function** editor. For more information, see **Business Action: Add Reference**.
	Add Referenced By	Adds a reference of a specified type from the specified source to the object that is executed as target.

Menu	Action	Description
		You can select the source object manually or use the **Function** editor.
	Automatic Classification	Classifies the selected objects automatically according to a set of rules. For more information, see **Business Action: Automatic Classification**.
	Remove Attribute Link	Makes it possible to remove a link to an attribute without additional configuration.
	Remove Reference	Removes references to a specific target or all references of a specific type. For more information, see **Business Action: Remove Reference**.
	Set Product to Classification Link Type	Assigns a new object type and classification link type to the selected product. For more information, see **Business Action: Set Product to Classification Link Type**.
Workflow	**Claim**	Claims the task of the current object in the specified workflow and state. For more information, see **Business Action: Claim**.
	Initiate Item in STEP Workflow	Initiates the object the action runs on in a specified workflow. For more information, see **Business Action: Initiate Item In STEP Workflow**.
	Trigger STEP Workflow Event	Triggers a specified workflow event. For more information, see Business Action: Trigger STEP Workflow Event.
Execute JavaScript		Enables you to use the methods available in the STEP Public Java API. For more information, see **Execute JavaScript**.
Overlap Analysis		Compares records in a pre-defined format with all record of the same object type by specifying a matching algorithm. For more information, see **Business Action: Overlap Analysis**.
Reference Other Business Action		Allows the node to reference other business actions without additional configuration.

Menu	Action	Description
Send Email		Sends email to specified recipients. For more information, see **Business Action: Send Email**.
Send Republish Event		Sends a republish event to an event queue You will have to specify which processor to send the republish event to. This can be an event processor, an outbound integration endpoint with an event queue, or a specific event queue.
Set Attribute Value		Sets a specified attribute values. For more information, see **Business Action: Set Attribute Value**.
Set Name		Sets the name of the object that the action runs on. Specify the name manually or use the Function editor.
Set Object Type		Changes the object type the action runs on. Select a specific object type or use the Function editor.
Set Workflow Variable		Enables assignment of workflow variables. For more information, see **Business Action: Set Workflow Variable**.

To add a business action, see **Specifying a Business Action**.

Specifying a Business Action

1. In **System Setup**, expand **Business Rules** and select the relevant business action.
2. On the **Business Rule tab** click **Edit Business Rule** that open the business rule editor.
3. In the **Valid Object Type** field, click the ellipsis button **(...)**, and then select the object types that you want to apply the action to.
4. In the lower left corner, click **Add New Business Action**.
5. Click the **Edit Operation** icon.
6. In the **Edit Operation** dialog, from the drop-down list, select the preferred action. The actions are described in the following table.
7. Click **Save** to save the changes.

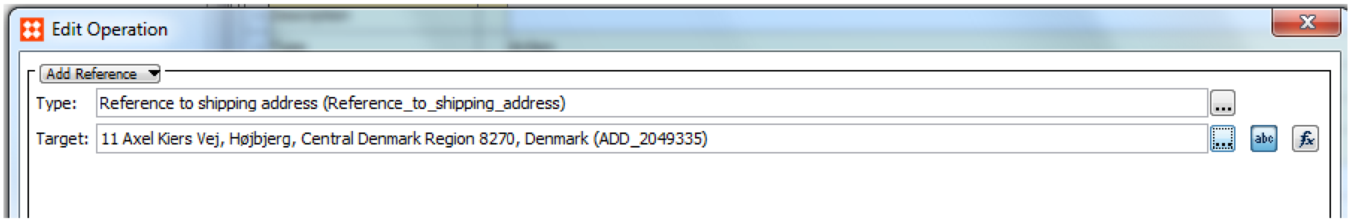
Important: If you specify more than one action, all actions are carried out when the overall business action is executed. If you specify more than one condition, all conditions have to be true for the overall condition to be true.

Business Action: Add Reference

Adds a reference of a specified type to a target.

1. In the **Type** field, click the ellipsis button **(...)**, and then select the preferred reference type.
2. In the **Target** field, click the ellipsis button **(...)**, and then select the relevant target. The target must be in accordance with the selected type.

   Alternatively, click the **Function** button and specify a function in the Function editor.



Type: Reference to shipping address (Reference_to_shipping_address)

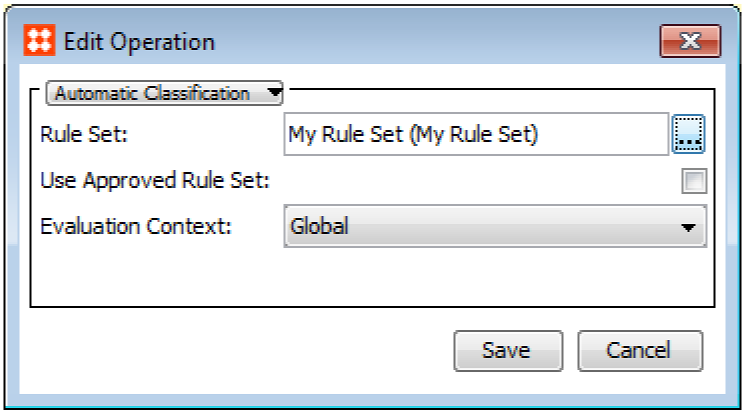
Target: 11 Axel Kiers Vej, Højbjerg, Central Denmark Region 8270, Denmark (ADD_2049335)

For more information about business rules actions, see **Business Actions**.

Business Action: Automatic Classification

Classifies the selected objects automatically according to a set of rules. When the action has been set up, the objects that the action is applied to are automatically classified on approval.

1. In **Rule Set**, click the ellipsis button **(...)**, and then select the **rule set** you want to use.
2. In **Type of new Reference or Links**, select the **link type** you want to use when linking products into classifications.
3. In Evaluation Context, select the **context** that you want the rule conditions to be evaluated in.



Rule Set: My Rule Set (My Rule Set)

Use Approved Rule Set: □

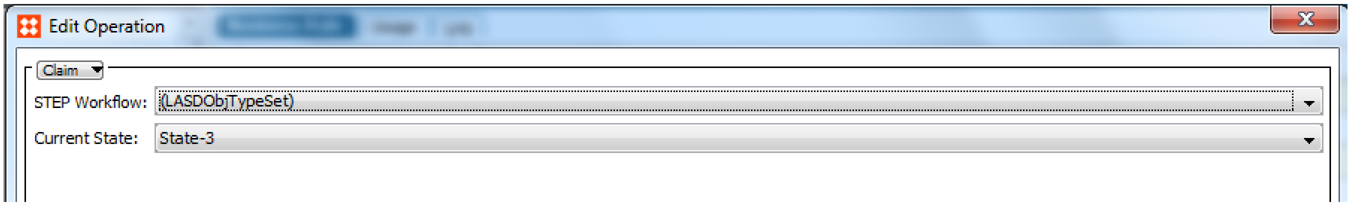
Evaluation Context: Global

For more information about automatic classification, see the Auto Classification documentation.

Business Action: Claim

Claims the task of the current object. If a workflow task has been assigned to a specific user, the Claim action can be used to reassign the task back to the group. This happens when the valid object or objects enter the specified state.

- From the STEP Workflow list, select the relevant workflow.
- From the Current State list, select the relevant state.



STEP Workflow: (LASDObjTypeSet)

Current State: State-3

Execute JavaScript

In addition to the standard business rule operations, more complex functions can be carried out using JavaScript and the Scripting API.

You can write JavaScript for your business conditions using the **Evaluate JavaScript** operation and for your business actions using the **Execute JavaScript** operation. Both operations include many JavaScript Binds, which provide access to the STEP data being worked on. For more information, see **JavaScript Binds**.

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

For more information about business rule actions, see **Business Actions**.

For more information about business rule conditions, see **Business Conditions**.

Documentation for the Scripting API is available on the STEP API Documentation page ([system]/sdk).

Example JavaScript to Set an Attribute Value

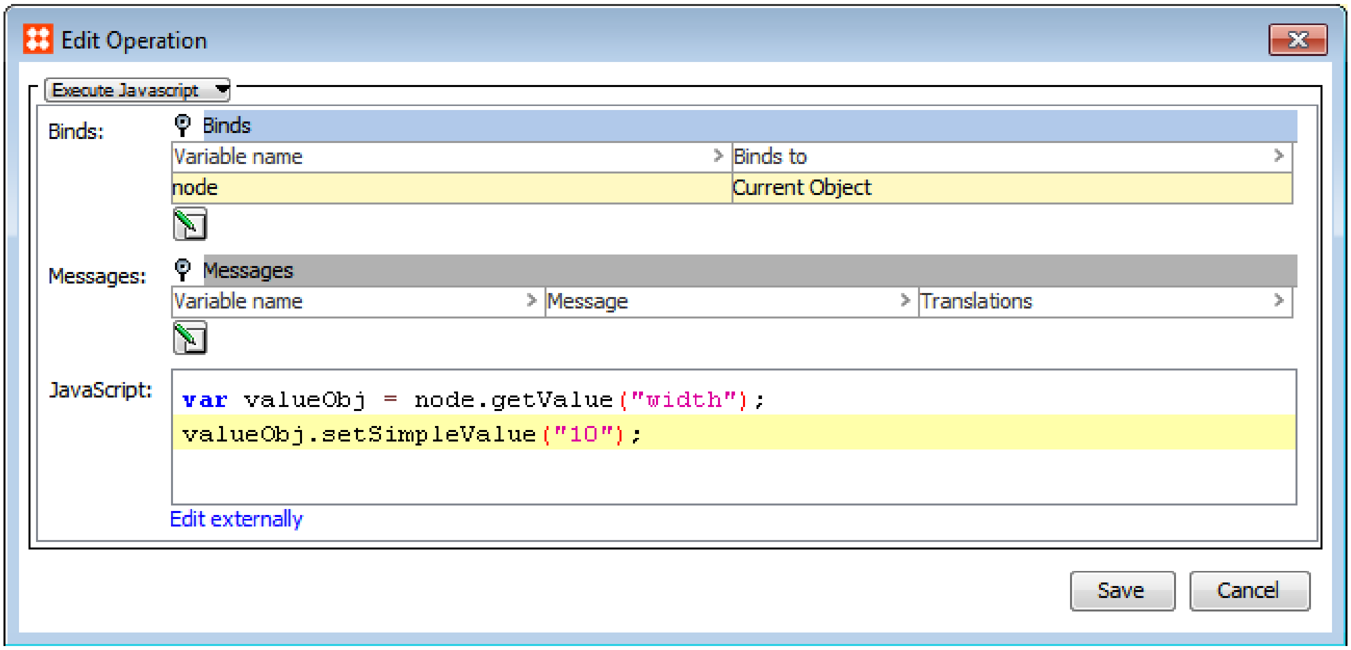
As a simple example, you could use the following JavaScript to set an attribute value.

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

```
var valueObj = node.getValue("Attribute ID");

valueObj.setSimpleValue("Value for the attribute");
```

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

In the following example, "width" is the attribute ID and "10" is the value the attribute will receive.



Binds

Variable name	Binds to
node	Current Object

Messages

Variable name	Message	Translations

```
var valueObj = node.getValue("width");
valueObj.setSimpleValue("10");
```

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

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

Handling JavaScript Exception Errors

Using JavaScript for business rules should always include a mechanism for handling exception errors. If not coded correctly, additional errors can be generated. The following recommendations should be followed to produce accurate errors that can be addressed.

Rethrow Exceptions

When using try-catch for handing exception errors, errors are logged to the step.0.log. While this is effective for documenting what happened, it does not give the user a chance to address the error from STEP, and the STEP framework is not informed of the error and therefore cannot handle other scenarios like Optimistic Lock Exceptions. To avoid this, it is important to include a 'throw' when catching errors.

Wrong	Right

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

Wrong	Right

}	`throw(e);`
	}

Ignore Expected Exceptions and Rethrow Others

Although not recommended setup, if you use try-catch for handling **expected** exceptions, it is important that your catch logic only ignores the specific intended exceptions, and that all other exceptions are rethrown. Again, this allows the framework to process unexpected exceptions correctly and avoids potential data inconsistency.

The code snippet below shows how it is possible to handle expected and unexpected exceptions differently. Notice the necessary Rhino specific 'e.javaException' notation.

```
try {
   currentObject.createReference(targetProduct, accRefType.getID());
} catch (e) {
   if (e.javaException instanceof
com.stibo.core.domain.reference.TargetAlreadyReferencedException) {
      logger.warning("Reference already exists");
   } else {
      throw (e);
   }
}
```

Translatable 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 Business Conditions used to indicate whether the condition is true or false. When boolean true is returned, the condition evaluates to true. Any other return value will make the condition evaluate to false.

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

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

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

The Return statement is not available in a Business Action.

Throw Statement

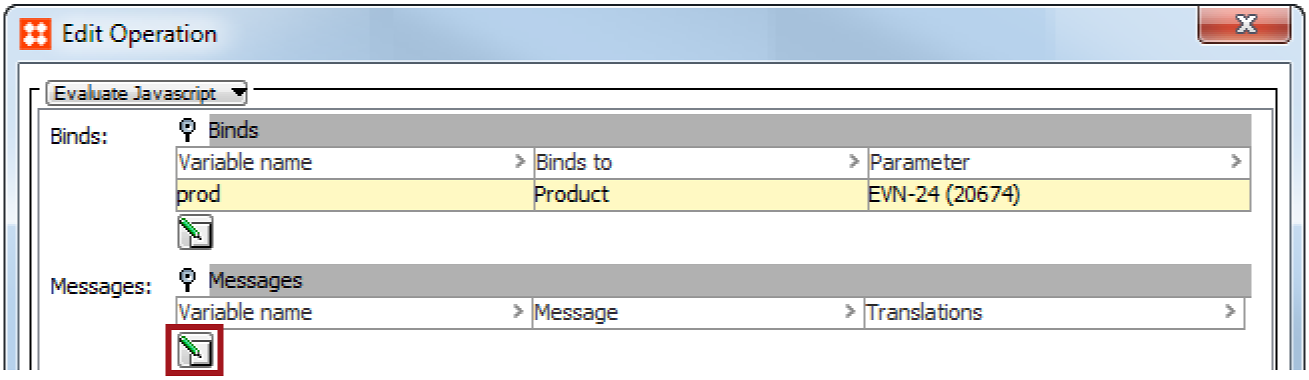
It is possible to deliberately throw exceptions from both Business Actions and Conditions if error states are encountered. The exception message can be a translatable message object. For more information on adding messages, see **Adding a Translatable Business Rule Message** in the **Business Rules**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

Adding a Translatable Business Rule Message

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

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

1. Open the Edit Operation dialog for an existing JavaScript Condition or Action.
2. On the Messages field click the Edit button.

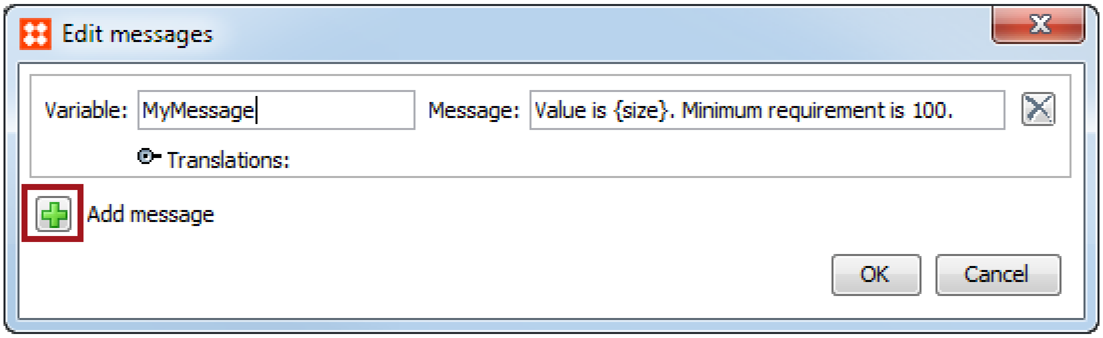


Variable name	Binds to	Parameter
prod	Product	EVN-24 (20674)

Variable name	Message	Translations

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.

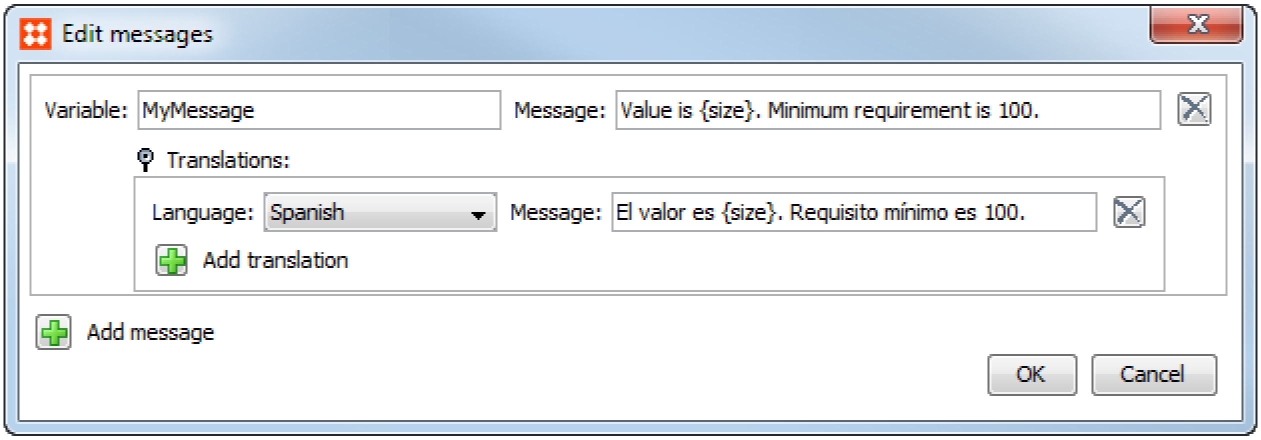


Variable: MyMessage

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.

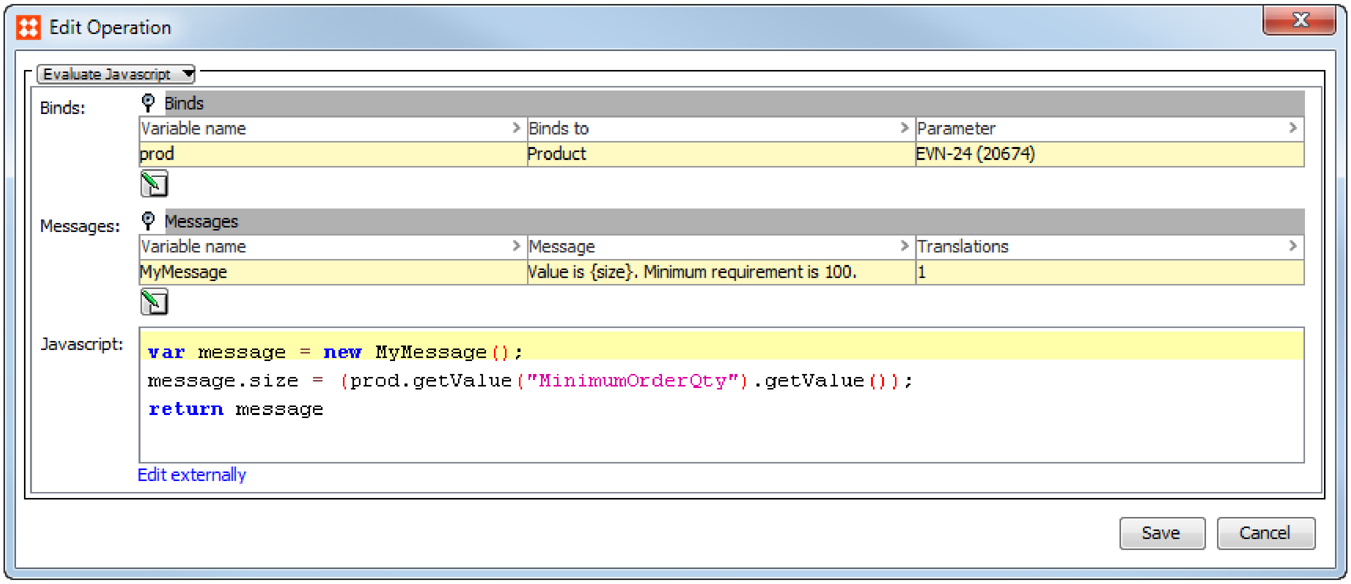
4. Add any translations needed. Do not translate variable tags.



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

Language: Spanish Message: El valor es {size}. Requisito mínimo es 100.

5. Click OK.
6. Add JavaScript to display the message.



Binds:

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

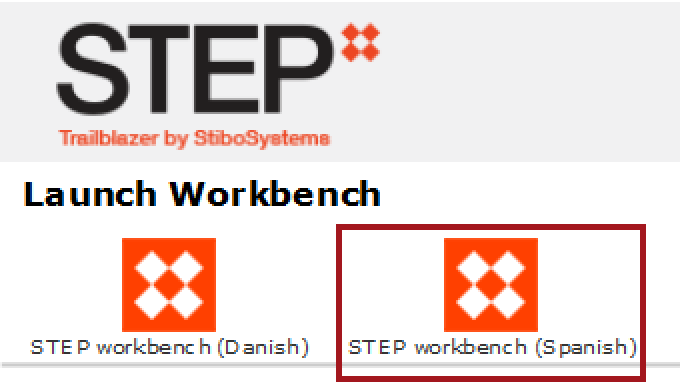
Messages:

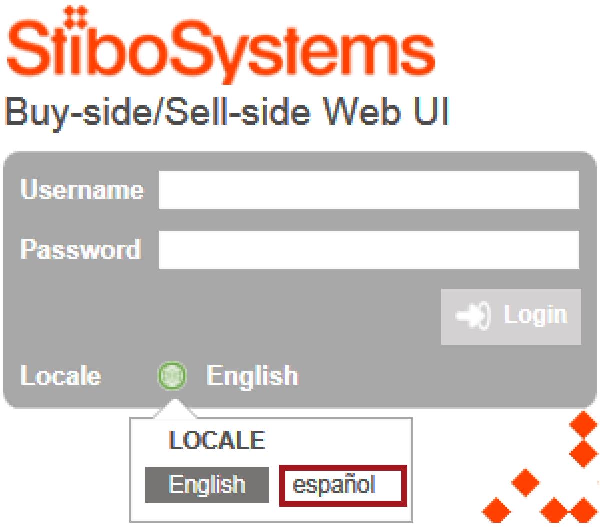
Variable name	Message	Translations
MyMessage	Value is {size}. Minimum requirement is 100.	1

Javascript:

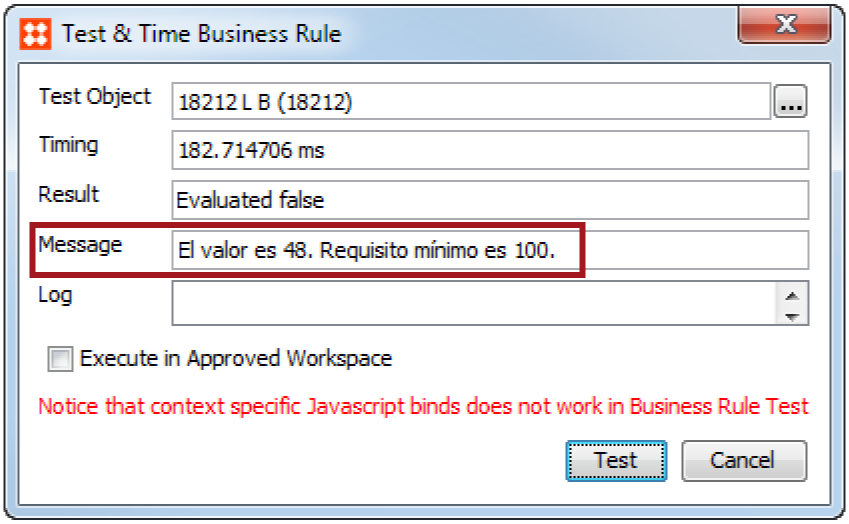
```
var message = new MyMessage();
message.size = (prod.getValue("MinimumOrderQty").getValue());
return message
```

7. If you added a translation, log into the locale that matches the language for the message using the STEP Workbench WebStart or the Web UI. For our example, we select the Spanish locale.





8. Test the business rule or run a workflow that uses it to display the translated message.



Test & Time Business Rule

Test Object	18212 L B (18212)
Timing	182.714706 ms
Result	Evaluated false
Message	El valor es 48. Requisito mínimo es 100.
Log	

☐ Execute in Approved Workspace

Notice that context specific Javascript binds does not work in Business Rule Test

Using JavaScript Libraries in Business Rules

JavaScript business libraries are useful when you want to use the same JavaScript functions in several different business actions, business conditions, or other business libraries.

Before you can create a JavaScript library, verify that a business library setup group exists that can hold the library object. If this is not the case, you have to create a setup group.

Set up a Business Library

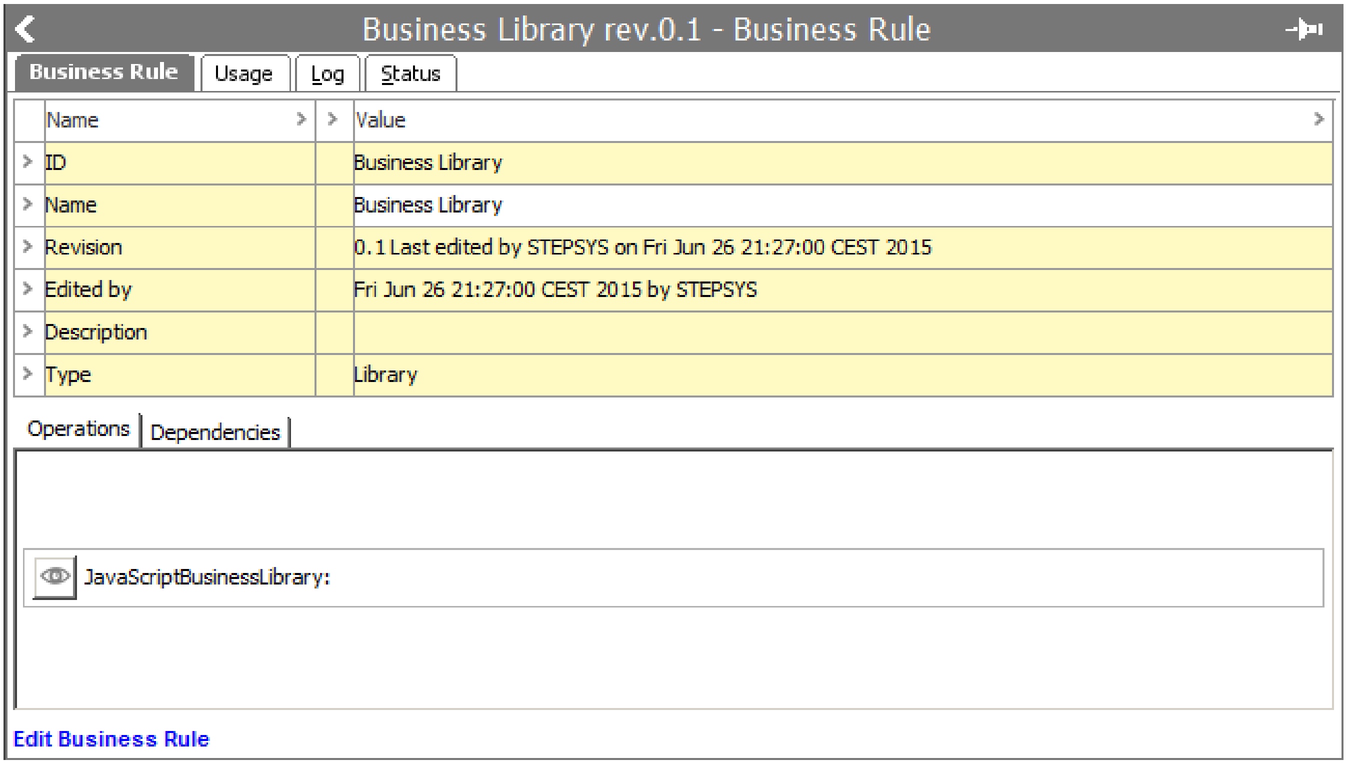
1. In **System Setup**, expand **Object Types & Structures**.
2. Expand**Setup Group type root**, right-click the **Business Rules** node where you want to add the library, and then select **New Object Type**.

3. Enter an **ID** and a **Name** and click **Create**.
4. In **Object Types & Structures**, expand **Basic Object Types**, and then select **Business Library Type**.
5. On the **References** tab, click **Add Parent**.
6. In the **Select New Parent** dialog, select the setup group you just created, and then click **Select** to make it a valid parent.
7. From the **Maintain** menu, point to **Insert**, and then choose **Setup Group Root**.
8. Select the object type you just created, enter an **ID** and a **Name**, and then click **Create**. An instance of the object type is created in **System Setup**.

You can now create a JavaScript library

Create a JavaScript Business Library

1. In **System Setup**, right-click the relevant **Business Rules** node, and then choose **New Business Library**.
2. Enter an **ID** and a **Name** for the library and click the Create button.
3. Click the **Edit Business Rule** link to open the Business Rule Editor.
4. Click the icon next to **JavaScriptBusinessLibrary**, and then write the relevant functions.



Business Library rev.0.1 - Business Rule

Name	Value
ID	Business Library
Name	Business Library
Revision	0.1 Last edited by STEPSYS on Fri Jun 26 21:27:00 CEST 2015
Edited by	Fri Jun 26 21:27:00 CEST 2015 by STEPSYS
Description	
Type	Library

JavaScriptBusinessLibrary:

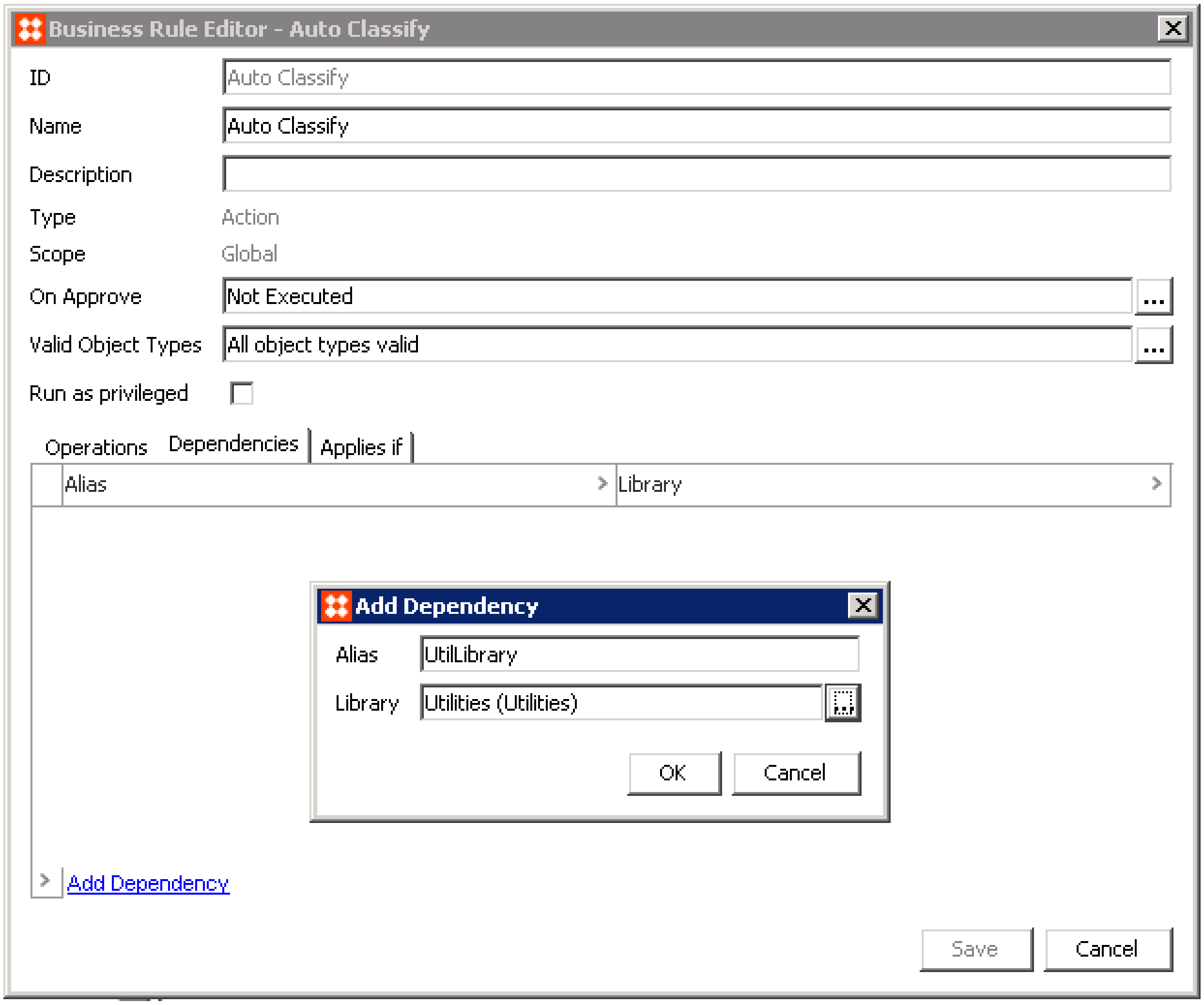
Edit Business Rule

JavaScript libraries do not allow you to bind in special variables. Variables must be supplied as arguments when functions are called in libraries. However, you can bind in special variables writing JavaScript in the business action and condition editors.

Set up a Reference to a JavaScript Business Library

You must set up a reference to the library from each business action, business condition, or library where you want to be able to use the library.

1. In **System Setup**, locate and select the relevant business action, a business condition, or a business library.
2. On the **Dependencies** tab, click **Add Dependency**, and then enter an **Alias** and select the relevant **Library** from the list.



Business Rule Editor - Auto Classify

ID: Auto Classify
Name: Auto Classify
Description:
Type: Action
Scope: Global
On Approve: Not Executed
Valid Object Types: All object types valid
Run as privileged: □

Alias	Library

Add Dependency

Alias: UtilLibrary
Library: Utilities (Utilities)

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
}
```

JavaScript Binds

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

Grouping	Bind
Approve Context Bind	• Approve Context
Attribute Related Binds	• Attribute • Attribute Group • Attribute Validated Parameters • Attribute Value • List of Values • Unit • Unit Group
Current Object Bind	• Current Object
e-Signature Related	• Basic e-Signature
Event Binds	• Current Event Batch • Current Event Queue • Current Event Type • Event Queue • Event Type
Gateway Integration Endpoint Bind	• Gateway Integration Endpoint
GDSN Binds	• GDSN Product • GDSN Provider Datapool • GDSN Publisher Data Map • GDSN Publisher Product Validation • GDSN Receiver Data Map • GDSN Recipient • GDSN Target Market • GDSN Validation Logger

Grouping	Bind
Mongo DB Binds	• JSON Document • MongoDB Context
Object Aspects Binds	• ID • Name
Other Binds	• Conditionally Invalid Values • Import Change Info • Logger • Lookup Table Home • Mailer
Reference Type Bind	• Reference Type
Secondary Object Bind	• Secondary Object
STEP Manager Bind	• STEP Manager
Tree Object Binds	• Asset • Classification • Entity • Product
User Binds	• User • User Group
Workflow Binds	• Current Transition • Current Workflow • Workflow Parameters • Workflow State

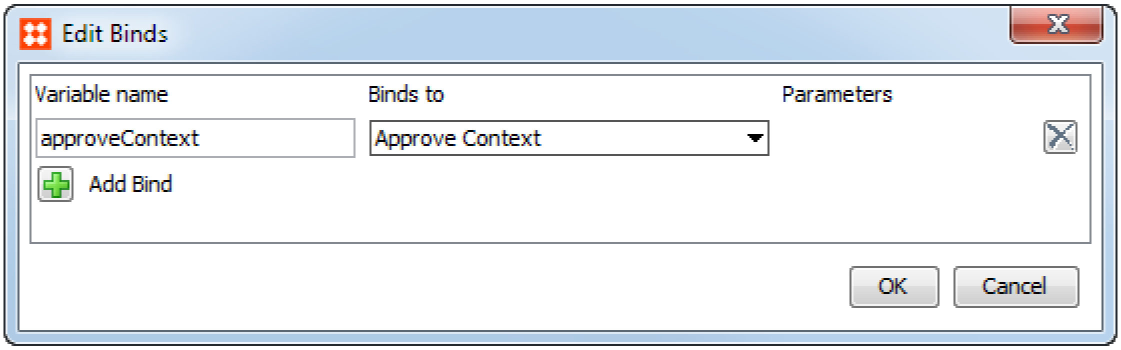
The Translatable message option is available in both JavaScript Business Actions and Conditions. For more information, see the **Translatable Messages for JavaScript Business Rules** section of the **Business Rules** documentation.

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

Approve Context Bind

Business rules tested / executed during approval have access to the Approve Context binding. For conditions hooked into the approval process and business rules hooked into event based outbound integration endpoints, this bind gives access to the methods of the ApprovePlugin.ApproveContext interface.

The Approve Context bind is applicable before approval or on approval. Its use varies between Business Actions and Business Conditions.



The interface has the following methods:

```
getApprovedManager()
getApprovedNode()
getMainManager()
getMainNode()
getPartObjects()
```

Using these methods, you can compare data in Main and Approved workspaces and also access the set of part objects to be synchronized.

Business Condition

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

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

In this example, Approve Context is bound to the JavaScript variable approveContext.

```
if(approveContext.getApprovedManager().getProductHome().getProductByID
(node.getID())) {
        return true;}
else {return "Node not in Approved Workspace";}
```

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

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

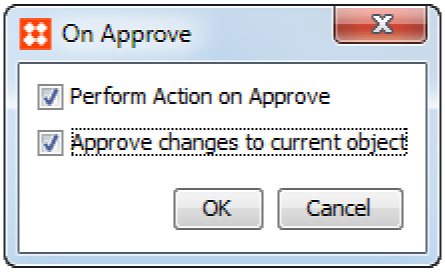
Business Action

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

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

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

Note that actions executed On Approval can have changes synchronized automatically by setting the Business Rule Editor 'On Approve' field to select the 'Approve changes to current object' checkbox for the action.

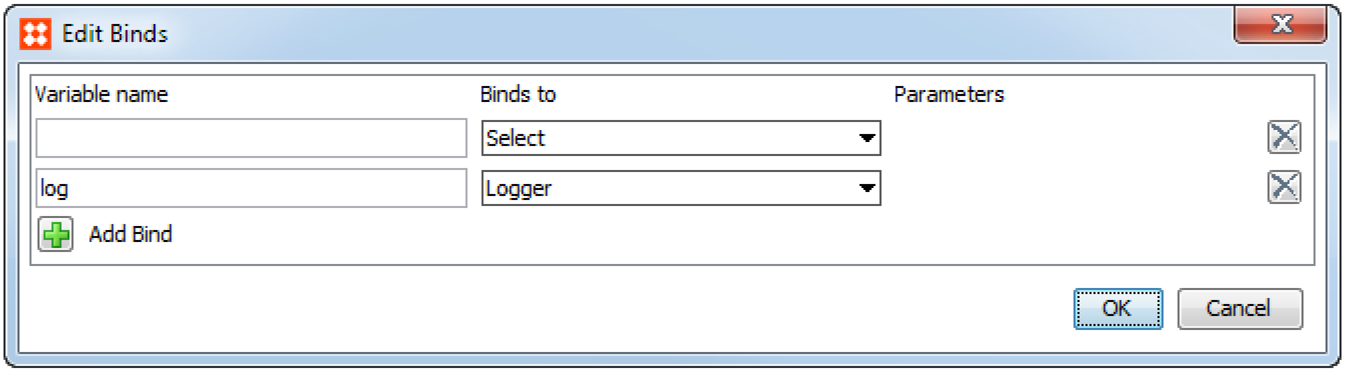


☑ Perform Action on Approve

☑ Approve changes to current object

Attribute Related Binds

The following binds allow access to the selected object or value. In the following table, coding samples expect a bind for Logger using the variable 'log' and a second bind as described in the table:



Variable name	Binds to	Parameters
	Select	
log	Logger	

Bind Name	Description

Attribute	Binds the selected attribute to the variable.
	In this example, the Variable Name is 'eanAttribute', the Binds to field is Attribute, and the Parameters field displays the EAN attribute. The following JavaScript returns the ID of the EAN attribute as an Info log message:
	`var bindDemo = eanAttribute.getID();`
	`log.info("Attribute ID is " + bindDemo);`

Attribute Group	Binds the selected attribute group to the variable.
	In this example, the Variable Name is 'displayAttrGroup', the Binds to field is Attribute Group, and the Parameters field shows the Display attribute. The following JavaScript returns the ID of the Display attribute group as an Info log message:
	`var bindDemo = displayAttrGroup.getID();`
	`log.info("Attribute Group ID is " + bindDemo);`

Attribute Validated Parameters	For more information, see the **Parameterized Business Actions in Web UI** section of the Web UI Getting Started documentation.
Attribute Value	Binds the value of the selected attribute directly as a String variable. Primarily used for Web UI live validation of Conditions and when defining Match Codes. **Note:** This variable also works in the Web UI. In this example, the Variable Name is 'primaryColorAttrValue', the Binds to field is Attribute Value, and the Parameters field displays the Primary Color attribute. The following JavaScript returns a message when the value of the Primary Color attribute does not match the expected value of 'Gray': `return primaryColorAttrValue == "Gray" ? true : "Primary Color is " + primaryColorAttrValue + " but it should be Gray.";`
List of Values	Binds the selected LOV (called 'Domain' in the workbench) to the action or condition variable.
Pair of Attribute	Binds the selected attribute to the variable. In this example, the Variable Name is 'email', the Binds to field is Pair of Attribute values, and the

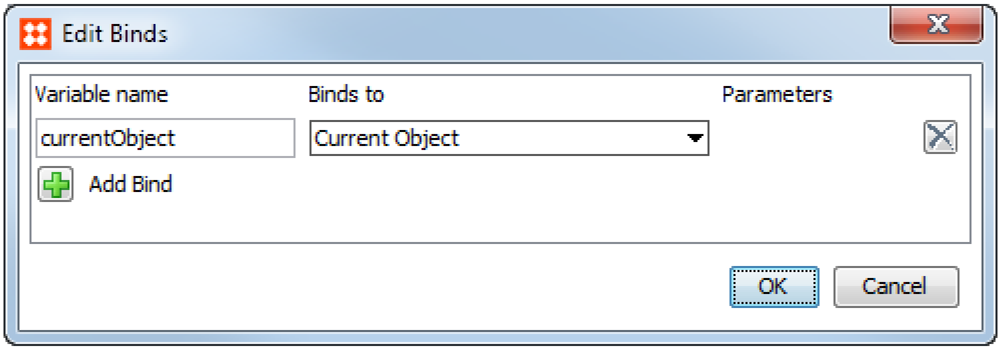
Bind Name	Description

Values	Parameters field displays the Email attribute used on a Data Container. The following JavaScript - used in a JavaScript Business Condition on the Data Containers Trusted Source or Most Recent Survivorship Rules - compares the attribute values on the source object and golden record:
	`var s1 = email.getValue1();`
	`var s2 = email.getValue2();`
	`if(s1==s2){`
	`return true;`
	`}`
	`else{`
	`return "Error"`
	`}`

Unit	Binds the selected unit to the action or condition variable.
Unit Group	Binds the selected unit group to the action or condition variable.

Current Object Bind

The Current Object bind gives access to the STEP object that the Business Rule is being evaluated or executed against. The following sample code expects a bind to **currentObject**:



Edit Binds

Variable name: currentObject

Binds to: Current Object

Parameters

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

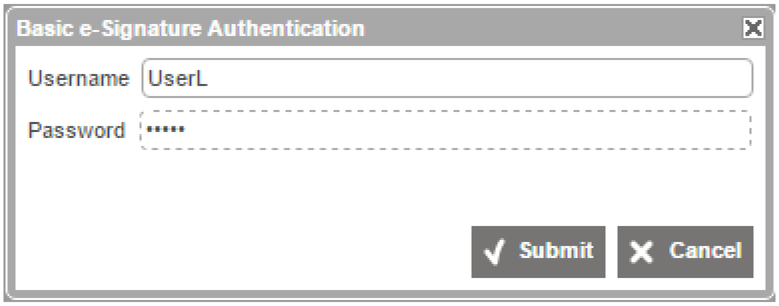
The following code gets the value of the 'Description' attribute on the current object and stores it in the variable 'Description'.

```
var description = currentObject.getValue("Description").getSimpleValue();
```

About e-Signature

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

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



Basic e-Signature Authentication

Username: UserL

Password: •••••

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

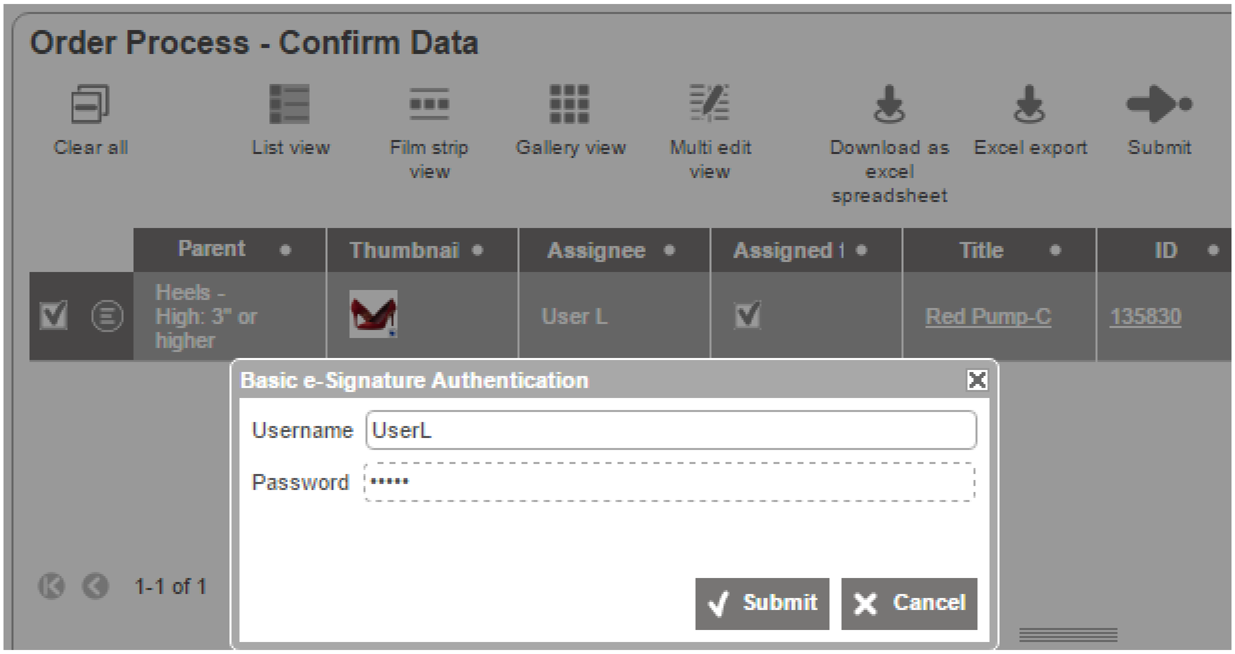
For more on how to configure e-Signature, see **Applying e-Signature to a Transition in a Workflow** in the **Business Rules** documentation.

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

Using e-Signature in Web UI

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

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



Order Process - Confirm Data

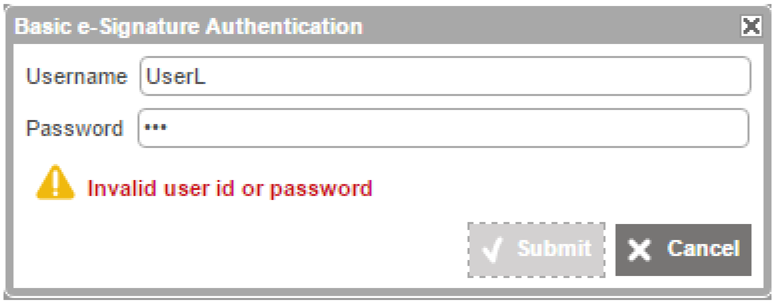
	Parent	Thumbnai	Assignee	Assigned t	Title	ID
☑	Heels - High: 3" or higher		User L	☑	Red Pump-C	135830

Basic e-Signature Authentication

Username: UserL

Password: •••••

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



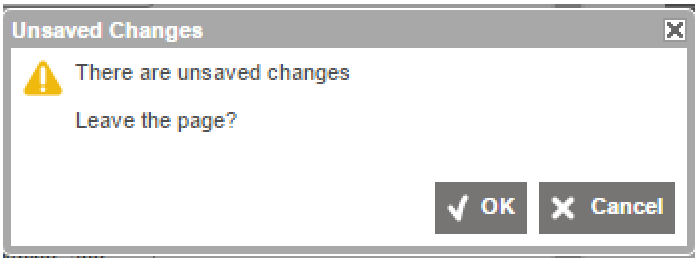
Basic e-Signature Authentication

Username: UserL

Password: •••

Invalid user id or password

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



Unsaved Changes

There are unsaved changes

Leave the page?

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

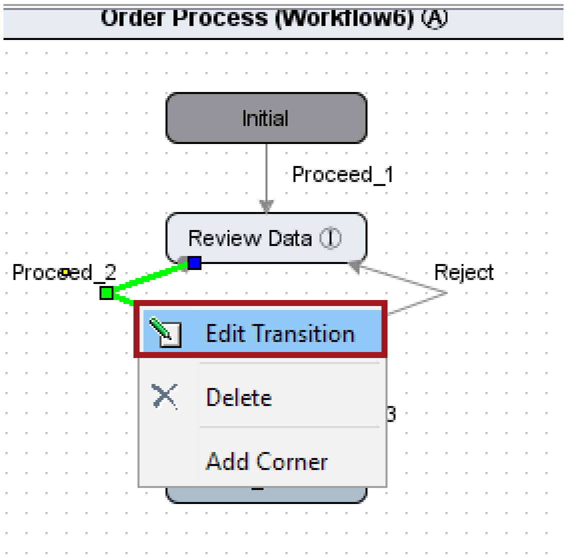
Applying e-Signature to a Transition in a Workflow

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

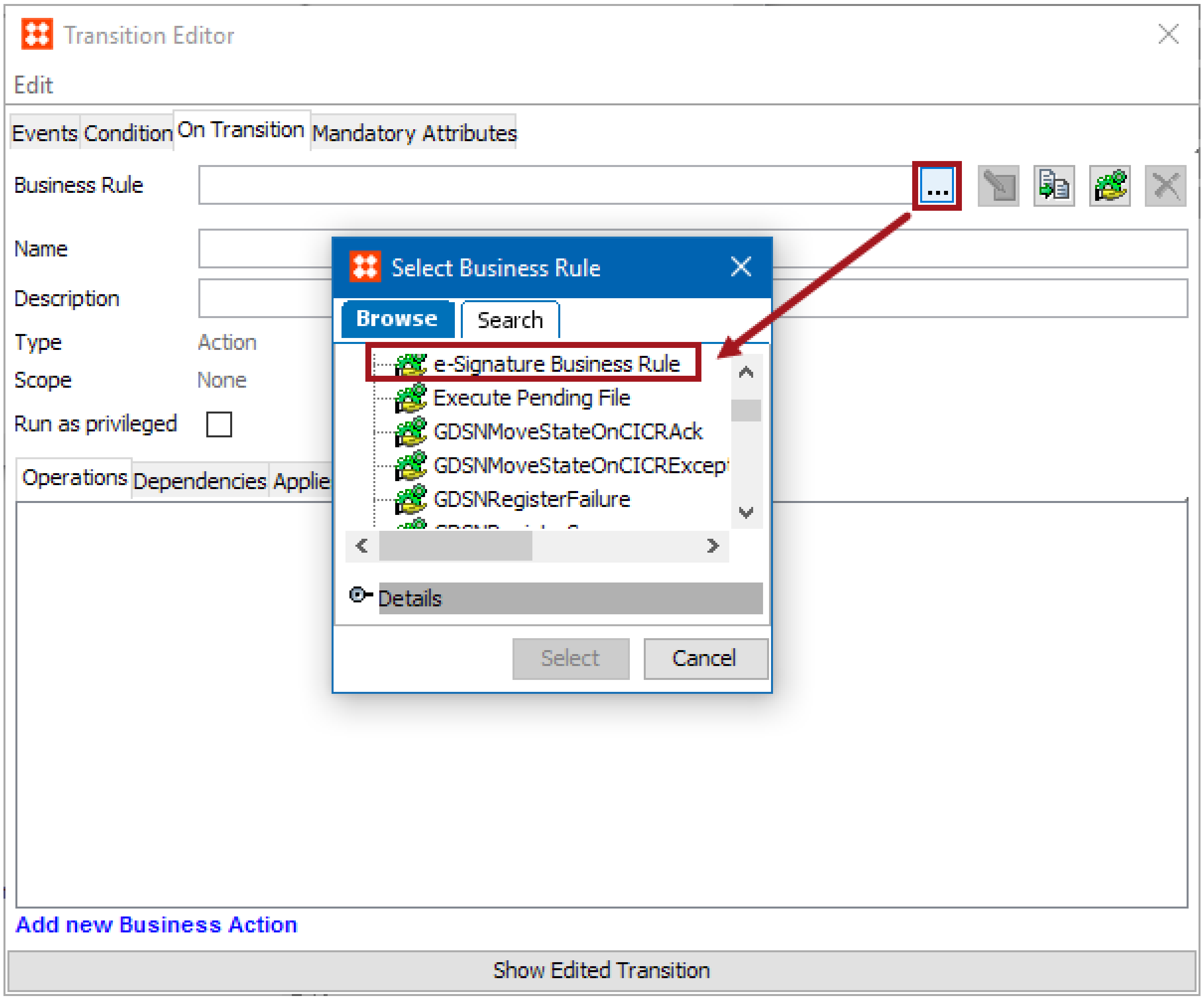
Note: Users attempting to call a business rule using e-Signature in the workbench will receive an error.

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

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



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



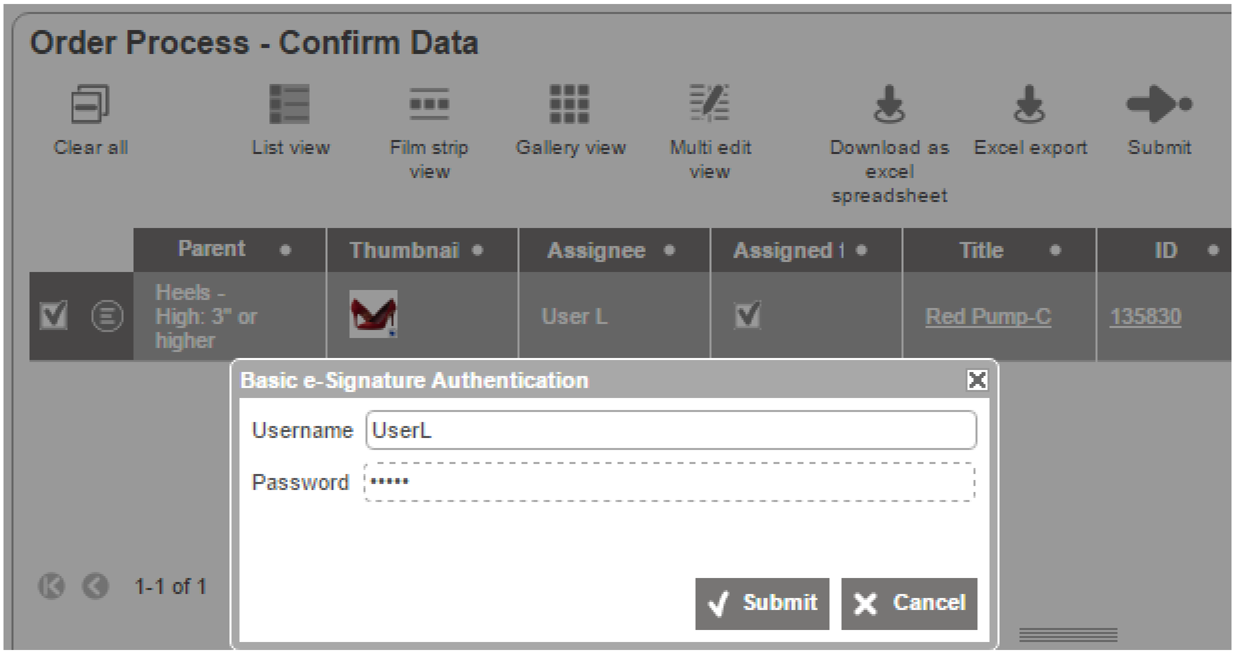
Type Action

Scope None

Run as privileged □

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

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



	Parent	Thumbnai	Assignee	Assigned t	Title	ID
☑	Heels - High: 3" or higher		User L	☑	Red Pump-C	135830

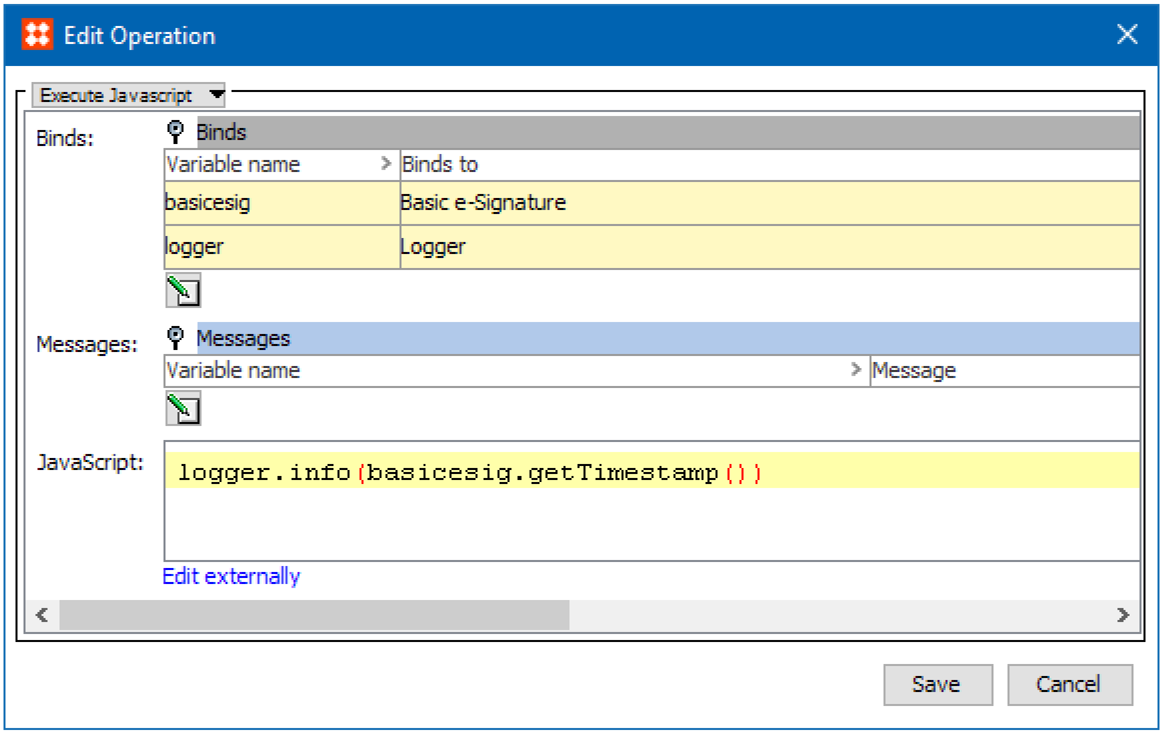
Username	UserL
Password	•••••

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

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

e-Signature Bind

The **Basic e-Signature** bind is used when re-authentication needs to occur.



Binds:

Variable name	Binds to
basicesig	Basic e-Signature
logger	Logger

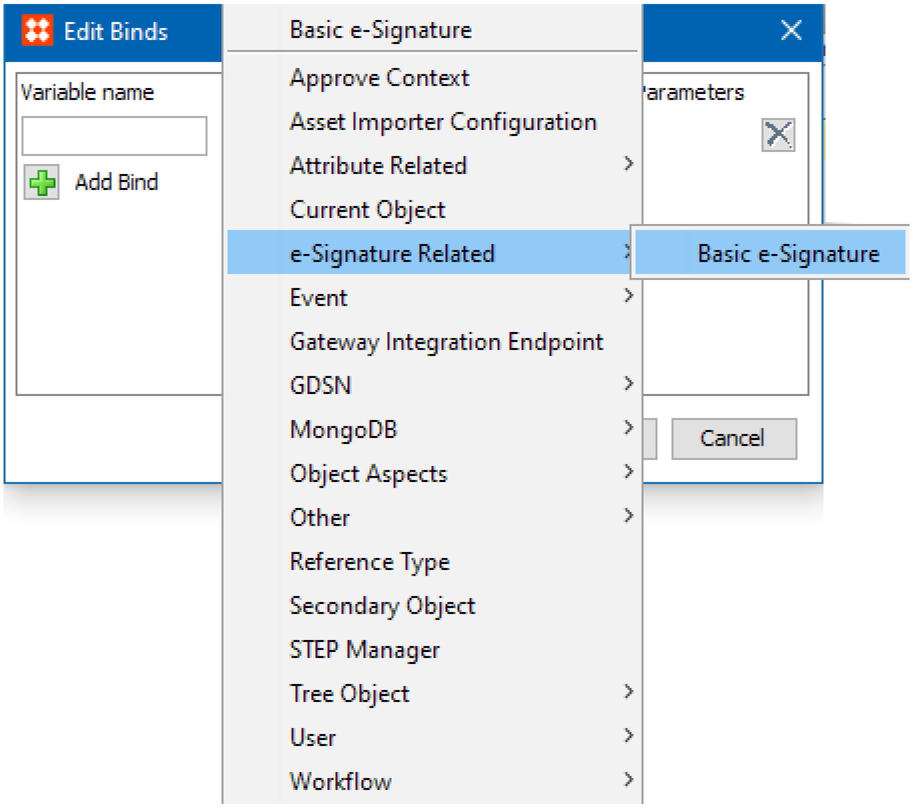
Messages:

Variable name	Message

JavaScript:

```
logger.info(basicesig.getTimestamp())
```

To add the e-Signature bind for a business action, choose Add new Business Action > Merge into > Execute JavaScript > Edit Bind > Add Bind > e-Signature Related > Basic e-Signature.



Configure any other binds needed, and then click OK.

Event Binds

Business Actions and Business Conditions can use any of the event binds to access the selected event object.

Bind Name	Description

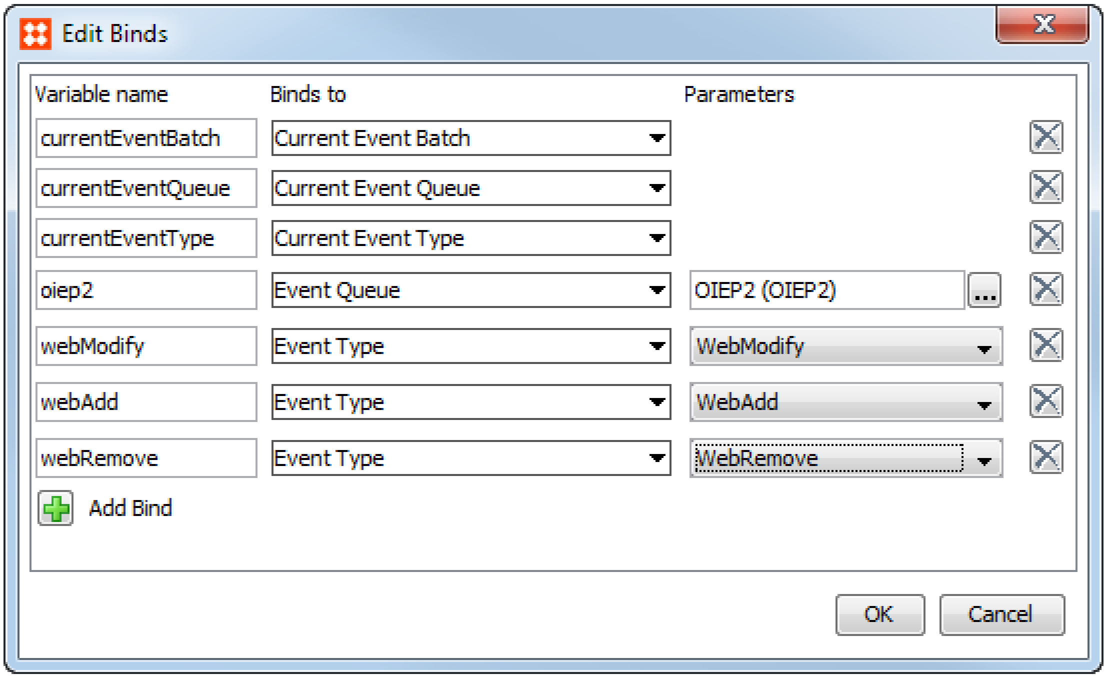
Current Event Batch	This bind is only available in actions used as an OIEP Preprocessor and gives access to the batch of events currently being handled. This makes it possible to examine the events in the batch and to add new objects to the batch.
	In this example, Current Event Batch is bound to the variable 'currentEventBatch'. For each event in the batch, the code accessed the object (node) for which the event was generated, checks if it is a product, and if so, adds the parent object to the batch.
	`eventsList = currentEventBatch.getEvents();`
	`for (var i = 0; i < eventsList.size(); i++) {`
	`var node = eventsList.get(i).getNode();`
	`if (node instanceof com.stibo.core.domain.Product) {`
	`currentEventBatch.addAdditionalNode(node.getParent());`
	`}`
	`}`

Current Event Queue	Bind available for conditions and actions used as OIEP event filters / event generators. This bind gives access to the event queue tied to the OIEP and from an action makes it possible to, for example, add derived events to the queue.
	In this example, Current Event Queue is bound to 'currentEventQueue', Current Object is bound to the variable 'currentObject', and a derived event is bound to the variable 'myEvent' via the Event Type bind. The code queues an event for the parent of current object if current object is a product.
	`if (currentObject instanceof com.stibo.core.domain.Product) {`
	`currentEventQueue.queueDerivedEvent(myEvent, currentObject.getParent());`
	`}`

Current Event Type	Binds the current event type and can be hooked into an endpoint as filters / generators to examine the current event. This bind is available for conditions and actions used as OIEP event filters / event generators. Use this to examine the type of the current event.
	In this example, Current Event Type is bound to 'currentEventType' and the code checks whether the current event is a core 'modify' event.
	`if (currentEventType.equals(com.stibo.core.domain.eventqueue.BasicEventType.Modify)) {`
	`// Do something if core modify event`
	`}`

Bind Name	Description
Event Queue	Bind available from conditions and actions, although typically used in actions. Use the bind from an action to queue an event on any event queue in the system. In this example, a derived event is bound to the variable 'myEvent' via the Event Type bind, an event queue is bound to the variable 'someEventQueue' via the Event Queue bind, and Current Object is bound to 'currentObject'. The example places a 'myEvent' event on the queue for current object. `someEventQueue.queueDerivedEvent(myEvent, currentObject);`
Event Type	Bind available for actions and conditions, although typically used in actions, binds a derived event to a JavaScript variable. In this example, a derived event is bound to the variable 'myEvent' via the Event Type bind, an event queue is bound to the variable 'someEventQueue' via the Event Queue bind and Current Object is bound to 'currentObject'. The example places a 'myEvent' event on the queue for current object. `someEventQueue.queueDerivedEvent(myEvent, currentObject);`

The example code below expects these bindings:



Variable name	Binds to	Parameters
currentEventBatch	Current Event Batch	
currentEventQueue	Current Event Queue	
currentEventType	Current Event Type	
oiep2	Event Queue	OIEP2 (OIEP2)
webModify	Event Type	WebModify
webAdd	Event Type	WebAdd
webRemove	Event Type	WebRemove

Checking Current Event

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

Enum) or a derived event (DerivedEventType interface).

This code checks whether current event is a Core event:

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

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

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

This code checks whether current event is a Derived event:

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

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

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

Queuing a Derived Event

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

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

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

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

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

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

Gateway Integration Endpoint Bind

The Gateway Integration Endpoint bind is accessed from JavaScript in business rule conditions and actions. The bind can work with the REST methods included in the table below.

REST Method	Syntax Example

`delete`	`delete().path("products").pathQuery({`
	`workspace: "Approved", context: "GL"}).body("Test").invoke();`

REST Method	Syntax Example

`get`	`get().path("products").pathQuery({`
	`workspace: "Approved", context: "GL"}).invoke();`

`head`	`head().path("products").pathQuery({`
	`workspace: "Approved", context: "GL"}).invoke();`

`options`	`options().path("products").pathQuery({`
	`workspace: "Approved", context: "GL"}).invoke();`

`post`	`post().path("products").pathQuery({`
	`workspace: "Approved", context: "GL"}).invoke();`

`put`	`put().path("products").pathQuery({`
	`workspace: "Approved", context: "GL"}).body("Test").invoke();`

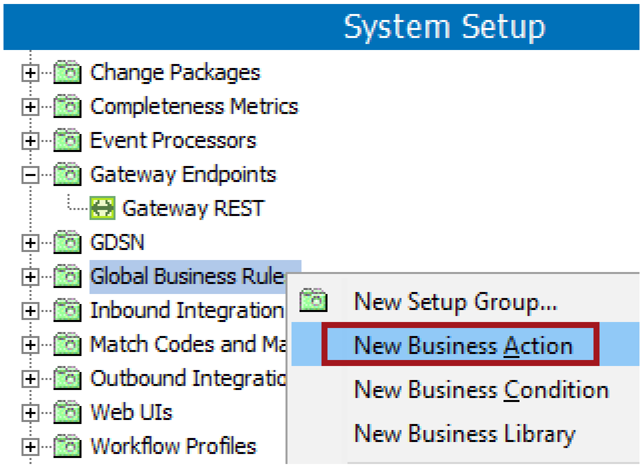
For more information, see the **Business Action Accesses Gateway Integration Endpoint** section.

For more information about the available methods, see the STEP API documentation.

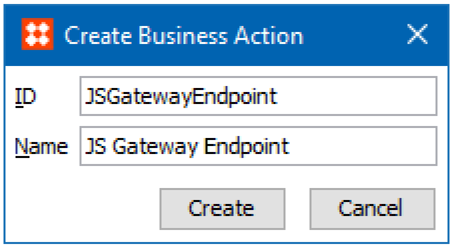
Business Action Accesses Gateway Integration Endpoint

Before a business action can access a gateway endpoint, use the following steps to first create the business action and then bind the gateway endpoint into the JavaScript code.

1. In System Setup, select the Business Rules setup group, and then right-click and select **New Business Action**.



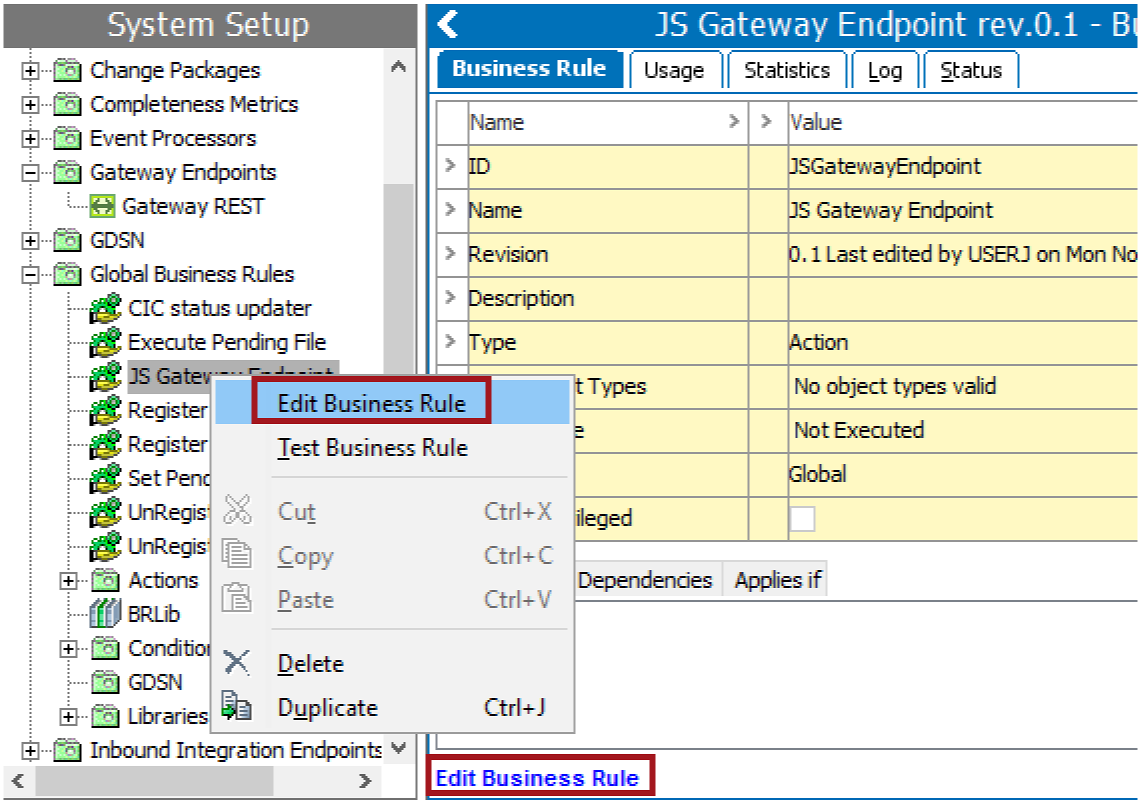
2. Add an **ID** and a **Name** for the new business action and click the **Create** button.



ID: JSGatewayEndpoint

Name: JS Gateway Endpoint

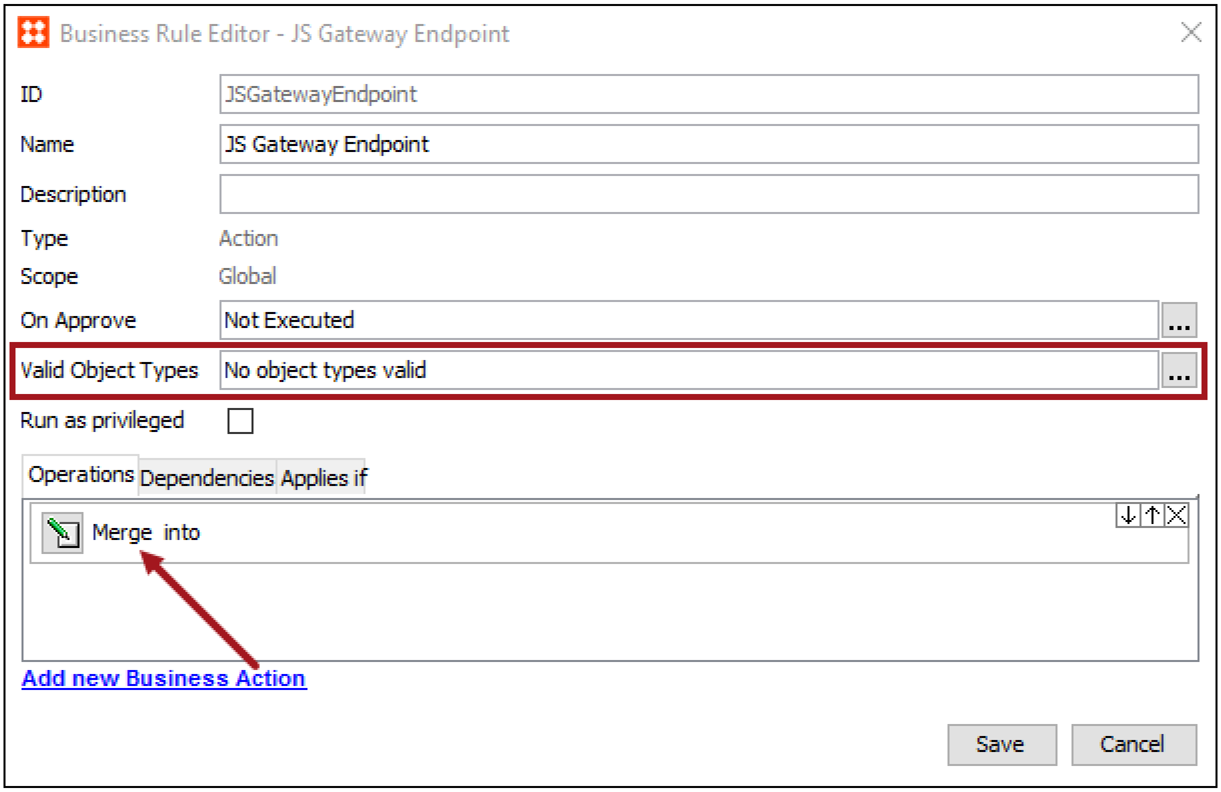
3. Right-click the new business action and choose **Edit Business Rule**, or click the Edit Business Rule link on the editor.



Name	Value
ID	JSGatewayEndpoint
Name	JS Gateway Endpoint
Revision	0.1 Last edited by USERJ on Mon No
Description	
Type	Action
t Types	No object types valid
e	Not Executed
	Global
ileged	☐

4. On the Business Rule Editor dialog:

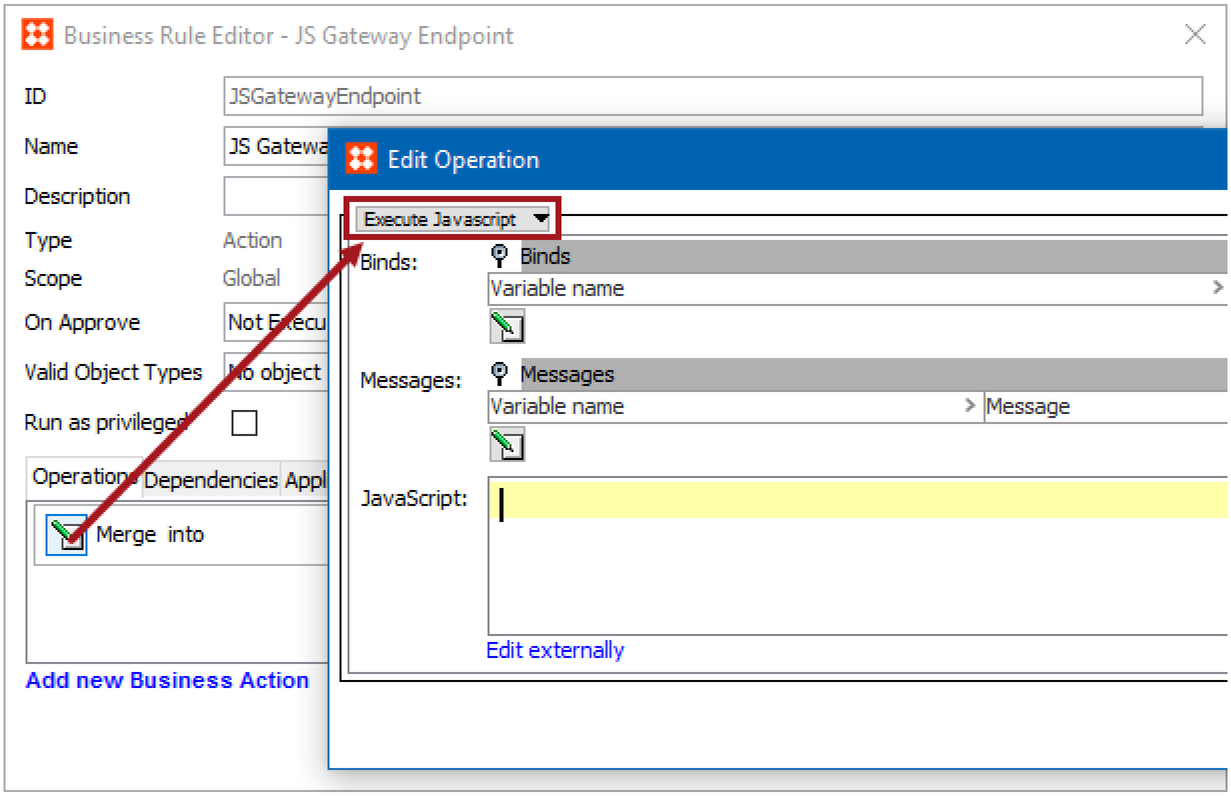
- For the **Valid Object Types** parameter, choose the object types where business action is allowed to be executed.
- Click the **Add New Business Action** link to add an operation.



Business Rule Editor - JS Gateway Endpoint

ID	JSGatewayEndpoint
Name	JS Gateway Endpoint
Description	
Type	Action
Scope	Global
On Approve	Not Executed
Valid Object Types	No object types valid
Run as privileged	□

5. Edit the new operation and select **Execute JavaScript** from the operation dropdown.

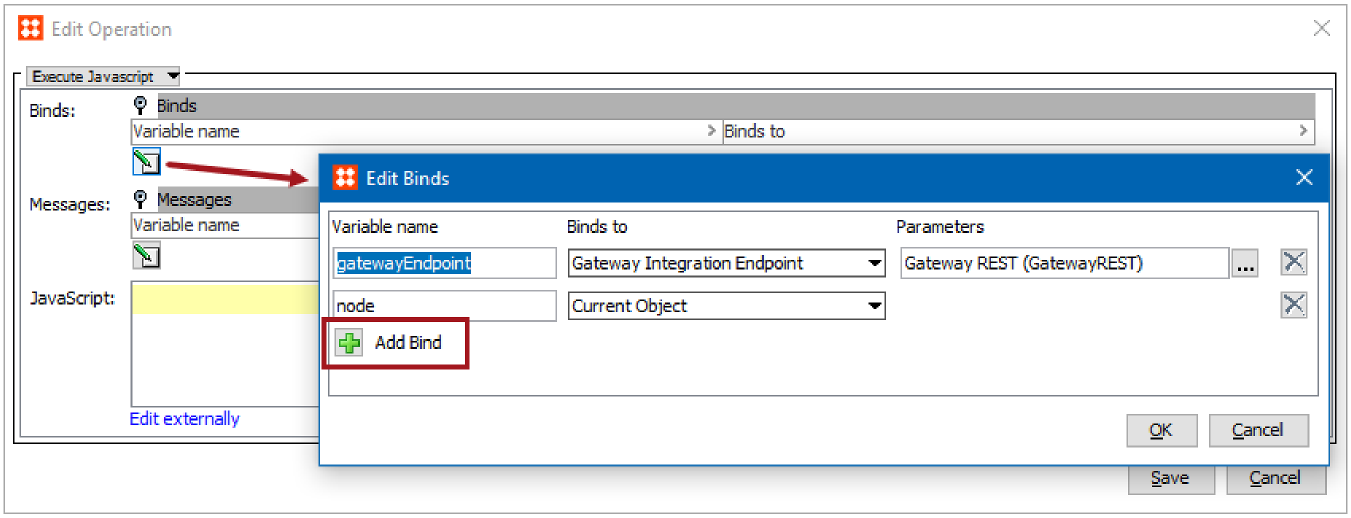


Business Rule Editor - JS Gateway Endpoint

Edit Operation

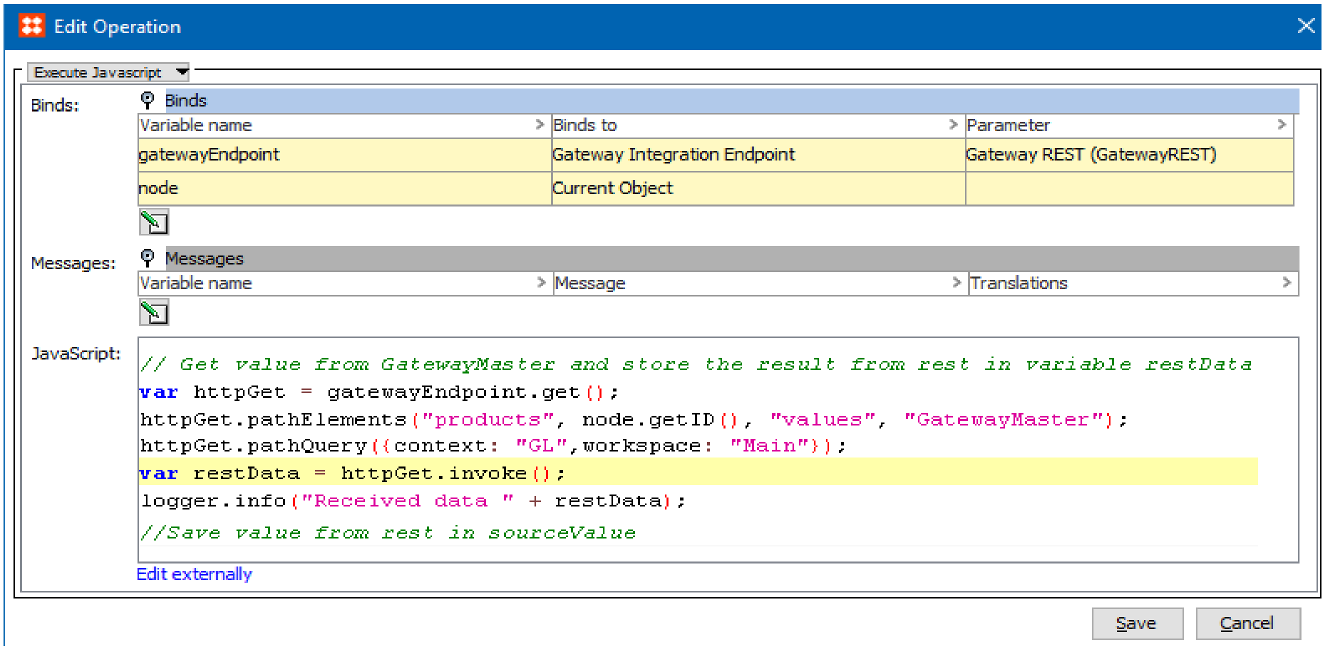
Execute Javascript

6. Click the **Edit Binds** button to display the Edit Binds dialog, then click the **Add Bind** button. Bind the JavaScript to the relevant gateway endpoint and click **OK**.



Variable name	Binds to	Parameters
gatewayEndpoint	Gateway Integration Endpoint	Gateway REST (GatewayREST)
node	Current Object	

7. In Edit Operation dialog, enter the JavaScript required to access the gateway. The following example uses a REST gateway integration endpoint to access the external server for information. For more information about the available REST methods, see the **Gateway Integration Endpoint Bind** section.



Variable name	Binds to	Parameter
gatewayEndpoint	Gateway Integration Endpoint	Gateway REST (GatewayREST)
node	Current Object	

Variable name	Message	Translations

```
// Get value from GatewayMaster and store the result from rest in variable restData
var httpGet = gatewayEndpoint.get();
httpGet.pathElements("products", node.getID(), "values", "GatewayMaster");
httpGet.pathQuery({context: "GL",workspace: "Main"});
var restData = httpGet.invoke();
logger.info("Received data " + restData);
//Save value from rest in sourceValue
```

8. In the **Edit Operations** dialog, click the **Save** button.

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

GDSN Binds

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

Bind Name	Description
GDSN Product	Binds the selected GDSN product for use in the Pre action business action parameter or in the validation conditions.
GDSN Provider Datapool	Binds the selected GDSN provider datapool for use in the Pre action business action parameter or in the validation conditions.
GDSN Publisher Data Map	For a business action used by the GDSN Provider, this bind can get the value of a specific XPath. The XPath value can subsequently be used to retrieve the corresponding string in the GDSN XML file.
GDSN Receiver Data Map	For a business action used by the GDSN Receiver, this bind can get the value of a specific XPath. The XPath value can subsequently be used to retrieve the corresponding string in the GDSN XML file.
GDSN Recipient	Binds the selected GDSN recipient for use in the Pre action business action parameter or in the validation conditions.
GDSN Target Market	Binds the selected GDSN target market for use in the Pre action business action parameter or in the validation conditions.
GDSN Validation Logger	Access the GDSN validation logger for use in the Pre action business action parameter or in the validation conditions.
GDSN Publisher Product Validation	Provides access to information about a product instance that is registered for a given data pool. For example, this could be used to write a script that can validate if all required attributes for a given data receiver have been populated on a given product. For more information about the Validation Condition option, see **GDSN in Web UI Overview**.

XPath Expressions

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

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

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

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

Parameter 1:

- name: ResponseCode
- xPath:

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

Parameter 2:

- name: DocumentException
- xPath:

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

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

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

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

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

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

MongoDB Binds

Business Actions and Business Conditions can use the following binds for the mongoDB integration.

Bind Name	Description
JSON Document	When used in a business action referenced from the 'Mongo delivery' plugin, the binding gives access to a JSON object for the data that has just been persisted.

MongoDB Context	Gives access to an object with the following methods:
	`getMongo()`
	The MongoDB database via the com.mongodb.Mongo object. For more information, see http://api.mongodb.org/java/current/com/mongodb/Mongo.html.
	`getContext()`

Bind Name	Description

	`getContextID()`
	`getRawDBName()`

Object Aspects Binds

Business Actions and Business Conditions can use the following binds.

Bind Name	Description
Current Object	For more information, see the **Current Object Bind** section of the Business Rules documentation.

ID	Bind resolves to the ID of current object (Java String). In this example, ID is bound to the variable 'id'. The code checks if the ID starts with an X.
	`if (id.charAt(0) == X) {`
	`// Do something`
	`}`

Name	Bind resolves to the name of current object (Java String or null if no name).
	In this example, 'Name' is bound to the variable 'name'. The code checks if there is a name.
	`if (name) {`
	`// Do something`
	`}`

Other Binds

Business Actions and Business Conditions can use any of the following binds.

Bind Name	Description
Conditionally Invalid Values	Use with Conditional Attributes. For details about this bind, see the conditional attribute section of the Business Rules documentation.

Bind Name	Description
	For information about creating conditional attributes, see the Conditional Attribute Handling section of the System Setup / Super User Guide.

Import Change Info	Use when the Business Rule is running in an import; accesses what has changed with the import (for example, new / existing object, updated attributes). This object will be populated for each object in the import.
	The object contains 2 members:
	`isUnmodified()`
	• true when an object is unchanged by the import
	• false when the object is new or modified
	`getChanges()`
	• returns a 'changes' object with a getAttributes() method that lets you see which Attribute values are changed by the import

Logger	Shorthand for the Java Logger class, Logger writes to the main STEP log file on the application server (step.*.log) when the condition or action runs. This variable is always bound in, gives access to a java.util.logging.Logger object, and allows logging of messages at the following severity levels:
	`logger.fine("Message to log");`
	`logger.info("Message to log");`
	`logger.warning("Message to log");`
	`logger.severe("Message to log");`
	By default, the STEP log does not register messages at the 'fine' severity level.
	After development and debugging a new Business Rule, including the Business Rule ID and the context in the log message aids in troubleshooting when required.
	Note: When testing a business rule, Logger writes directly to the test dialog, not to the STEP log file.

Lookup Table Home	The interface has a single public method that returns a String containing either the lookup value or the original value if no match could be made: `getLookupTableValue(String assetID, String value)` • The first argument is the ID of the lookup table (an asset) • The second argument is the string for which you want to get the lookup value.

Bind Name	Description
	For more information, see the Using Lookup Tables section of the Business Rules documentation.
Mailer	Allows the JavaScript to send emails. The bound class is the API Mailer object. Binds to the Mailer domain interface available for actions and conditions (although typically used with actions). After mail server settings are configured on STEP, this bind allows sending simple emails. In this example, 'Mailer' is bound to the variable 'mailer'. `mailer.send("yourfriend@work.com ", "someone@somewhere.com", "Hello! ", "Please use Web UI to review the imported items.");`

Using Business Rules with Conditional Attributes

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

Using business rule binding for Conditional Validity

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

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

Here is a JavaScript example for clearing conditionally invalid values.

Required bindings:

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

```
//acquire an iterator - easiest way to deal with Sets
var iter = invalid.iterator();
//iterate over conditionally invalid values
while (iter.hasNext()){
//get the value
var val = iter.next();
//test if there is a value and if it is local
var clean = val.isLocal() && val.getSimpleValue();
//do some chatting to the log
var msg
if (clean)
msg = "Cleaning up";
else
msg = "Ignoring";
msg += " conditionally invalid ";
if (val.isLocal())
msg += "local value";
else
msg += "non-local value";
msg+= val.getAttribute().getID() + " [" + val.getSimpleValue() + "]";
logger.info( msg );
//in case of local value - delete it
if (clean) val.setSimpleValue("");//empty string handled as null and deletes local values
}
```

Using business rule binding of Workflow State

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

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

Required bindings:

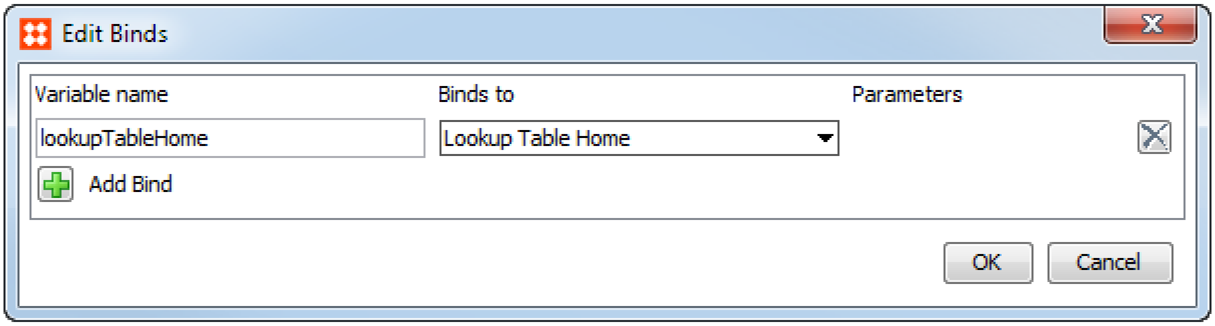
- state:"Workflow state"-binding
- node:"Current Object"-binding
- logger:"Logger"-binding

```
//up front test if the import carries expectation of specific workflow state
if (!state) {
logger.info("No import state provided");
//could have instead return some rejection to say no import if a state is not supplied
return true;
}
var instance, task, msg;
//acquire the workflow instance for the node and workflow
instance = node.getWorkflowInstance(state.getWorkflow());
if (!instance) {
```

```
//simply reject since workflow has been terminated or never started
msg = "Rejected : Workflow " + state.getWorkflow().getTitle() + " not started for " +
node.getTitle();
logger.info(msg);
return msg
}
//now look for actual task for node in supplied state
task = instance.getTask(state);
if (!task){
//no task also causes rejection
msg = "Someone might have changed data " + node.getTitle() + " has no task for state " +
state.getID() + " in worflow " + state.getWorkflow().getTitle();
logger.info(msg);
return msg;
}
logger.info('Accepted: we have a task and will now continue with import');
return true;
```

Using JavaScript with Lookup Tables

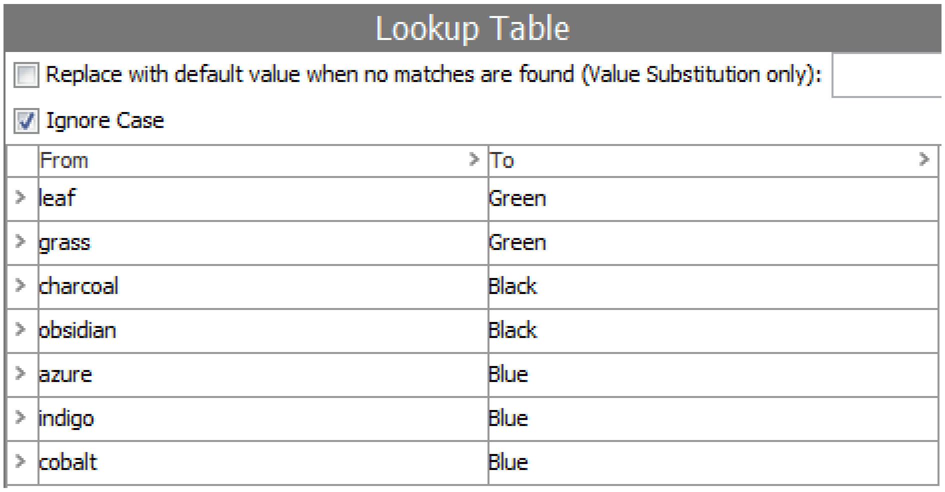
Transformation Lookup Tables are used to make replacements via the 'Lookup Table Home' binding.



Edit Binds

Variable name	Binds to	Parameters
lookupTableHome	Lookup Table Home	

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



Lookup Table

☐ Replace with default value when no matches are found (Value Substitution only):

☑ Ignore Case

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

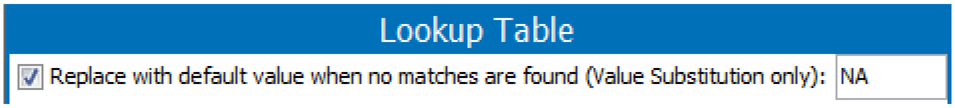
The following JavaScript accesses the ColorReplacement lookup table and returns 'Green':

```
var output = lookupTableHome.getLookupTableValue("ColorReplacement", "leaf");

logger.warning(output); // WARNING: Green
```

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

```
var output = lookupTableHome.getLookupTableValue("ColorReplacement", "orange");

logger.warning(output); // WARNING: orange
```

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



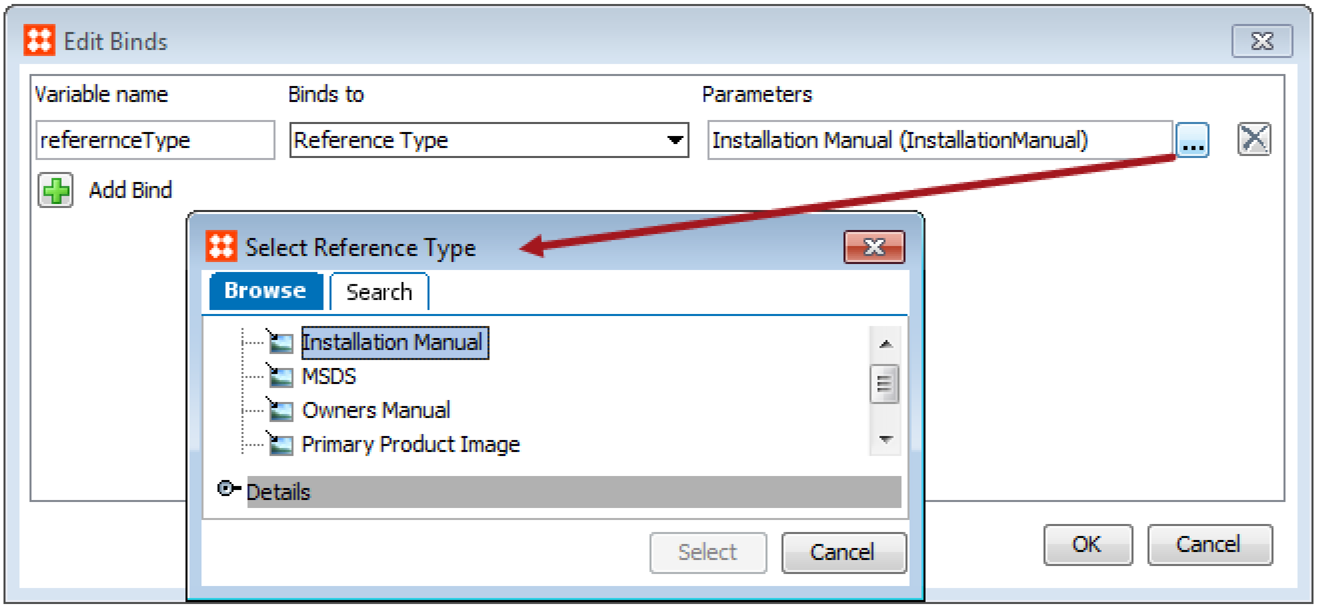
Lookup Table

☑ Replace with default value when no matches are found (Value Substitution only): NA

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

Reference Type Bind

Business Actions and Business Conditions can use the Reference Type bind to access the Reference Type or Link Type selected from the picker.



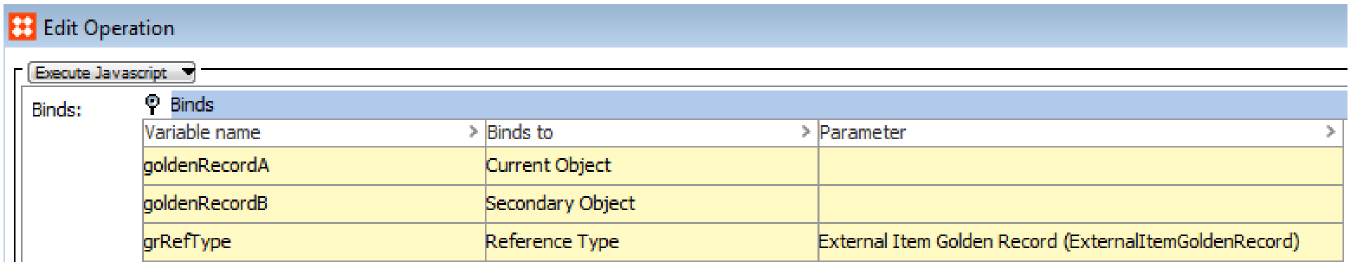
Variable name	Binds to	Parameters
referernceType	Reference Type	Installation Manual (InstallationManual)

Secondary Object Bind

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

For example, a business action selected in the Matching Algorithm's Match Action when merging two golden records could work on the two golden records to be merged using the Current Object and Secondary Object binds.

For more information about golden record handlers, see **Updating Golden Records**.



Variable name	Binds to	Parameter
goldenRecordA	Current Object	
goldenRecordB	Secondary Object	
grRefType	Reference Type	External Item Golden Record (ExternalItemGoldenRecord)

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

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

STEP Manager Bind

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

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

In the example code below, the STEP Manager is bound in as **manager**, and gets the value of the Attribute 'Description' on a specific product in current Context / Workspace.

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

Tree Object Binds

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

Bind Name	Description
Asset	Binds the selected asset to the action or condition variable.
Classification	Binds the selected classification to the action or condition variable.
Entity	Binds the selected entity to the action or condition variable.
Product	Binds the selected product to the action or condition variable.

User Binds

Business Actions and Business Conditions can use any of the following simple binds. These binds are especially useful when a particular object is needed repeatedly in the JavaScript.

Bind Name	Description
User	Binds the selected user to the action or condition variable.
User Group	Binds the selected user group to the action or condition variable.

Workflow Binds

Business Actions and Business Conditions can use any of the following workflow binds.

Bind Name	Description

Current Transition	Bind available to track properties of a transition currently being performed in a STEP workflow. This allows JavaScript business rules access to information about the event that started the transition and to the message entered as part of submitting the event.
	The bind is only available from JavaScript-based business conditions and actions executed from within the workflow. That is, actions executed ‘On Entry,’ ‘On Transition,’ or ‘On Exit’ and conditions on transitions. It is not available from workflow start conditions, actions executed when deadlines are reached or conditions on conditionally mandatory attributes.
	Below is an example of a simple JavaScript action that stores the submit message in a workflow variable. Setup includes the following prerequisites:
	• Binding on Current Object named: node
	• Binding on Current Workflow named: workflow
	• Binding on Current Transition named: currentTranstion
	• Simple Workflow Variable named: lastMessage
	`var workflowInstance = node.getWorkflowInstance(workflow);`
	`var msg = currentTransition.getMessage();`
	`if (!msg)`
	`        msg = '[none]';`
	`workflowInstance.setSimpleVariable('lastMessage',msg);`
	Here is an example of a JavaScript condition that checks for presence of message when a specific event is submitted. Setup includes the following prerequisites:
	• Binding on Current Transition named: currentTransition
	• Business rule Message named: messageRequired
	`if (currentTransition.getEvent() &&`
	`   'submit' == currentTransition.getEvent().getID() &&`
	`   !currentTransition.getMessage()){`
	`        return new messageRequired();`
	`} else {`
	`        return true;`
	`}`

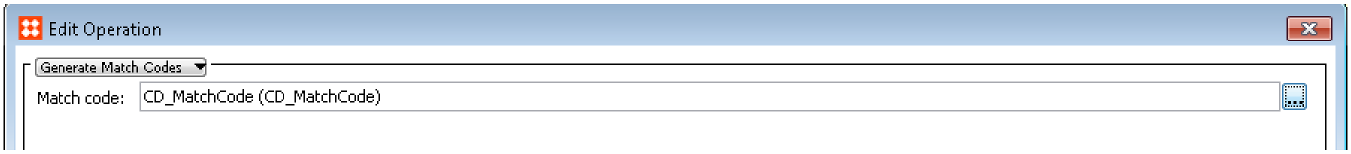
Bind Name	Description
	getEvent() will return the event (object) used to trigger the current transition; should be null if there is no event; getMessage() will return the submit message as a string or null if no message has been supplied.
Current Workflow	Bind available for conditions evaluated and actions executed from a workflow, giving access to the current workflow. In this example, Current Object is bound to 'currentObject' and Current Workflow is bound to 'currentWorkflow'. The code stores the assignee for the 'Enrich' task in a variable. Notice that this code will only work if the JavaScript is executed from within the workflow and current object is in the 'Enrich' state. `var assignee = currentObject.getTaskByID(currentWorkflow.getID(),"Enrich").getAssignee();`
Workflow Parameters	Bind that allows parameters to be passed to a workflow when it is started. Typically used for Flatplanner / page planner workflows started from the Flatplanner component.
Workflow State	Bind allows access to Workflow State information from a Smartsheet.

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

Business Action: Generate Match Codes

It is possible to generate match codes as part of a business action.

In the Match code field, click the ellipsis button **(...)**, and then select the relevant match code.



Generate Match Codes

Match code: CD_MatchCode (CD_MatchCode)

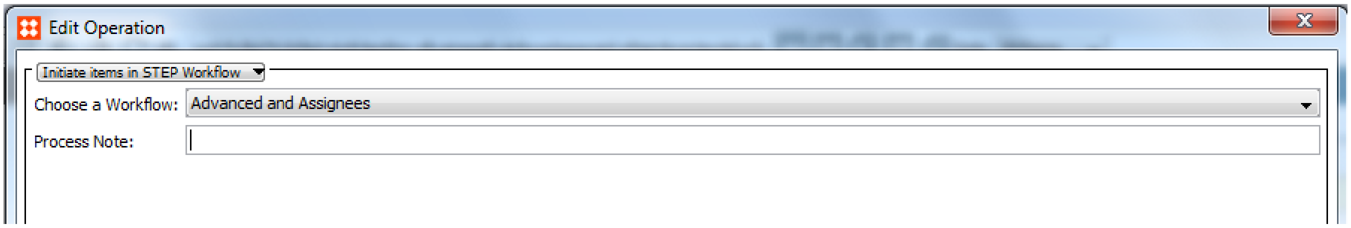
When the operation is performed it generates match code values for the given object.

For more information about match codes, see Creating Match Codes in the Data Quality documentation.

Business Action: Initiate Item In STEP Workflow

Initiates the valid object or objects in the selected workflow.

You can add a note that describes the cause of the action. The text is displayed in the state log for the objects and the workflow.



Choose a Workflow: Advanced and Assignees

Process Note:

Business Action: Merge Attribute Values

Copies the value of one attribute into another attribute. You can specify whether to keep or overwrite existing values.

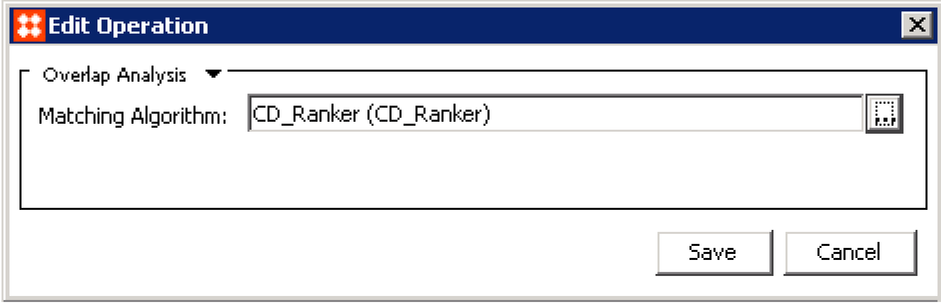
Example: Merge Attribute Values

Source Attribute value	Target Attribute value	Overwrite setting	Target after merge
<empty>	OriginalValue	<any>	OriginalValue
NewValue	<empty>	<any>	NewValue
NewValue	OriginalValue	[Overwrite Existing]	NewValue
NewValue	OriginalValue	[Keep Original Values]	OriginalValue

Business Action: Overlap Analysis

This plugin compares records in pre-defined format to each record with the same object type by specifying a matching algorithm.

This operation is only intended to be used together with an import running in Test mode. In this case it will report statistics of how many objects in import file will match existing objects in STEP without actually importing the data.

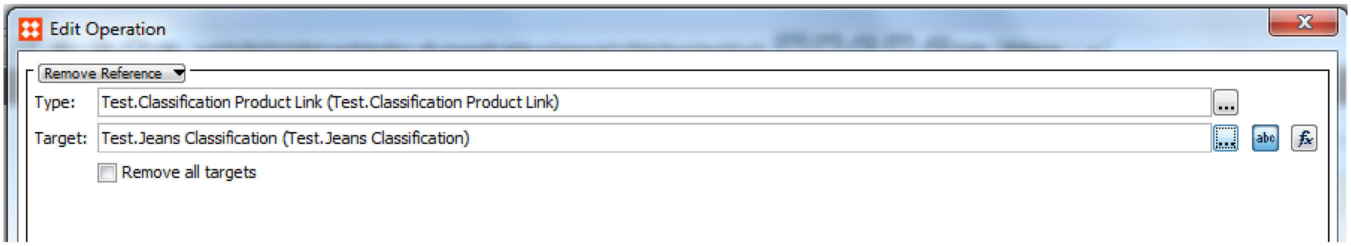


Matching Algorithm: CD_Ranker (CD_Ranker)

Business Action: Remove Reference

Removes references to a specific target or all references of a specific type.

- To remove all references of a specific type, select the reference type and then check Remove All.
- To remove references to a specific target, select a reference type and then the target or specify a function in the Function editor.



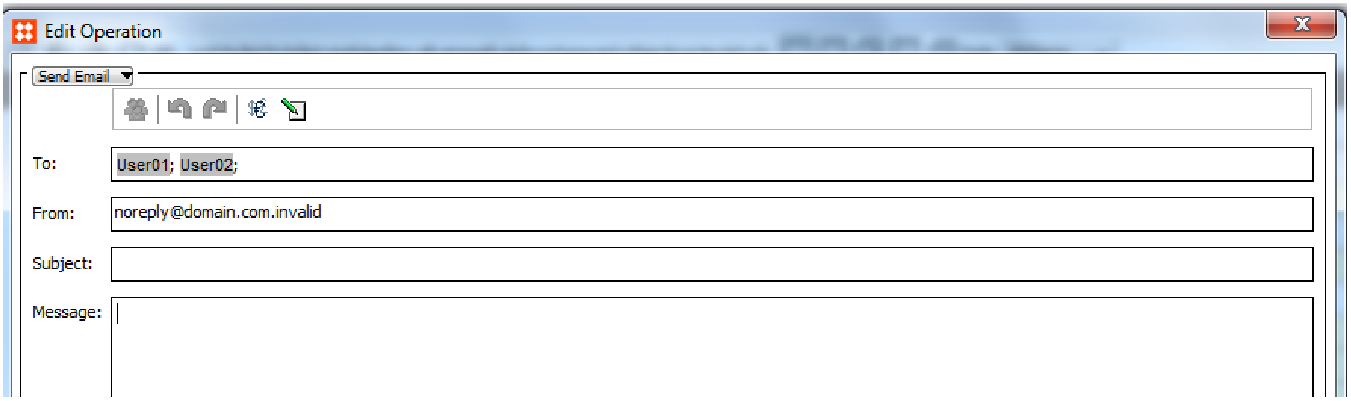
Type: Test.Classification Product Link (Test.Classification Product Link)

Target: Test.Jeans Classification (Test.Jeans Classification)

□ Remove all targets

Business Action: Send Email

Sends emails to specified recipients.



To: User01; User02;

From: noreply@domain.com.invalid

Subject:

Message:

To and From

In the **To** field, enter the relevant email addresses manually, or click the **Add recipient** icon .

- Manually typed email addresses separated by semicolon (;), comma (,) or a space.
- A STEP User or User Group
- An email address from an attribute of the object the business action is executed on.
- An email address read from a workflow variable. This requires that the object the business action is executed on has been initiated in the specified workflow.

An attribute or a workflow variable can contain several email addresses if they are separated by semicolon (;), comma (,) or spaces.

The actual email addresses are resolved when the business action is run. You can specify any number of recipients.

The default **From** address is the default address specified in the system's configuration properties. You can change the from address to something else.

Inline References

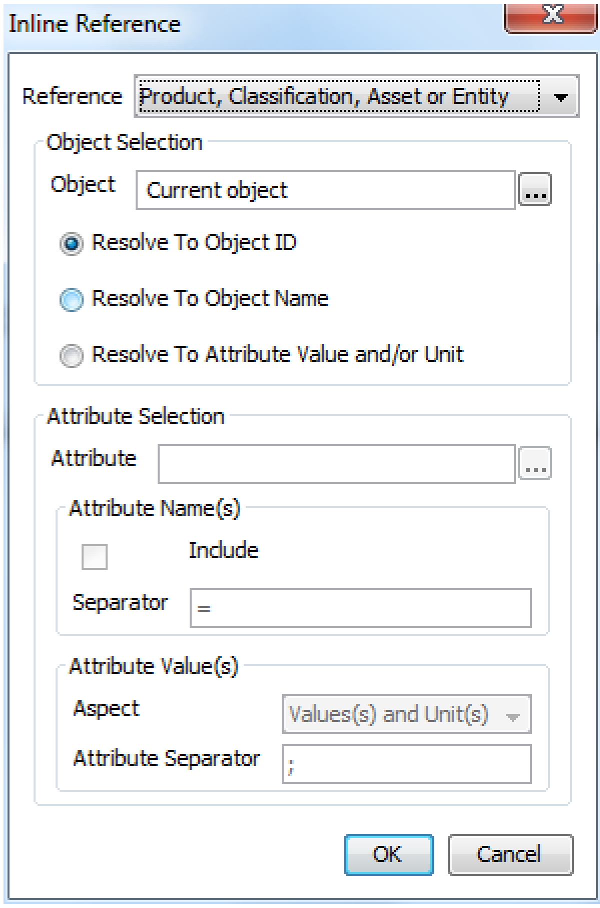
Click the Insert inline reference icon to insert inline references to data, including object IDs and names, attributes and workflow variables.

For workflow variables, the object the business action is executed on must have been initiated in the specified workflow.

You can select the current object as well as other specific objects. The current object is the object the business action is executed on. This provides the recipient with information about which change caused the email to be sent.

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

In a message, you can edit an inline reference by right-clicking it and choosing Edit Inline Reference.



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)

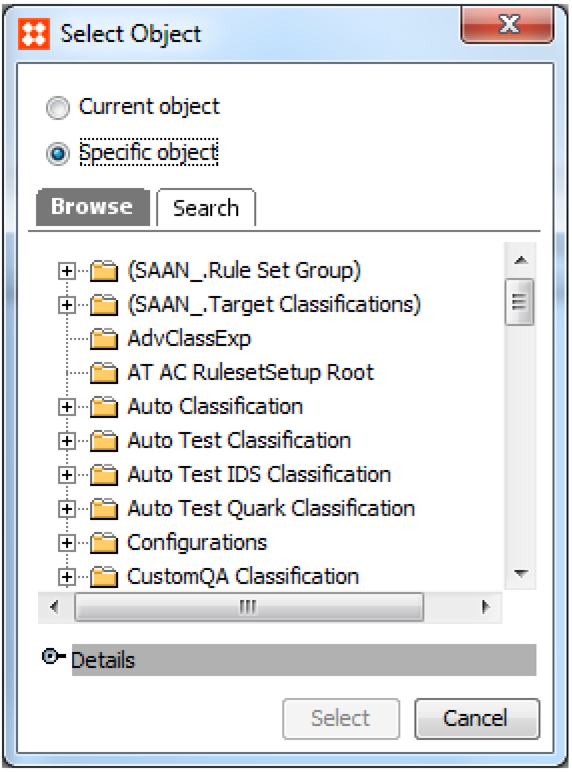
□ Include

Separator: =

Attribute Value(s)

Aspect: Values(s) and Unit(s)

Attribute Separator: ;



Select Object

○ Current object

● Specific object

Execution

Addresses must have the following format.

```
user@domain.com
```

If an invalid email address is found when the action is executed, the action fails and results in an error. This happens, for example, if part of an address has been left out by mistake, or if an attribute contains an unexpected value. If an address is of the right format, typing errors are not detected.

If no recipient is specified or if no valid email address can be found for any of the specified recipients, an error occurs upon execution.

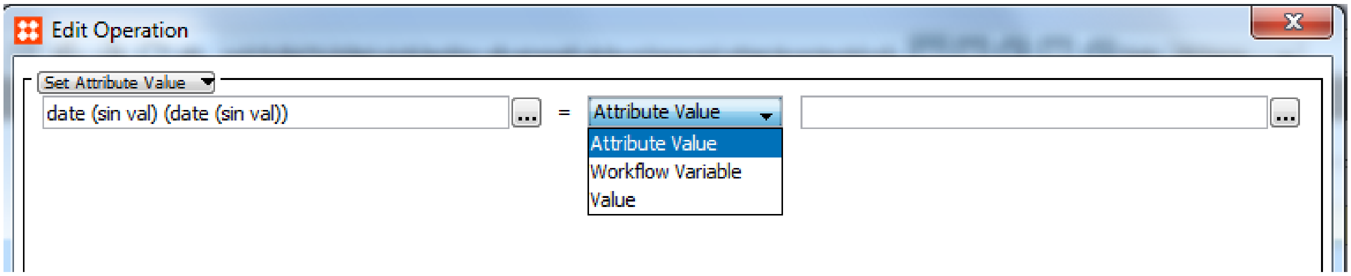
If an inline reference cannot be resolved during execution, it results in an error.

Business Action: Set Attribute Value

Enables the assignment of static values to attributes, attribute value, workflow variables or static values to attributes.

You specify a target attribute and either a source attribute, workflow variable or a static text.

1. In the **Attribute** field, click the ellipsis button **(...)** and search or browse for a target attribute.
2. In the **Selection** field, choose if the source content is an attribute value, workflow variable or a static text. The available fields depend on your selection.
   - **Attribute Value**: click the ellipsis button **(...)** to search or browse for the relevant value
   - **Workflow Variable**: click the ellipsis button **(...)** to search or browse for the preferred workflow, and then select the relevant variable from the list.
   - **Value**: enter the relevant value.


Edit Operation

Set Attribute Value

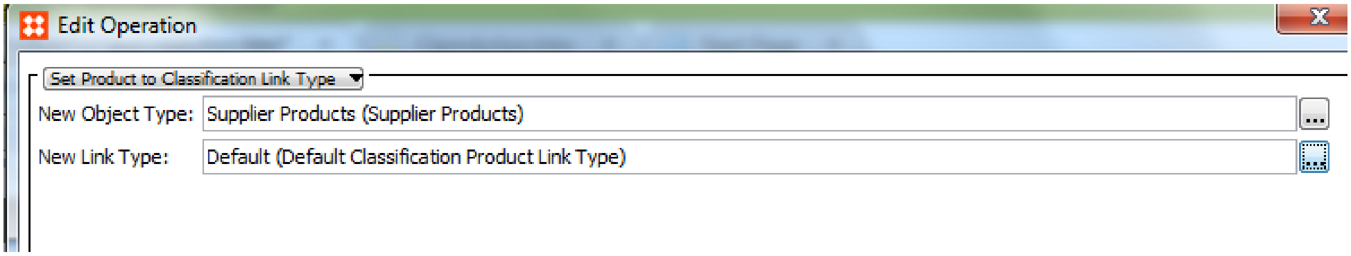
date (sin val) (date (sin val)) = Attribute Value

Business Action: Set Product to Classification Link Type

Assigns a new object type and classification link type to the current product. The business action can be used to change product to classification link types for products that are linked in to a given classification.

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.

- Select the new object type, and then select the relevant classification link type.



Set Product to Classification Link Type

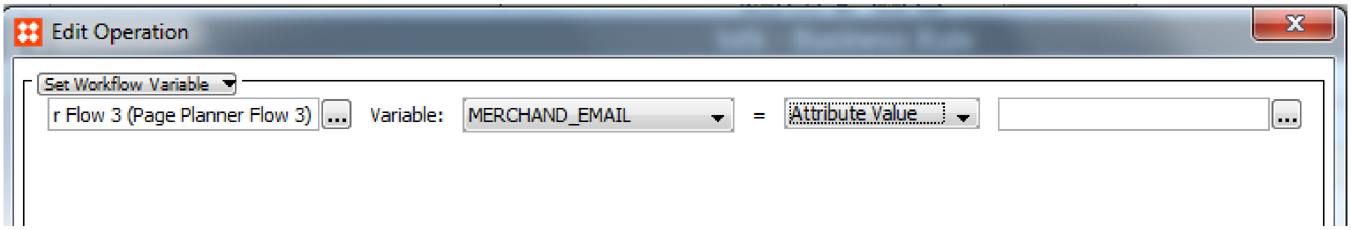
New Object Type: Supplier Products (Supplier Products)

New Link Type: Default (Default Classification Product Link Type)

Business Action: Set Workflow Variable

Enables assignment of the following attributes, workflow variables and static values to workflow variables.

Selecting the **Set Workflow Variable** Action will enable specifying a target Workflow Variable and either a source Attribute, Workflow Variable or a Static Text.



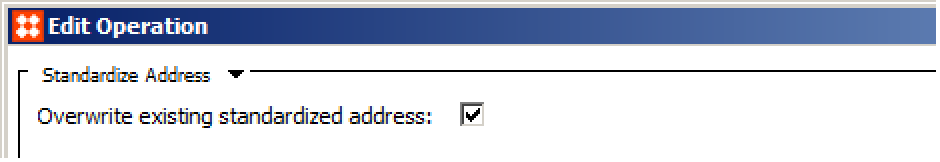
Set Workflow Variable

r Flow 3 (Page Planner Flow 3) ... Variable: MERCHAND_EMAIL = Attribute Value ...

1. For the first field, click the ellipsis button **(...)** , and then browse or search for the relevant target workflow.
2. In the **Variable** field, select the relevant variable.
3. From the list, select whether the source content is **Attribute Value**, **Workflow Variable** or **Static Text**. The available fields depend on the selection.
   - **Attribute Value**: click the ellipsis button **(...)** to search or browse for the relevant value
   - **Workflow Variable**: click the ellipsis button **(...)** to search or browse for the preferred workflow, and then select the relevant variable from the list.
   - **Value**: enter the relevant value.

Business Action: Standardize Address

Overwrites the existing standardized address.



Standardize Address

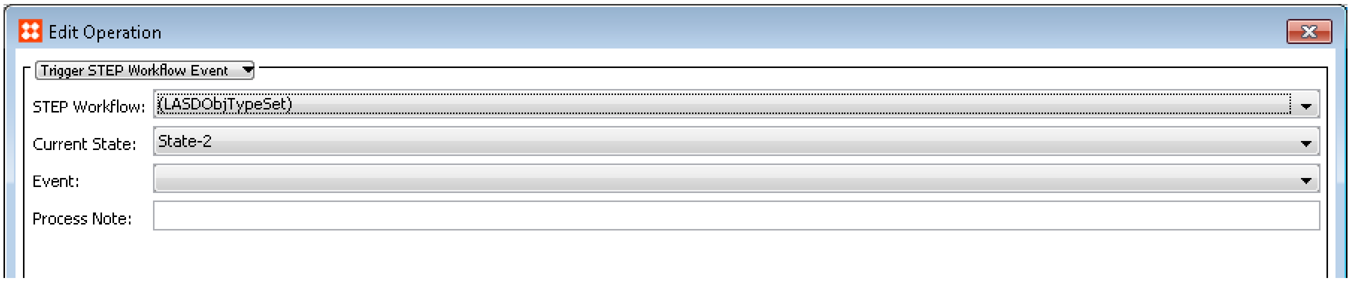
Overwrite existing standardized address: ☑

Business Action: Trigger STEP Workflow Event

Triggers a specified workflow event.

1. From the **STEP Workflow** list select the relevant workflow.
2. From the **Current State** list, select the state the workflow object must be in when the action is applied.

3. From the **Event** list, select the event that the action triggers.
4. In the **Process Note** field, you can enter a comment which will be written in the state log when the event is triggered.



STEP Workflow: (LASDObjTypeSet)

Current State: State-2

Event:

Process Note:

Business Conditions

These business conditions are available and are described in further detail in following individual sections.

Important: Business conditions are optimized for determining the true / false result of read-only scenarios. Although it is technically possible to write business conditions to change data, doing so can result in unpredictable errors and should therefore not be attempted. To change data effectively, use a business action. For more information on business actions, see **Business Action**.

Condition	Description
Attribute Value Comparison	Compares the value of one attribute with either a constant or the value of another attribute on the current node. For more information, see **Attribute Value Comparison**.
Evaluate JavaScript	Runs defined JavaScript code added on the business condition using the same bindings that are available for the business action 'Execute JavaScript'. **Important:** Although the same binds are available, changing STEP data via an Evaluate JavaScript business condition is not supported. For more information, see Execute JavaScript.
Function	Makes it possible to use functions to define business conditions. For more information, see **Business Condition: Function**
LOV Cross-Validation	Specifies that the value of one List of Value (LOV) attribute limits which values are valid for another LOV attribute. For more information, see **Business Condition: LOV Cross-Validation**.
Reference other Business Condition	Allows the current business condition to reference other business conditions. For more information, see **Business Condition: Reference other Business Condition**.
Valid Hierarchies	Specifies sub-hierarchies where the condition is true. For more information, see **Business Condition: Valid Hierarchies**.
Validate Google Shopping Product	Tests that a product contains the attributes that are required for all products submitted to Google Shopping.

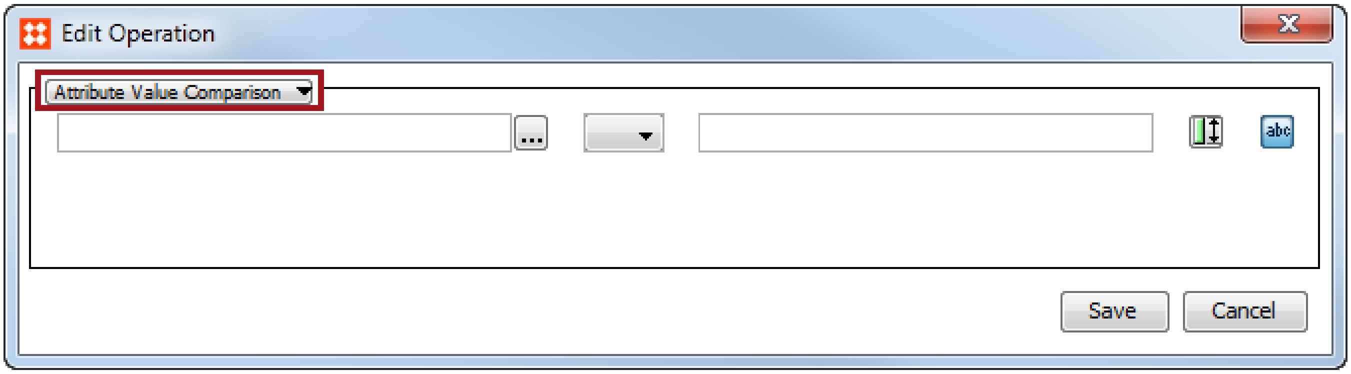
Condition	Description
	For more information, see **Google Shopping Business Condition** in the Google Shopping documentation.
Validate Product Variant	Validates product variants for duplicates. This condition uses the configuration created when setting up Product Variant Families, so there is no additional configuration. For more information, see **Business Condition: Validating Product Variant**.

To add a business condition, see **Specifying a Business Condition**.

Specifying a Business Condition

1. In **System Setup**, expand **Business Rules** and select the relevant business condition.
2. On the **Business Rule tab** click **Edit Business Rule** to open the business rule editor.
3. In the **Valid Object Type** field, click the ellipsis button **(...)** and select the object types for the condition.
4. In the lower left corner, click **Add New Business Condition** link.
5. On the Operations tab, click the **Edit Operation** button.
6. Use the Business Condition dropdown to select the preferred condition. For more information about the options, see the **Business Conditions** section of **Business Rules** documentation.

Important: If you specify more than one condition, all conditions must be true for the overall condition to be true.



7. Click **Save** to save the changes.

Business Condition: Attribute Value Comparison

Compares the value of one attribute on the current node with a constant or the value of another attribute on the current node.

Numeric comparison is used when both values are number based (fractions, integers and numbers). Otherwise, alphanumeric comparison is used, except for dates (date, ISO date and time).

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.

The following are examples of the Attribute Value Comparison condition and the results:

- 15-jun-2015 (date attribute value) < 15-jul-2015 (constant) = true
- 15-jun-2015 (text attribute value) < 15-jul-2015 (constant) = false
- 15-jun-2015 (date attribute value) < 2015-07-15 (ISO date attribute value) = true
- 80 cm (number attribute value) < 1 (constant) = true (base unit is m)
- 80 cm (number attribute value) < 1 cm (constant) = false
- a (text attribute value) > 1 (number attribute value) = false

Evaluate JavaScript

The Business Condition operation 'Evaluate JavaScript' enables you to use the same STEP Public Java API methods that are available in the Business Action operation 'Execute JavaScript'.

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, see the **Execute JavaScript** section of the **Business Rules** documentation.

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 (translatable).

The Translatable message option is available in both JavaScript Business Actions and Conditions. For more information, see the **Translatable Messages for JavaScript Business Rules** section of the **Business Rules** documentation.

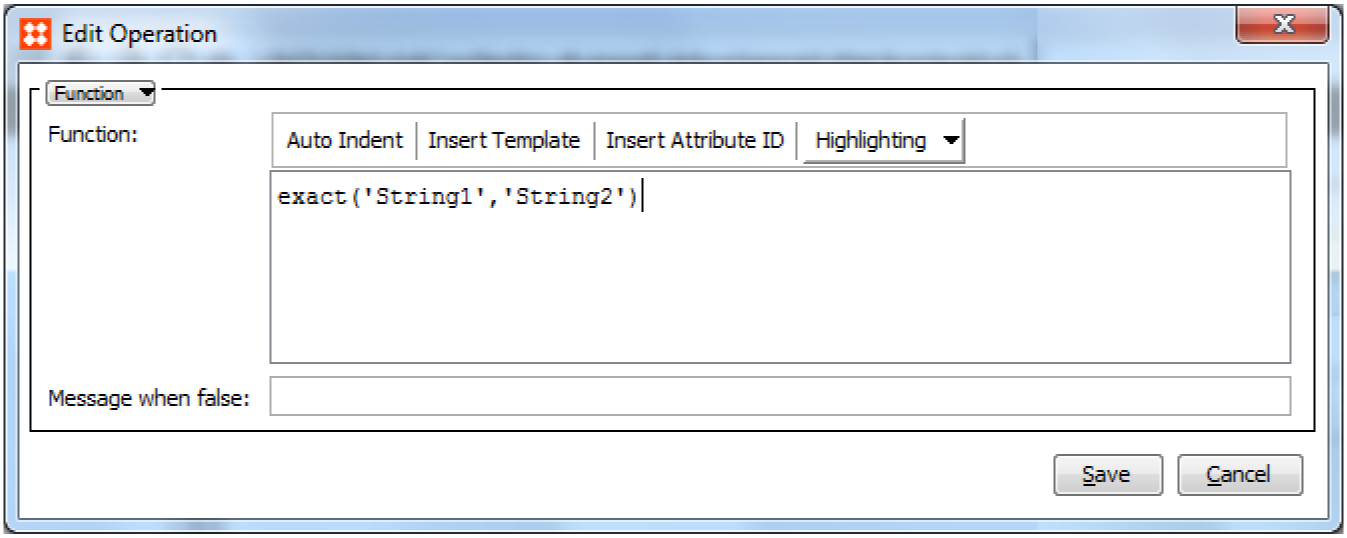
Business Condition: Function

It is possible to define business conditions by using functions.

In the Edit Operation dialog, you can add function templates and you can insert an attribute ID, by navigating to the relevant attribute.

- In the **Edit Operation** dialog, right-click **Insert Template**, to insert a function template. You can select a template from the template library.

For more information on functions and function templates, see **Function Editor** in the **System Setup / Super User Guide**.



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

If a function returns `1`, the condition is true. If a function returns `0` or anything else but `1`, the condition is false. I

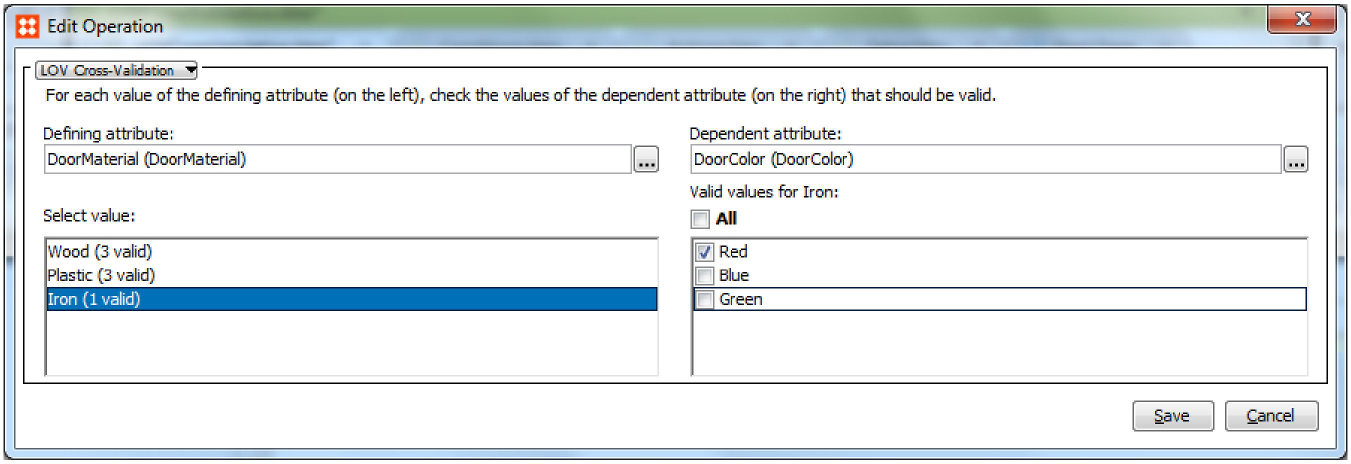
As the following example shows, it is not necessary for a function to explicitly return `0` or `1`

The `exact(prodval('Description'),"Expected value")` returns `1` if the value of the `Description` attribute on current node equals `Expected value`. Otherwise it returns `0`.

Business Condition: LOV Cross-Validation

For Web UI only, this condition allows a user to specify legal relationships between the values in two List of Values (LOV) attributes, meaning that the value of the defining LOV attribute determines the valid values for the dependent LOV attribute. This cross-validation is not applicable in workbench.

1. Verify that the **Allow Users to Add Values** option is set to **No** for both the Defining attribute LOV and the Dependent attribute LOV. For more information, see **Creating an LOV** in the **System Setup** documentation.
2. Verify that the **Use IDs on values** option is set to **Yes** for both the Defining attribute LOV and the Dependent attribute LOV. For more information, see **Adding IDs to Values in LOV** in the **System Setup** documentation.
3. Verify that the **Defining attribute** is single-valued and of the validation type LOV.
4. Verify that the **Dependent attribute** is of the validation type LOV. This attribute can be single or multi-valued.
5. For **Defining attribute**, click the ellipsis button **(...)** and select the attribute that will determine valid values for the dependent attribute.



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: DoorMaterial (DoorMaterial)

Dependent attribute: DoorColor (DoorColor)

Select value:

Wood (3 valid)
Plastic (3 valid)
Iron (1 valid)

Valid values for Iron:

□ All

☑ Red
□ Blue
□ Green

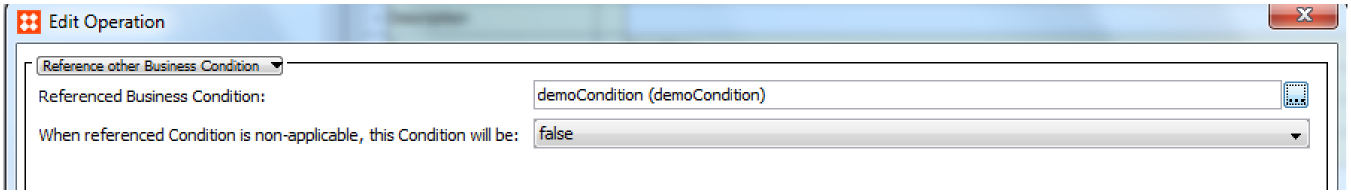
6. For **Dependent attribute**, click the ellipsis button **(...)** and select an attribute that will have limited valid values.
7. In the **Select value** area, highlight a value and then in the **Valid values for** area, select the values you want to be valid for the currently selected defining attribute value.

> **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.

8. Review the additional values in the **Select value** area and repeat the previous step as necessary.
9. Follow the steps described in **Using Business Conditions in Web UI** to set up the LOV cross-validation in Web UI.

Business Condition: Reference other Business Condition

Allows the current business condition to reference other business conditions.



Edit Operation

Reference other Business Condition

Referenced Business Condition: demoCondition (demoCondition)

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

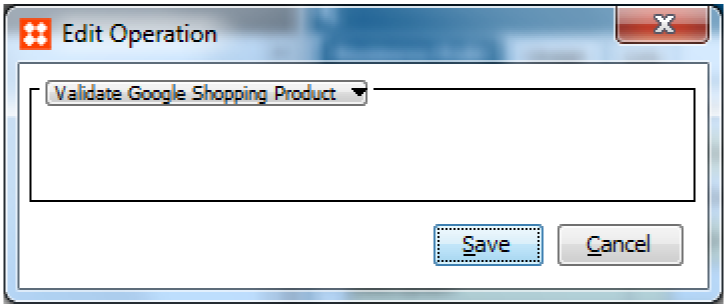
1. In the **Reference Business Condition** field, click the ellipsis button **(...)**, and select the business condition to reference.
2. Specify whether the current condition is **true** or **false** when the referenced condition does not apply.
3. Click **Save**.

Business Condition: Validate Google Shopping Product

Performs a simple validation before a product is submitted to Google Shopping. This makes it possible to validate products on approval or inside a workflow.

The function tests whether the node provides all attributes required for a product for Google Shopping. If a required attribute is missing from a product, the product may not be listed in Google Shopping.

All information required to perform the validation is found using the Google Shopping component model. Therefore no further configuration is required.



For more information about Google Shopping, see Google Shopping Business Conditions.

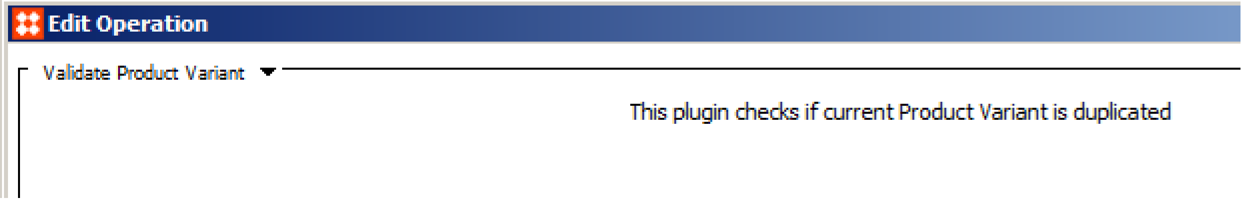
Business Condition: Validating Product Variant

It is possible to use a business condition to perform validation for product variant duplicates.

For more information about business conditions, see Business Conditions in the Business Rules documentation.

To use a business condition to validate variants, first create a new business condition.

1. In **System Setup**, expand **Business Rules**, and select the relevant business condition.
2. On the **Business Rule** tab, in the **Valid Object Type** field, click the ellipsis button **(...)**, and then select the object types that you want to apply the action to.
3. In the lower left corner, click **Add New Business Condition**.
4. Click the **Edit Operation** icon.
5. In the **Edit Operation** dialog, from the drop-down list, select **Validate Product Variant**.



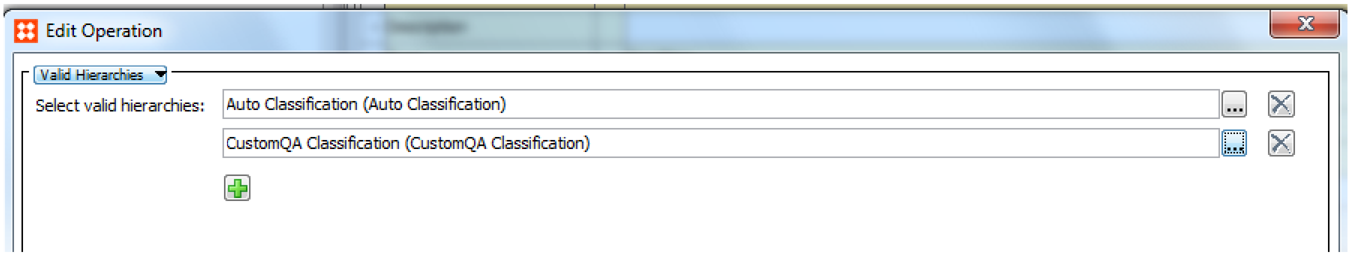
This plugin checks if current Product Variant is duplicated

The Validate Product Variant condition uses the configuration your created when you set up Product Variant Families, so there is no configuration to do. The action works on the Product Variant level, and returns false if the product variant is a duplicate.

Note: Running the Validate Product Variant business rule may affect performance depending on how many siblings and variant attributes that have to be checked.

Business Condition: Valid Hierarchies

Returns true when the object of the Condition is present below one of the specified hierarchy nodes. This operation is typically used on the 'Applies If' tab and serves as a pre-condition for testing the main Condition or applying an Action.



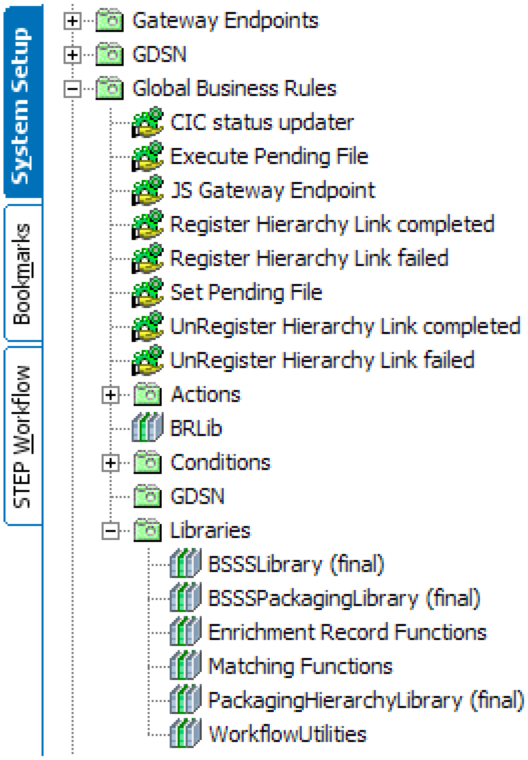
Select valid hierarchies: Auto Classification (Auto Classification)
CustomQA Classification (CustomQA Classification)

1. In **Select valid hierarchies**, click the ellipsis button **(...)**, and select the relevant hierarchy.
2. Click the plus icon to add all required hierarchies.
3. Click **Save**.

Business Libraries

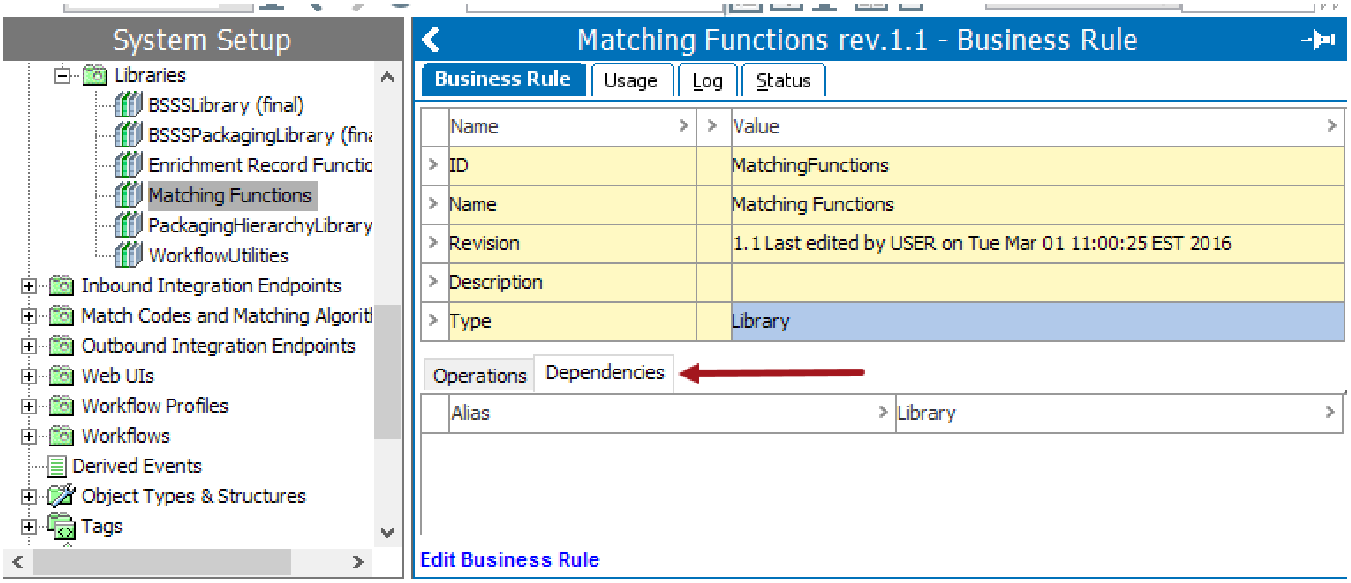
Business Libraries are a set of functions that can be reused in multiple business rules or other libraries. Libraries cannot be called independently and must be referenced from other actions or conditions.

Like actions and conditions, business libraries are objects that live in System Setup.



In business libraries, any number of JavaScript business rules can be drawn from 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 to be used, you must declare a dependency to the libraries. This is done so on the Dependencies tab.


System Setup
Libraries
BSSSLibrary (final)
BSSSPackagingLibrary (fina
Enrichment Record Functic
Matching Functions
PackagingHierarchyLibrary
WorkflowUtilities
Inbound Integration Endpoints
Match Codes and Matching Algorit
Outbound Integration Endpoints
Web UIs
Workflow Profiles
Workflows
Derived Events
Object Types & Structures
Tags
Matching Functions rev.1.1 - Business Rule
Business Rule
Usage
Log
Status
Operations
Dependencies
Edit Business Rule

Name	Value
ID	MatchingFunctions
Name	Matching Functions
Revision	1.1 Last edited by USER on Tue Mar 01 11:00:25 EST 2016
Description	
Type	Library

Alias	Library

For example: To check if a product is below another product , the following library function is used:

```
// Checks whether a Product is below another Product
function isProductBelow(prod, checkProdID) {
if(!prod instanceof com.stibo.core.domain.Product) throw "Function only works with
Products";
if(checkProdID == "Product hierarchy root") return true;
if(prod.getID() == "Product hierarchy root") throw "The top level Product is never below
another Product.";
var currentParentId;
var currentProd = prod;
while(true) {
currentParentId = currentProd.getParent().getID();
if(currentParentId == "Product hierarchy root") return false;
else if (currentParentId == checkProdID) return true;
else currentProd = currentProd.getParent();
}
}
Condition using function (Library alias = “lib”, Current Object bound to variable "obj")
return lib.isProductBelow(obj, "ID of STEP Product");
```

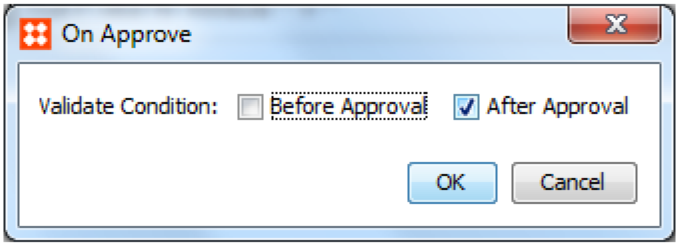
Using Business Rules During the Approval Process

Global business rules can be executed during the approval process.

To Validate Conditions or Perform Actions During Approval

1. In **System Setup**, expand **Business Rules**, and then select the relevant condition or action.
2. On the **Business Rule** tab, select **Edit Business Rule**.
3. In the **On approve** field, click the ellipsis button **(…)**.

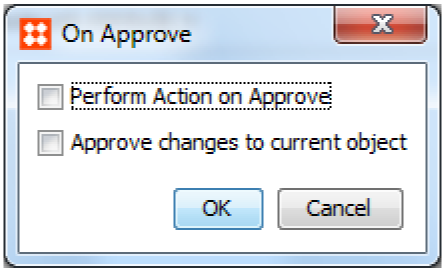
On conditions, specify when you want to validate the condition. It is possible to validate the condition both before and after approval.



On Approve

Validate Condition: □ Before Approval ☑ After Approval

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



On Approve

□ Perform Action on Approve

□ Approve changes to current object

To view a list of Business Rules to be run during Approve

1. In **System Setup** select **Users & Groups**, and then click the **System Settings** tab.
2. Scroll to **Business Rule Approve Overview**. Here you can see a list of the business rules that are configured to be executed or tested on Approve.

Example: The following describes what happens when conditions are validated and actions executed during the approval process.

When you have selected objects for approval the system carries out the following process.

1. Runs actions that are configured to be executed on approval.
2. Runs all business conditions configured to be validated before approval (for sub-conditions only until one of them fails). If one or more fail the system goes to step 5.
3. Performs the approval itself including standard dependency checks. If one or more fail the system goes to step 5.

4. Runs all business conditions configured to be validated after approval (for sub-conditions only until one of them fails, and performs standard mandatory checks. If one or more fail, the system goes to step 5.
5. Commits or rolls back depending on whether all conditions are fulfilled.

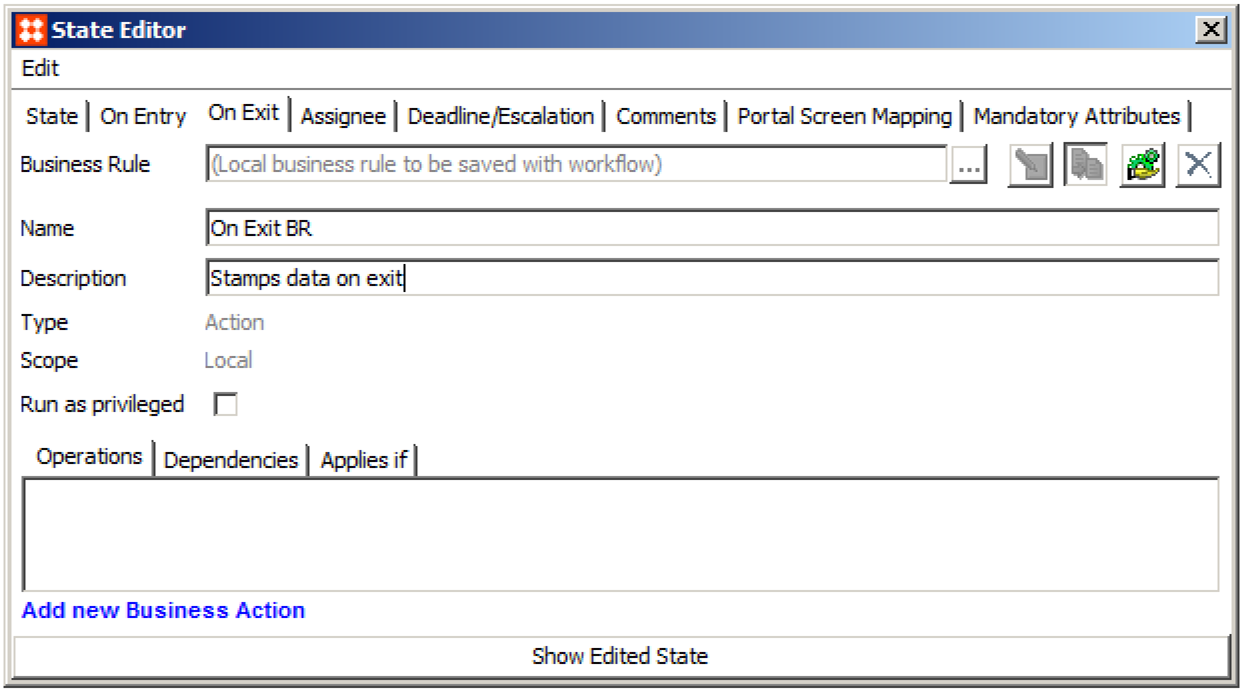
Using Business Rules in STEP Workflows

In workflows, business conditions are used to define the transitions that are legal in the workflow. Business actions can be executed whenever something happens in the workflow.

Workflow Designer

Defining a new business rule from within the State Editor of the STEP Workflow Designer, creates a local business action. Local business rules can only be used in the workflow where they were created.

You can also click the ellipsis button **(...)** and select an existing business rule.



Business Rule (Local business rule to be saved with workflow)

Name On Exit BR

Description Stamps data on exit

Type Action

Scope Local

Run as privileged ☐

Promoting a local business rule to a global business rule, creates a global copy and requires an ID. Click the hand icon to promote a business rule.

Editing a global business rule by clicking the edit icon results in changing all the places where the rule is used.

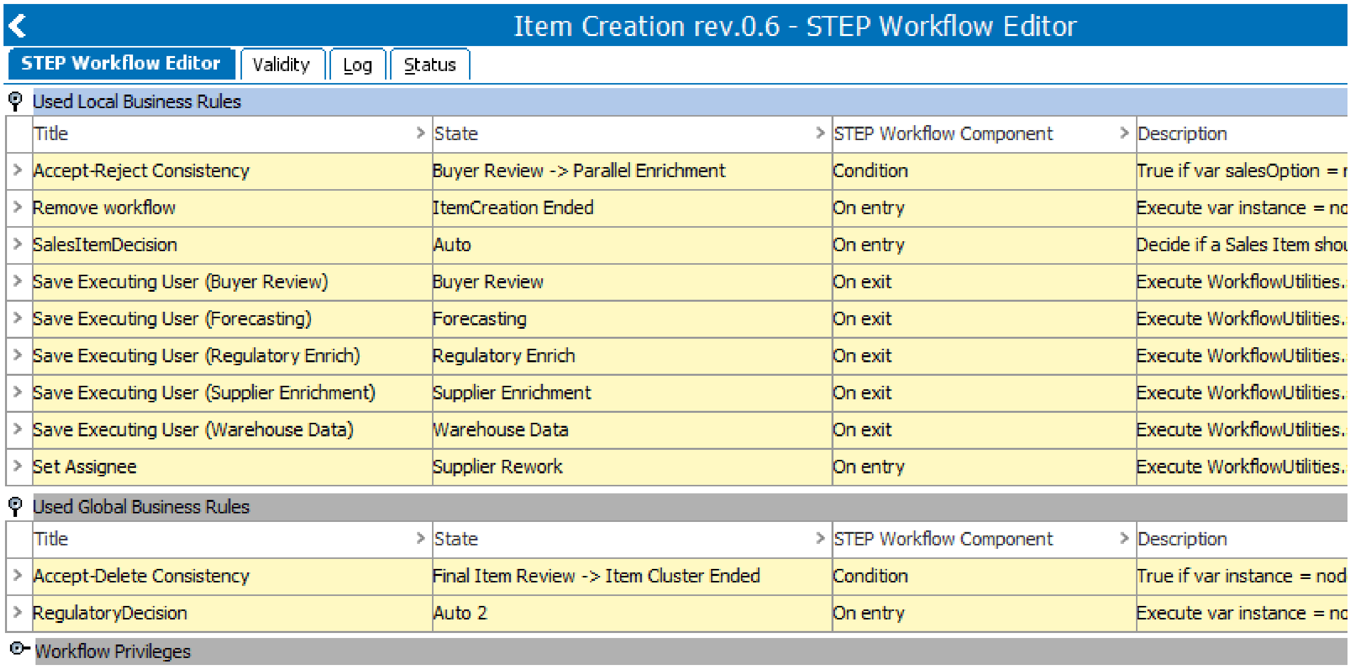
Making a local copy of the business rule, ensures that any changes made to the business rule will only affect the current workflow.

Local business rules created or edited through the STEP Workflow Designer are saved when the Workflow is saved. Global business rules are saved when one saves them.

For more information, see the **Global and Local Business Rules** section of the **Business Rules** documentation.

Workflow Node Editor

The STEP Workflow node editor displays both local and global business rules that are used in the selected workflow.



Used Local Business Rules

Title	State	STEP Workflow Component	Description
Accept-Reject Consistency	Buyer Review -> Parallel Enrichment	Condition	True if var salesOption = r
Remove workflow	ItemCreation Ended	On entry	Execute var instance = no
SalesItemDecision	Auto	On entry	Decide if a Sales Item shou
Save Executing User (Buyer Review)	Buyer Review	On exit	Execute WorkflowUtilities.
Save Executing User (Forecasting)	Forecasting	On exit	Execute WorkflowUtilities.
Save Executing User (Regulatory Enrich)	Regulatory Enrich	On exit	Execute WorkflowUtilities.
Save Executing User (Supplier Enrichment)	Supplier Enrichment	On exit	Execute WorkflowUtilities.
Save Executing User (Warehouse Data)	Warehouse Data	On exit	Execute WorkflowUtilities.
Set Assignee	Supplier Rework	On entry	Execute WorkflowUtilities.

Used Global Business Rules

Title	State	STEP Workflow Component	Description
Accept-Delete Consistency	Final Item Review -> Item Cluster Ended	Condition	True if var instance = nod
RegulatoryDecision	Auto 2	On entry	Execute var instance = no

Workflow Privileges

Additional information on using business rules in STEP Workflows, including detailed configuration instructions to accomplish specific use cases can be found throughout the **Advanced Workflow Topics** section of the **Workflows** material. Specifically, the **Business Rules in Workflows** topic provides an introduction, as well as links to the more detailed topics.

Using Business Rules in an Import Configuration

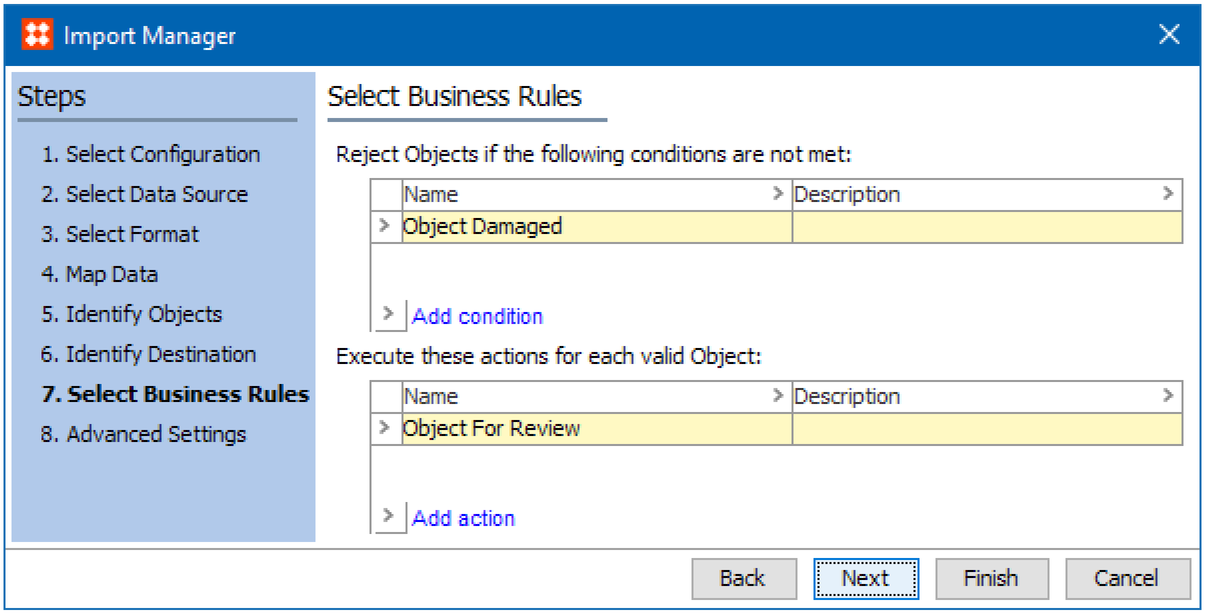
You can apply business rules during an import process. Business conditions enable you to validate data while it is being imported, and business actions enable data changes during import to cause other changes as well.

The number of business rules that are available depend on your system setup. You cannot create new business rules in the Import Manger. In System Setup, you can see if a business rule is used by an import configuration.

To See if a Business Rule is Used by an Import Configuration

- In **System Setup**, expand **Business Rules**, select the relevant business rule, and then click the **Usage** tab. If the business rule is used by an import configuration, you cannot delete the rule.

Business rules are configured on step 7 of the Import Manager wizard, and are saved with the import configuration.



Reject Objects if the following conditions are not met:

Name	Description
Object Damaged	

Add condition

Execute these actions for each valid Object:

Name	Description
Object For Review	

Add action

The Import Process

In the Import Manager, you can apply business rules to products, classifications, entities, and assets. If you apply business rules to other object types, they rules will not be evaluated.

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

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

Validation

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

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

Actions

Actions are executed after validation of conditions. 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.

Errors

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

Detection of Changes

Business rules access the imported objects using the JavaScript "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 2 members: `isUnmodified()` and `getChanges()`.

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

Business Rule Limitations

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

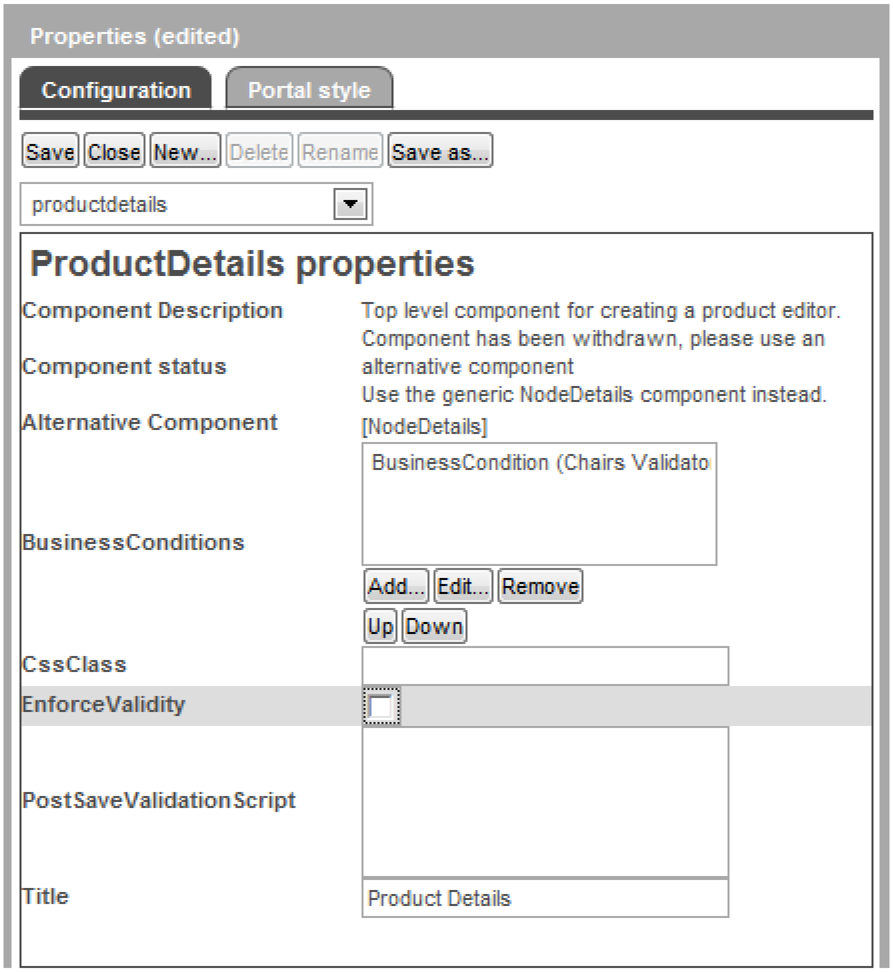
Using Business Conditions in Web UI

Conditions can be validated live in Web UI while you enter data. This means you know immediately if a condition is not fulfilled. You can use conditions to filter List of Values so that it is not possible to select a value that violates the condition.

Configuring a Condition

To configure a condition to be checked in Web UI, enter design mode. Click the Gear Wheel icon to enter design mode. You can add the condition on two levels. On Web UI level, the condition is checked on all screens in Web UI. On screen level, the condition is only checked on the current screen.

The following is an example that shows the configuration of a Web UI screen called ProductDetails. Here you can add, edit and remove conditions for this screen. A similar editor is available in Main properties.

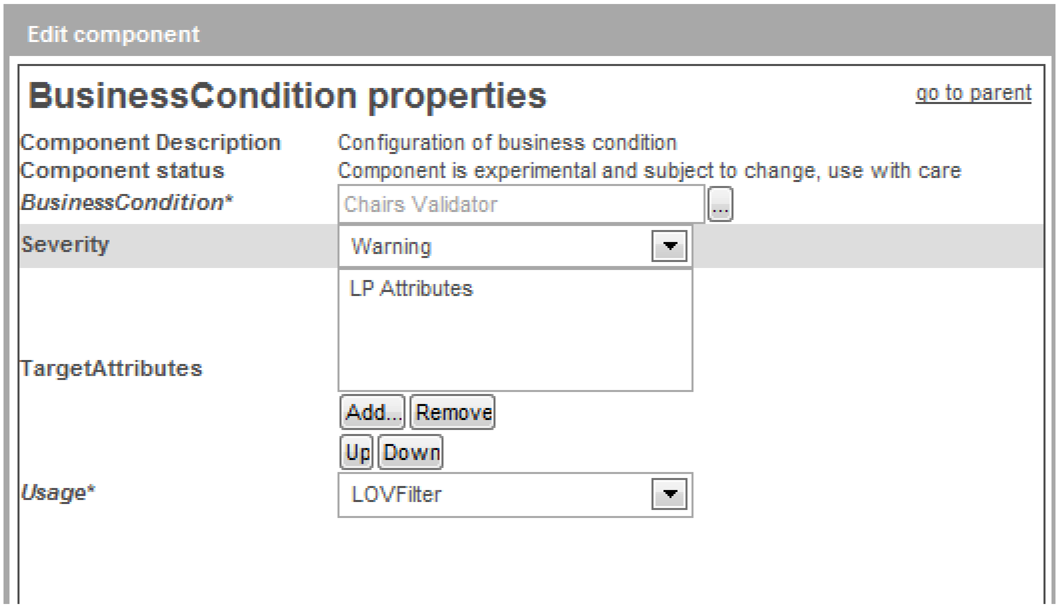


productdetails

ProductDetails properties

Component Description	Top level component for creating a product editor.
Component status	Component has been withdrawn, please use an alternative component
Alternative Component	Use the generic NodeDetails component instead. [NodeDetails]
BusinessConditions	BusinessCondition (Chairs Validato
CssClass	
EnforceValidity	□
PostSaveValidationScript	
Title	Product Details

When you click **Edit**, the following configuration dialog is displayed.



BusinessCondition properties

Component Description	Configuration of business condition
Component status	Component is experimental and subject to change, use with care
*BusinessCondition**	Chairs Validator
Severity	Warning
TargetAttributes	LP Attributes
*Usage**	LOVFilter

You have the following options.

- **Target Attributes**: here you can add the attributes that should be marked as Wrong if the condition is not met.
- **Usage**: can be set to LOVFilter or Validation. If the target attribute is a list of values attribute, usage can be set to LOVFilter and illegal values are filtered away. If validation is selected, the target attribute is marked as wrong if the condition fails.

Note: JavaScript conditions where the current object is bound in cannot be used in Web UI.

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

Using Business Rules with Conditional Attributes

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

Using business rule binding for Conditional Validity

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

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

Here is a JavaScript example for clearing conditionally invalid values.

Required bindings:

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

```
//acquire an iterator - easiest way to deal with Sets
var iter = invalid.iterator();
//iterate over conditionally invalid values
while (iter.hasNext()){
//get the value
var val = iter.next();
//test if there is a value and if it is local
var clean = val.isLocal() && val.getSimpleValue();
//do some chatting to the log
var msg
if (clean)
msg = "Cleaning up";
else
msg = "Ignoring";
msg += " conditionally invalid ";
if (val.isLocal())
msg += "local value";
else
msg += "non-local value";
msg+= val.getAttribute().getID() + " [" + val.getSimpleValue() + "]";
logger.info( msg );
//in case of local value - delete it
if (clean) val.setSimpleValue("");//empty string handled as null and deletes local values
}
```

Using business rule binding of Workflow State

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

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

Required bindings:

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

```
//up front test if the import carries expectation of specific workflow state
if (!state) {
logger.info("No import state provided");
//could have instead return some rejection to say no import if a state is not supplied
return true;
}
var instance, task, msg;
//acquire the workflow instance for the node and workflow
instance = node.getWorkflowInstance(state.getWorkflow());
if (!instance) {
//simply reject since workflow has been terminated or never started
msg = "Rejected : Workflow " + state.getWorkflow().getTitle() + " not started for " +
node.getTitle();
logger.info(msg);
return msg
}
//now look for actual task for node in supplied state
task = instance.getTask(state);
if (!task){
//no task also causes rejection
msg = "Someone might have changed data " + node.getTitle() + " has no task for state " +
state.getID() + " in worflow " + state.getWorkflow().getTitle();
logger.info(msg);
return msg;
}
logger.info('Accepted: we have a task and will now continue with import');
return true;
```

Using Business Rules with GDSN

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

Business action on inbound messages

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

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

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

Business Actions for GDSN

1. In **System Setup**, expand **Business Rules**, and then select the relevant action.
2. On the **Business Rule** tab, in the lower left corner, click **Add New Business Action**.
3. Click the **Edit Operation** icon.
4. In the **Edit Operation** dialog, from the drop-down list, select GDSN.

The following actions are available:

Execute command

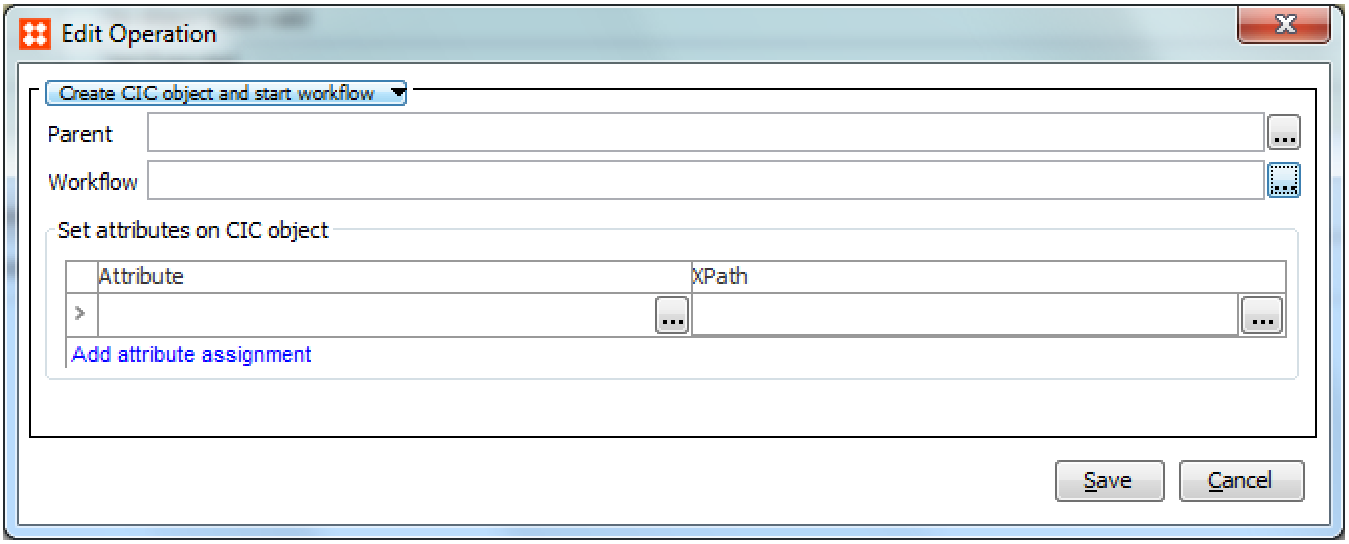
This action sends out a message to a data pool.

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

Field	Description
	parameter tag will contain the GLN of the recipient.
Target Market	Specify the target market the message should be sent in. The target market parameter tag will contain the target market attribute value of the selected target market.

Create CIC Object and Start Workflow

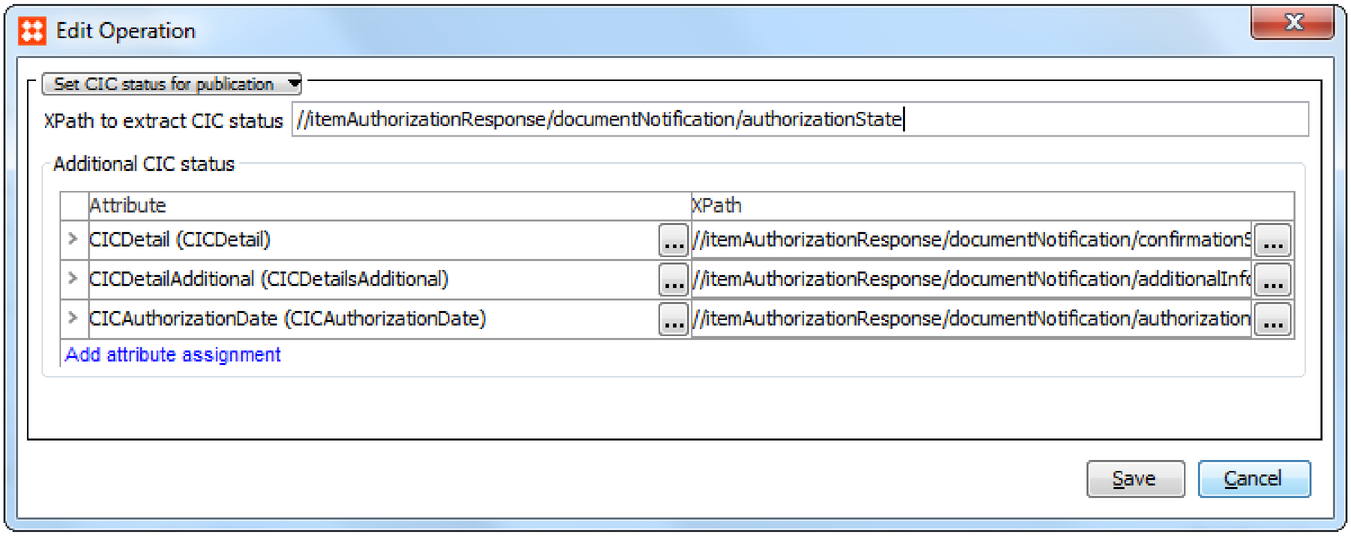
This plug-in creates a CIC entity and starts it in a workflow. It is connected with the product the CIC applies to and with the recipient that sends the CIC. Furthermore, using XPaths, it is possible to place parts of the CIC message in an attribute on the CIC object.



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

Set CIC Status for Publication

Sets the CIC status for a publication to a recipient.



XPath to extract CIC status: //itemAuthorizationResponse/documentNotification/authorizationState

Additional CIC status

Attribute	XPath
CICDetail (CICDetail)	//itemAuthorizationResponse/documentNotification/confirmationS
CICDetailAdditional (CICDetailsAdditional)	//itemAuthorizationResponse/documentNotification/additionalInfo
CICAuthorizationDate (CICAuthorizationDate)	//itemAuthorizationResponse/documentNotification/authorization

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

Update Package Hierarchy Status

Sets the registration status of a package hierarchy link.

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

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

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

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

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

Set Pending File

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

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

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

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

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

Execute Pending File

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

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

Exporting Business Rule Definitions as Comments

Business Rules definitions, including conditions, actions, 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'?>

<STEP-ProductInformation DefinitionsAsComments="true">

<BusinessRule ExportSize="All"/>

</STEP-ProductInformation>
```

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

```
<BusinessRule...>
<!--Definition:[Business rule definition goes here. Removed for brevity.-->
[Remaining configuration would populate here]
</BusinessRule>
```

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

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