STEP Business Rules

Good Practices Guide

VERSION: 4.1

AUTHOR: Stibo Systems Professional Services

CONFIDENTIALITY LEVEL: Internal

Contents

1 Document Information

1.1 Document Control Information

Version	4.0
First release date	
Last updated date	2025-01-24
Author(s)	
Document owner	Director, Solution Delivery, Professional Services
Content owner	SME
Intended audience	Director, Solution Delivery, Professional Services
Document purpose	JavaScript Subject Matter Expert
Document objective	Solution Consultants, Implementation Partners, Customers
Product domain(s)	All domains
Target customer(s)	All customers
Software Version	All supported software versions of STEP
Software license component(s)	STEP SaaS V2

1.2 Document Change History

Version #	Date	Author	Comment
1.0			Initial release.
3.1	2023-09-29	Daniel Baker	Updated document with the most recent corporate branding guidelines.
4.0	2023-10-02	Daniel Baker	Added new section regarding the changes in the use of reflection in JavaScript business.
4.1	2025-01-24	Daniel Baker	Added new section 4 STEP Online Documentation Support. Updated document with the most recent corporate branding guidelines.

2 Document Control

2.1 Document Purpose

This document describes good practice guidelines for writing STEP business rules.

The good practices are compiled from other Stibo documents and experiences found by the Stibo Systems Professional Services team.

2.2 Content of the document

The document contains good practices for the most important topics when developing business rules.

- **Types of business rules**

  We have 3 types of business rules in STEP. Business conditions, Business actions and Business functions. And then we have business rule libraries which are a collection of java script methods.

- **Readability**

  Code that is easy to read is easy to maintain and support. Choose good names for variables and functions. Use proper indentation to make it easy to see where a block of code starts and ends.

- **Use of libraries and business functions**

  Reuse code instead of copy and pasting it.

- **Logging**

  Good log messages make debugging easier, but make sure that it can be turned on and off, so it does not clutter the log files.

- **Exception handling**

  Wrong exception handling can introduce random errors.

- **Performance**

  Keep your business rules small and fast. Business rules that are run during approval or during import may get invoked many times, so here performance is especially important. Business rules used occasionally for bulk updates may be less critical regarding performance.

- **Appendix**

  Tricks and code examples that can help you to write STEP business rules.

3 About Business Rules

Writing complex business rules is recommended to be undertaken by a software developer. Business rules can be maintained without involving Stibo Systems. Business rules are a powerful way to extend the behavior of your STEP system, but it must be used with great responsibility as it can have a significant impact on functionality and performance.

Business rules can be used in imports, approval processes, workflows, Web UI screens, etc. and provide a flexible way to tailor the core functionality in a very precise manner.

Alternatives to business rules are extensions. Either extensions written using the extension API, or custom extensions developed by Stibo Systems. The advantages of extensions are, that they run faster, as they do not have to be compiled at run-time, and that they can be developed using a Java IDE, which offers code completion and syntax checking while you write the code. Extensions should be considered for complex solutions.

3.1 Types of Business Rules

Business rules come in three variants:

	Input	**Output**	**Side Effects Allowed**
Business actions	Current object, current event batch, etc. provided by the context in which the action is executed.	None.	Yes
Business conditions	Current object, current event, etc. provided by the context in which the condition is evaluated.	Boolean result of evaluating the condition and a message for the user.	No
Business functions	Input parameters defined by the function and provided by the functionality evaluating the function. Business functions are basic units of logic that produce an output from an input without affecting the state of data in STEP. Business functions will typically serve as helpers allowing other functionality to delegate a part of their logic to reusable business functions. Business functions are valid on all object types.	Result of evaluating the function.	No

A business library allows users to define JavaScript library functions that can be called from other JavaScript-based business rules.

Side effects are only allowed for business actions, so only business actions can change data in STEP.

4 STEP Online Documentation Support

Important Information

The STEP online help documentation page Performance Recommendations > Business Rule Recommendations provides a great deal of good practice information relating to the configuration of performance analysis of business rules. Subtopics include:

- Business Rule Analysis
- Business Rule Elements to Use
- Business Rule Elements to Avoid

It is highly recommended anyone working with JavaScript within the context of STEP read and understand this information.

5 Readability

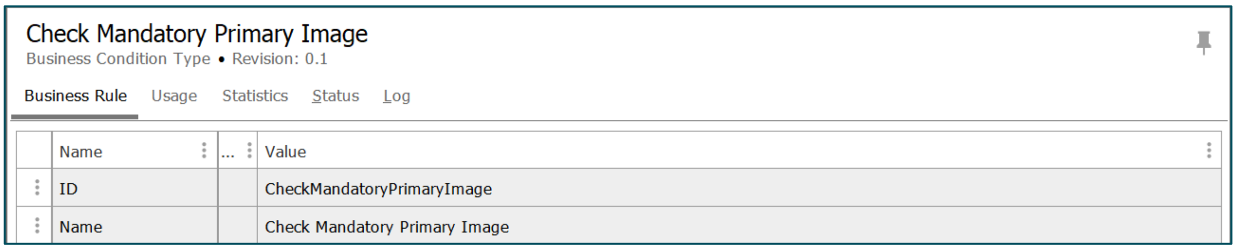
Code which is easy to read is easy to maintain and support. Readability improves when using self-explaining names and proper indentation. Keep all names and descriptions in English, to make them easier to understand for a broader audience.

5.1 Business rules objects in STEP

The business rules objects in STEP should be named according to these good practices:

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

Example:



Check Mandatory Primary Image
Business Condition Type • Revision: 0.1

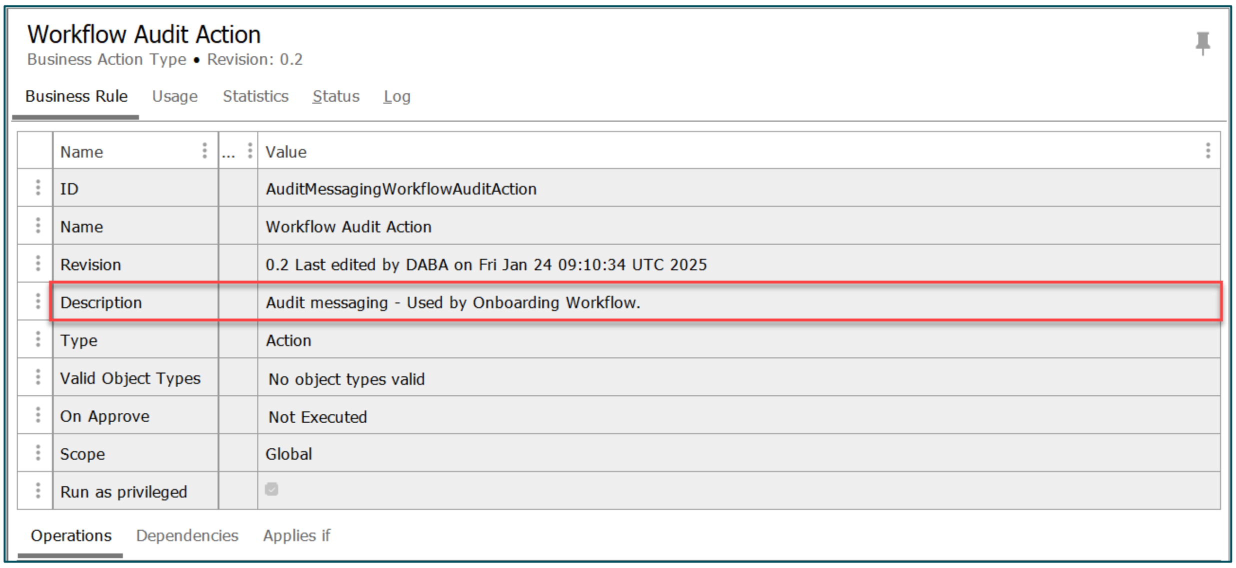
Business Rule | Usage | Statistics | Status | Log

Name	...	Value
ID		CheckMandatoryPrimaryImage
Name		Check Mandatory Primary Image

Figure 1: Naming of business rules.

5.2 Description Field

Every business rule has a description field. It is good practice to write a concise description of what the business rule is doing and in which context it is used.



Workflow Audit Action
Business Action Type • Revision: 0.2

Business Rule | Usage | Statistics | Status | Log

Name	...	Value
ID		AuditMessagingWorkflowAuditAction
Name		Workflow Audit Action
Revision		0.2 Last edited by DABA on Fri Jan 24 09:10:34 UTC 2025
Description		Audit messaging - Used by Onboarding Workflow.
Type		Action
Valid Object Types		No object types valid
On Approve		Not Executed
Scope		Global
Run as privileged		☐

Operations | Dependencies | Applies if

Figure 2: Description field.

5.3 Bind Variables

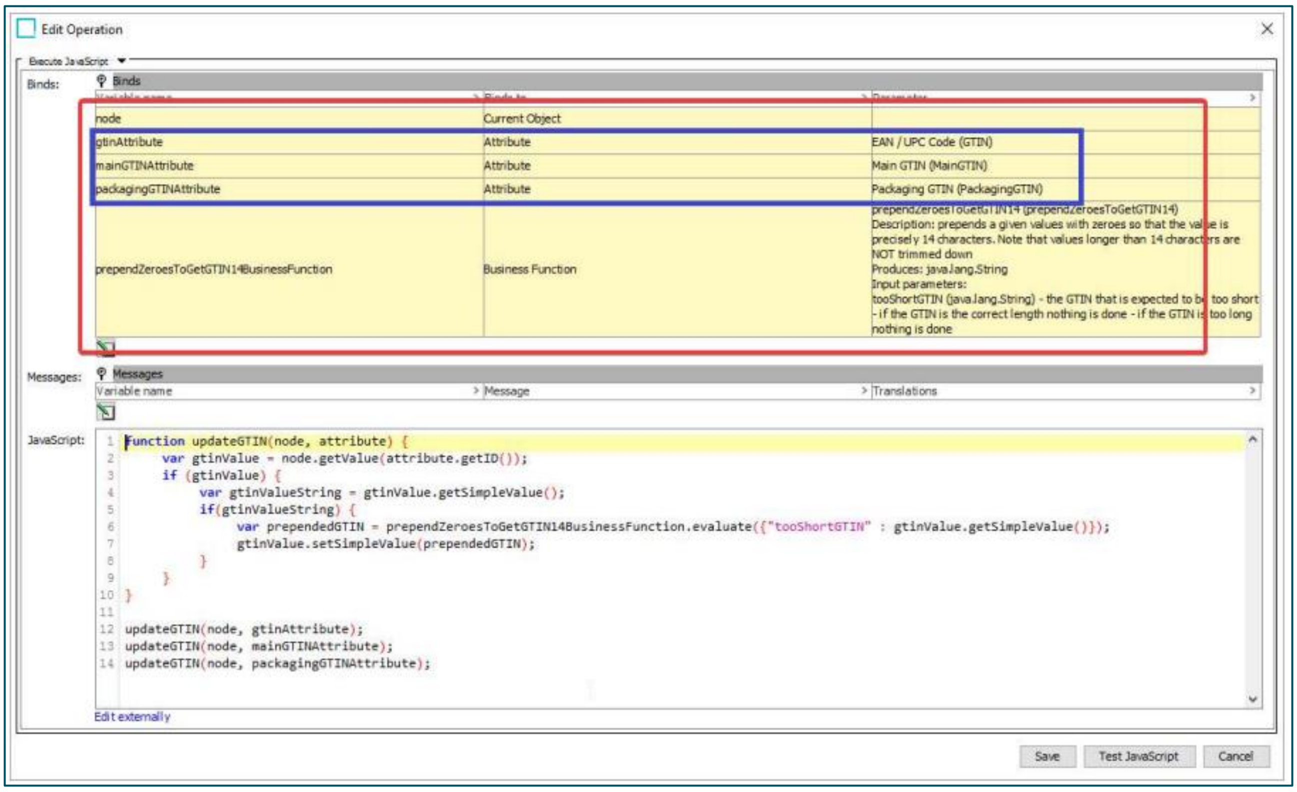
Bind variables should be named in camel case and start with a letter. It should be named, so that the name of the bind variable clearly identifies which bind is being used.

Example:

Bind Variable Name to Use	For Binding to Type
approveContext	Approve Context
currentEventQueue	Current Event Queue
currentEventType	Current Event Type
currentObject	Current Object
currentWorkflow	Current Workflow
<derivedEventTypeID>EventType, e.g., webLinkEventType	Event Type
<OIEPID>EventQueue, e.g., eCommEventQueue	Event Queue
gdsnDataMap	GDSN Data Map
currentObjectID	ID
importChangeInfo	Import Change Info
jsonDocument	JSON Document
Logger	Logger
lookupTableHome	Lookup Table Home
Mailer	Mailer
Manager	STEP Manager
mongoDBContext	MongoDB Context
currentObjectName	Name
<attributeID>Value, unless auto-ID is used, e.g., gtinValue	Attribute Value
workflowParameters	Workflow Parameters

5.3.1 Use binds instead of hardcoding ID's

When creating business rules, it should be considered whether to use the bind functionality over hardcoding ID's. Good practice here is that if the business rule does not need any context specific binds – that is binds that renders the business rule untestable using the standard business rules test functionality – then ALL variables referring to specific should be bound.



Variable name	Binds to	Parameter
node	Current Object	
gtinAttribute	Attribute	EAN / UPC Code (GTIN)
mainGTINAttribute	Attribute	Main GTIN (MainGTIN)
packagingGTINAttribute	Attribute	Packaging GTIN (PackagingGTIN)
prependZeroesToGetGTIN14BusinessFunction	Business Function	prependZeroesToGetGTIN14 (prependZeroesToGetGTIN14) Description: prepends a given values with zeroes so that the value is precisely 14 characters. Note that values longer than 14 characters are NOT trimmed down Produces: java.lang.String Input parameters: tooShortGTIN (java.lang.String) - the GTIN that is expected to be too short - if the GTIN is the correct length nothing is done - if the GTIN is too long nothing is done

Variable name	Message	Translations

```
function updateGTIN(node, attribute) {
     var gtinValue = node.getValue(attribute.getID());
     if (gtinValue) {
          var gtinValueString = gtinValue.getSimpleValue();
          if(gtinValueString) {
               var prependedGTIN = prependZeroesToGetGTIN14BusinessFunction.evaluate({"tooShortGTIN" : gtinValue.getSimpleValue()});
               gtinValue.setSimpleValue(prependedGTIN);
          }
     }
}

updateGTIN(node, gtinAttribute);
updateGTIN(node, mainGTINAttribute);
updateGTIN(node, packagingGTINAttribute);
```

Figure 3: Use of binds.

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

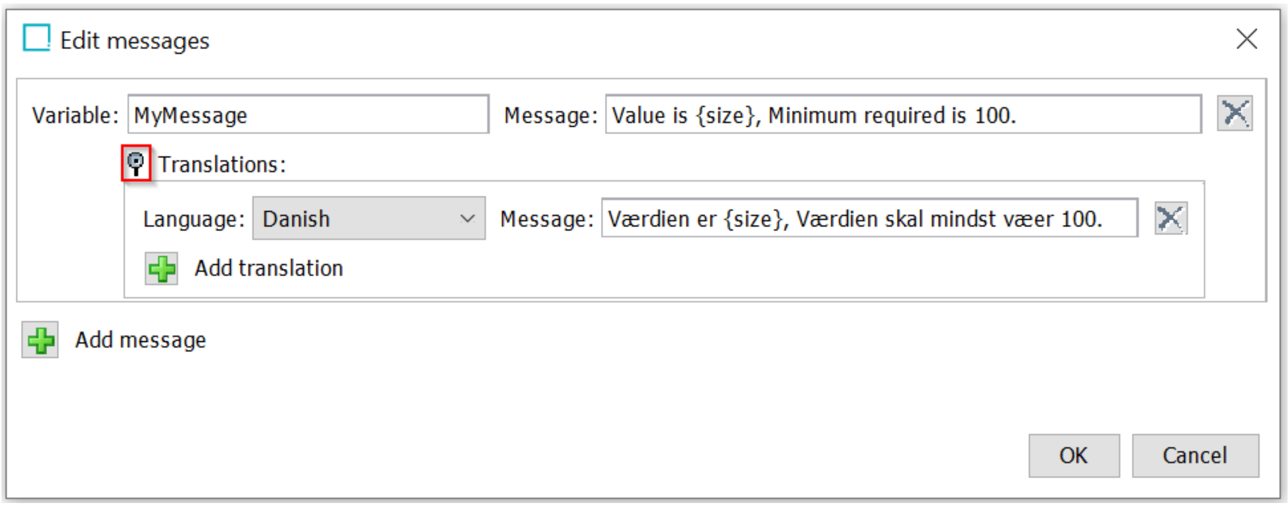
5.4 Variable and function names

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

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

5.5 Error Messages

When defining error messages for business rules, then always make sure that one of the messages is in English to make them easier to understand for a broader audience.


Edit messages

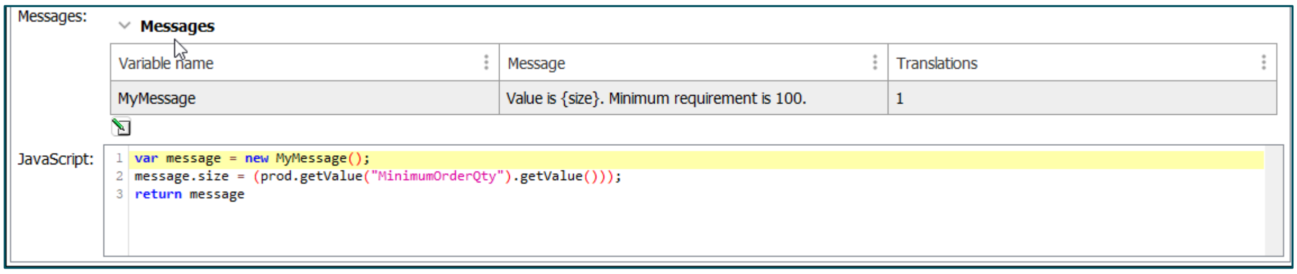
Variable: MyMessage Message: Value is {size}, Minimum required is 100.

Translations:

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

Figure 4: Translated error messages.

Use variables (in this case “size”) to make the error messages meaningful.


Messages:

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
```

Figure 5: Use of variables in error messages.

5.6 Code layout

Keep lines shorter than 80 characters. Use the indentation (tab) which is suggested by the workbench editor.

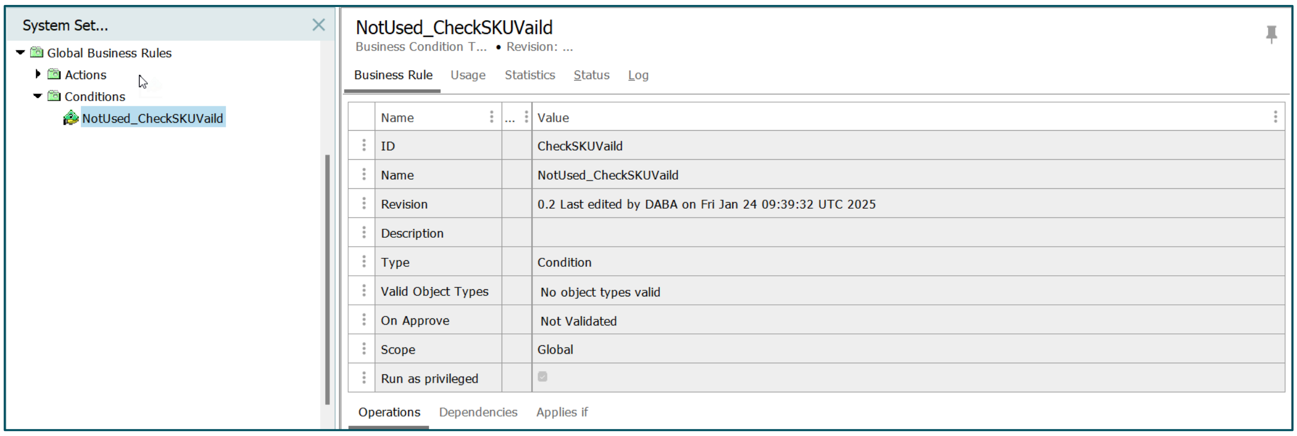
Examples:

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

5.7 House keeping

It is good practice to clearly mark unused business rules as being obsolete, by adding a NotUsed prefix to the name, if you do not want to delete them.

It is also good practice to clearly mark temporary business rule with a prefix like Test.



Name	...	Value
ID		CheckSKUVaild
Name		NotUsed_CheckSKUVaild
Revision		0.2 Last edited by DABA on Fri Jan 24 09:39:32 UTC 2025
Description		
Type		Condition
Valid Object Types		No object types valid
On Approve		Not Validated
Scope		Global
Run as privileged		☐

Figure 6: Obsolete business rule prefixed by NotUsed.

6 Use of Libraries

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

- WorkflowUtilities
- PackagingUtilities
- NumberFormattingUtilities
- ApprovalUtilities

6.1 Avoid large business rule libraries

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

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

Important

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

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

The cache reduces the problem, but it is still good practice to keep business rules libraries small, to improve performance.

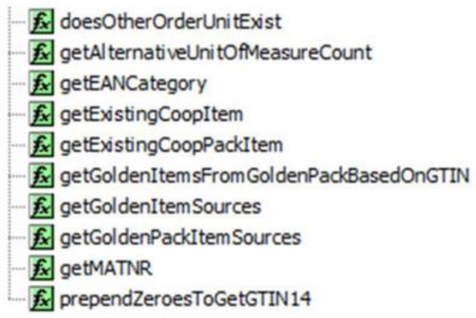
An alternative to business rule libraries is business functions. A business function is one specific function, so unlike a library, you will not get additional unused functions included and compiled.

6.2 Business Functions

Business functions are a newer construct available from the release of STEP 9. The business function functionality was originally intended to be used for certain areas of STEP where business actions or conditions was insufficient due to their lack of returning a result.

As business functions can be utilized as part of a business action or condition they also prove quite useful as a 1 function library – often more so than a library as the function itself can be named to © Stibo Systems STEP Performance Good Practices 13/30 make sense in what it does and in that the entire function is utilized every time – as opposed to a library where often only parts of it is used at a given situation. On the downside a business function cannot be used to alter data stored in the database – much like a business condition.

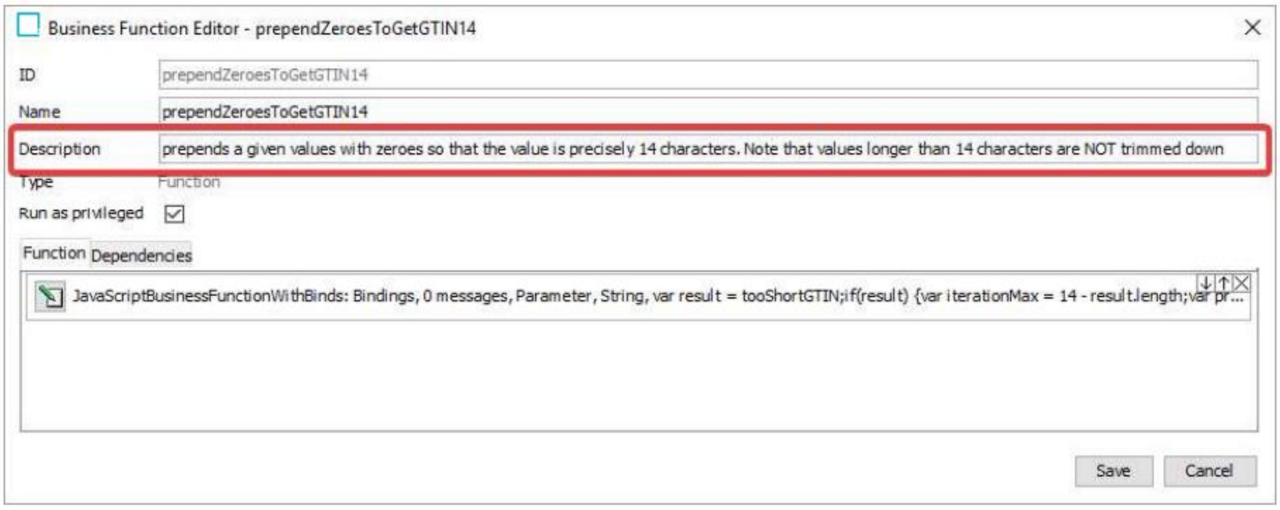
A business function can be thought of as a JavaScript method with input and output. To that respect the naming of the business function should be carefully considered so that the business function communicates as best possible the functionality it offers.



doesOtherOrderUnitExist
getAlternativeUnitOfMeasureCount
getEANCategory
getExistingCoopItem
getExistingCoopPackItem
getGoldenItemsFromGoldenPackBasedOnGTIN
getGoldenItemSources
getGoldenPackItemSources
getMATNR
prependZeroesToGetGTIN14

Figure 7: Examples of named business functions.

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



Business Function Editor - prependZeroesToGetGTIN14

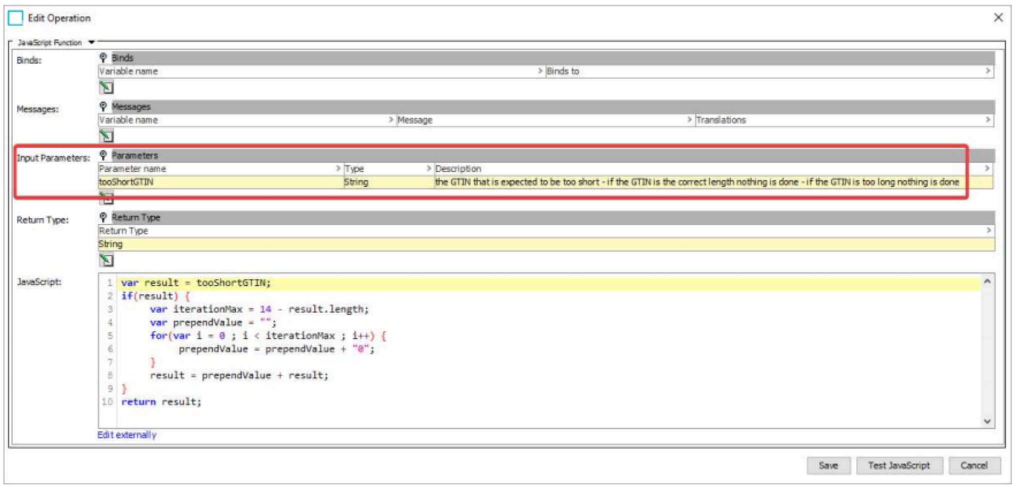
ID	prependZeroesToGetGTIN14
Name	prependZeroesToGetGTIN14
Description	prepends a given values with zeroes so that the value is precisely 14 characters. Note that values longer than 14 characters are NOT trimmed down
Type	Function
Run as privileged	☑

Function Dependencies

```
JavaScriptBusinessFunctionWithBinds: Bindings, 0 messages, Parameter, String, var result = tooShortGTIN;if(result) {var iterationMax = 14 - result.length;var pr...
```

Figure 8: The Description field should be filled out for the business function describing the functionality in more detail as well as any assumptions or limitations that the business function might have.

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



Variable name	Binds to

Variable name	Message	Translations

Parameter name	Type	Description
tooShortGTIN	String	the GTIN that is expected to be too short - if the GTIN is the correct length nothing is done - if the GTIN is too long nothing is done

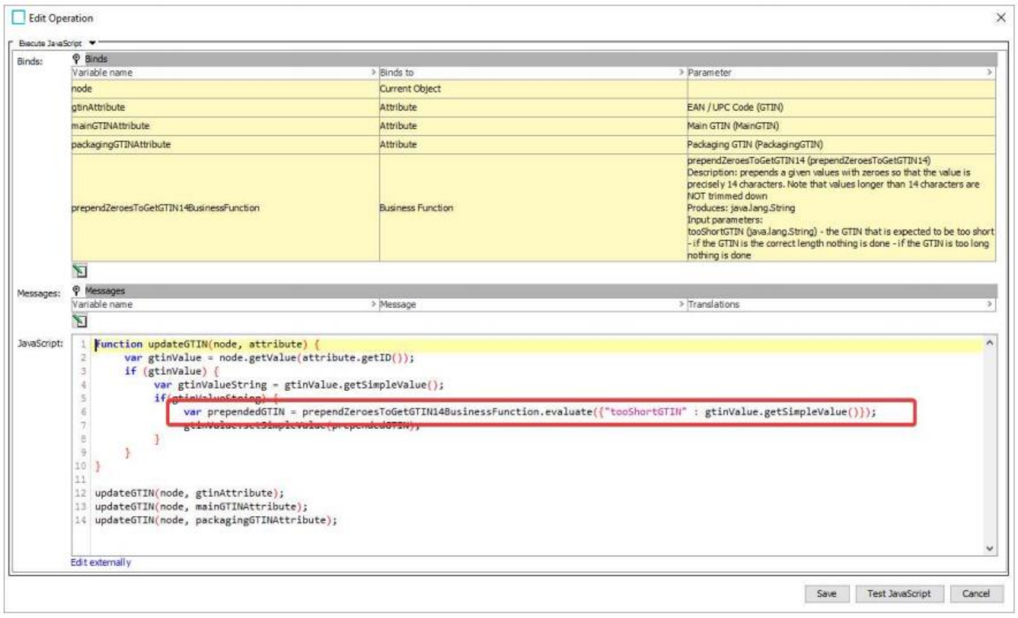
Return Type
String

```
var result = tooShortGTIN;
if(result) {
    var iterationMax = 14 - result.length;
    var prependValue = "";
    for(var i = 0 ; i < iterationMax ; i++) {
        prependValue = prependValue + "0";
    }
    result = prependValue + result;
}
return result;
```

Figure 9: Adding description to parameters.

The business function code should adhere to normal good coding practice with comments where necessary. Using JavaScript functions or binding other business functions to structure code are considered good practice just as with any other business rule.

Calling the business function is done by parsing a Json object containing the input parameters to the evaluate function.



Variable name	Binds to	Parameter
node	Current Object	
gtinAttribute	Attribute	EAN / UPC Code (GTIN)
mainGTINAttribute	Attribute	Main GTIN (MainGTIN)
packagingGTINAttribute	Attribute	Packaging GTIN (PackagingGTIN)
prependZeroesToGetGTIN14BusinessFunction	Business Function	prependZeroesToGetGTIN14 (prependZeroesToGetGTIN14) Description: prepends a given values with zeroes so that the value is precisely 14 characters. Note that values longer than 14 characters are NOT trimmed down Produces: java.lang.String Input parameters: tooShortGTIN (java.lang.String) - the GTIN that is expected to be too short - if the GTIN is the correct length nothing is done - if the GTIN is too long nothing is done

Variable name	Message	Translations

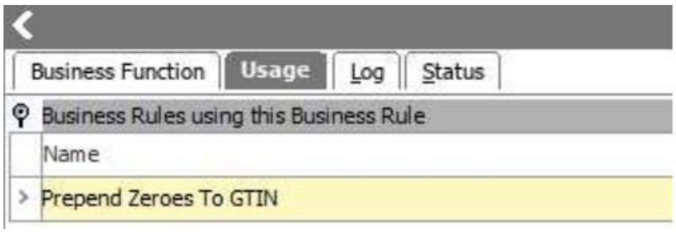
```
function updateGTIN(node, attribute) {
    var gtinValue = node.getValue(attribute.getID());
    if (gtinValue) {
        var gtinValueString = gtinValue.getSimpleValue();
        if(gtinValueString) {
            var prependedGTIN = prependZeroesToGetGTIN14BusinessFunction.evaluate({"tooShortGTIN" : gtinValue.getSimpleValue()});
            gtinValue.setSimpleValue(prependedGTIN);
        }
    }
}

updateGTIN(node, gtinAttribute);
updateGTIN(node, mainGTINAttribute);
updateGTIN(node, packagingGTINAttribute);
```

Figure 10: Using a business function.

The conclusions for business functions are that for obfuscating code it is a great tool as the user can use the function as a black box like when using libraries but unlike libraries it is easy to see from the user what the business function offers. On the downside the business function runs in non-transactional scope and thus cannot alter the database in any way.

One additional advantage of using the business function over the library is that the business function offers "usage" information about which business rules that has a bind to it.



Business Rules using this Business Rule

Name
Prepend Zeroes To GTIN

Figure 11: The business function offers information about which business rules that currently has a bind to it.

7 Logging

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

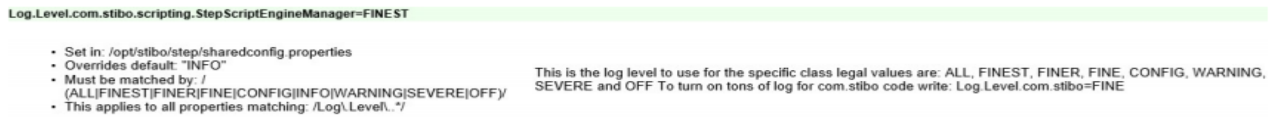
It is therefore recommended to configure the business rule logging to avoid unnecessary logging of business rule details.

The amount of logging can be controlled globally (for all business rules) using the configuration property in sharedconfig.properties.

```
Log.Level.com.stibo.scripting.StepScriptEngineManager=INFO
```

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

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


Log.Level.com.stibo.scripting.StepScriptEngineManager=FINEST

- Set in: /opt/stibo/step/sharedconfig.properties
- Overrides default: "INFO"
- Must be matched by: /(ALL|FINEST|FINER|FINE|CONFIG|INFO|WARNING|SEVERE|OFF)/
- This applies to all properties matching: /Log\.Level\..*/

This is the log level to use for the specific class legal values are: ALL, FINEST, FINER, FINE, CONFIG, WARNING, SEVERE and OFF To turn on tons of log for com.stibo code write: Log.Level.com.stibo=FINE

Figure 12: Viewing configuration in STEP – System Administration UI.

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

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

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

For example:

```
//Debug 'flag' REMEMBER to turn 'false' when you are done
var isDebug = false;
//Function to handle whetever logging of debug information should occur or not
function logDebug(message) {
 if(isDebug) {logger.info(message)}
}

...
logDebug("Here’s a message for the log file")
...
```

8 Exception Handling

If an error occurs during approval, an exception is thrown from the domain layer. If this exception is cached in a business rule but not re-thrown, it will not reach the exception approval handler. In this case you can end up with objects that can be inconsistent where some parts are approved, and others are not. This behavior also has a negative effect on the performance of the business rule.

You need to be careful when doing exception handling to avoid this behavior when using "try-catch" in business rules.

8.1 What is NOT valid

Exception is not re-thrown and therefore will not reach the exception approval handler which may cause inconsistent objects (some parts are approved, and others are not).

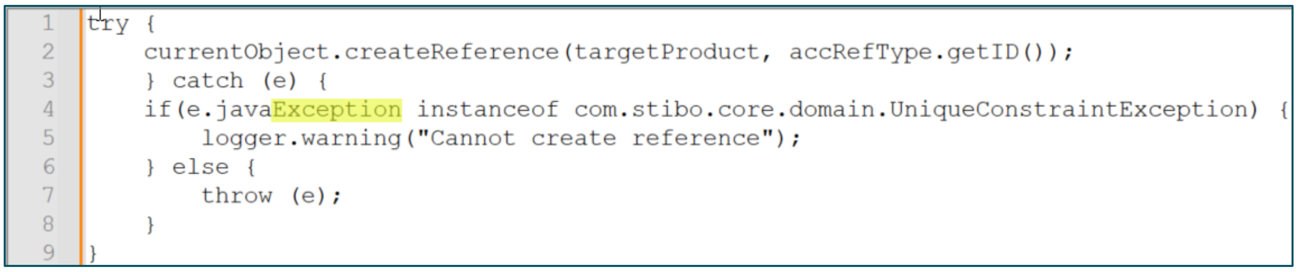
```
try
{
      node.approve();
} catch (exception)
{
}
```

8.2 What IS required

Exception is re-thrown and therefore will reach the exception approval handler avoiding inconsistent objects.

```
try
{
      node.approve();
} catch (exception) {
      throw exception;
}
```

An example:



```
try {
    currentObject.createReference(targetProduct, accRefType.getID());
    } catch (e) {
    if(e.javaException instanceof com.stibo.core.domain.UniqueConstraintException) {
        logger.warning("Cannot create reference");
    } else {
        throw (e);
    }
}
```

Figure 13: Example of error handling.

9 Use of reflection in JavaScript business

Use of reflection in JavaScript business rules will be disabled!

In JavaScript business rules, reflection has been used to access non-public methods in the API. Since reflection can be a security risk and can also potentially lead to performance issues and unexpected issues during upgrade, it will be disabled in a future release. With the 2023.3 (11.3) release, a new configuration health check ‘Reflection usage in business rules’ identifies business rules that use reflection. To prepare for restricted access of reflection usage, rewrite the reported business rules to use publicly available methods.

If the needed functionality is not public, use the customer support portal to create an enhancement request to make the functionality public.

10 Performance

Some business rules are invoked frequently, like rules running during approval and import. To preserve the performance of the STEP system, it is important to make sure that your business rules perform well.

A business rule runs in a single transaction, so it is also important that it finished within a short time, to avoid optimistic locking errors.

10.1 Investigating business rule performance

There are several ways to analyze and monitor business rules.

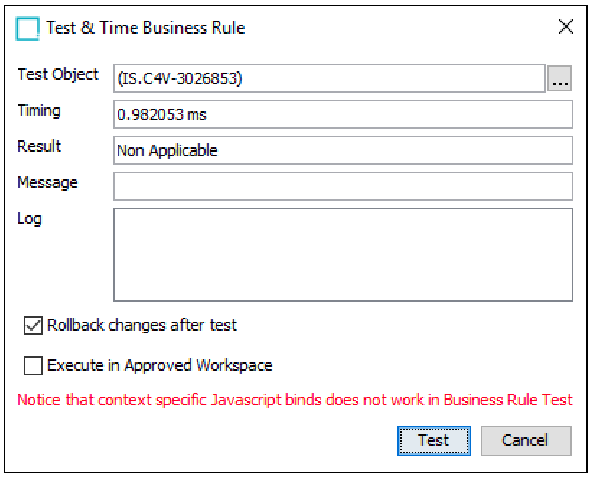
10.1.1 Workbench business rule test menu

The Workbench business rule test menu is typically used during development, and performance can be taken from the tests. This method of business rule analysis gives a first indication of the performance of the business rule for a certain item.

- Testing: Right click the business rule and select Test Business Rule.
- Run the business rule a couple of times separate against objects (e.g., products) of which you are certain the business rule will fail or pass.
- Then analyze the timing of the business rule to see its performance.

When a long running business rule is identified, use the test menu to evaluate the performance of the business rule.

For example: The following screenshot shows that the business rule took about 0.54 milliseconds to complete the business rule for item `IS.C4V-3026853`. Nevertheless, be aware that it might take longer or shorter for other Items.


Test & Time Business Rule

Test Object	(IS.C4V-3026853)
Timing	0.982053 ms
Result	Non Applicable
Message	
Log	

☑ Rollback changes after test

☐ Execute in Approved Workspace

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

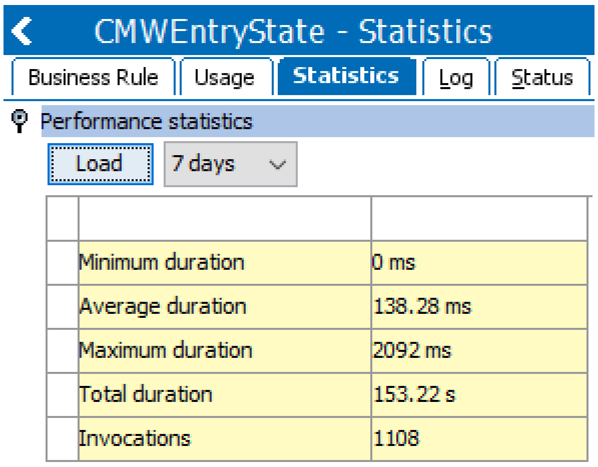
Figure 14: Testing business rule from the Workbench UI.

10.1.2 Workbench business rule statistics tab

Secondly, the Workbench provides a business rule statistics tab to see the performance of the business rule over time.

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

For example: The following screenshot shows the same business rule which was invoked more than 1000 times during the last 7 days. That average duration was about 138 ms.



Minimum duration	0 ms
Average duration	138.28 ms
Maximum duration	2092 ms
Total duration	153.22 s
Invocations	1108

Figure 15: Viewing business rule statistics from the Workbench UI.

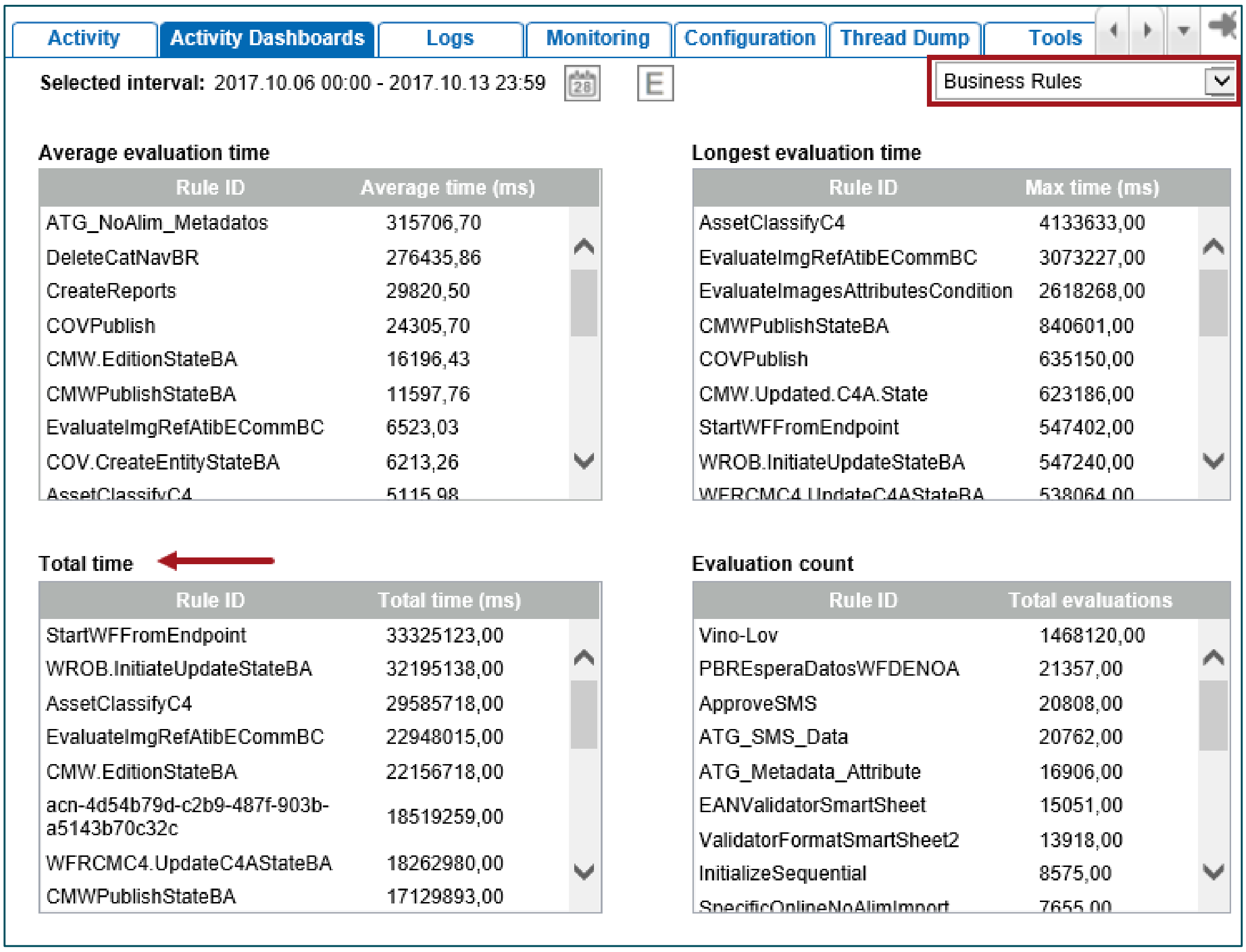
It is possible to click on the maximum duration of about 2092 milliseconds. This shows which item the business rule took longest to execute.

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

10.1.3 Admin Portal activity dashboard for business rules

The Admin Portal provides the possibility to track and trace business rules performance over a given period. The dashboard for business rules is available in STEP Admin Portal > Activity Dashboard > Business Rules.

For example:



Selected interval: 2017.10.06 00:00 - 2017.10.13 23:59

Business Rules

Average evaluation time

Rule ID	Average time (ms)
ATG_NoAlim_Metadatos	315706,70
DeleteCatNavBR	276435,86
CreateReports	29820,50
COVPublish	24305,70
CMW.EditionStateBA	16196,43
CMWPublishStateBA	11597,76
EvaluateImgRefAtibECommBC	6523,03
COV.CreateEntityStateBA	6213,26
AssetClassifyC4	5115,98

Longest evaluation time

Rule ID	Max time (ms)
AssetClassifyC4	4133633,00
EvaluateImgRefAtibECommBC	3073227,00
EvaluateImagesAttributesCondition	2618268,00
CMWPublishStateBA	840601,00
COVPublish	635150,00
CMW.Updated.C4A.State	623186,00
StartWFFromEndpoint	547402,00
WROB.InitiateUpdateStateBA	547240,00
WFRCMC4.UpdateC4AStateBA	538064,00

Total time

Rule ID	Total time (ms)
StartWFFromEndpoint	33325123,00
WROB.InitiateUpdateStateBA	32195138,00
AssetClassifyC4	29585718,00
EvaluateImgRefAtibECommBC	22948015,00
CMW.EditionStateBA	22156718,00
acn-4d54b79d-c2b9-487f-903b-a5143b70c32c	18519259,00
WFRCMC4.UpdateC4AStateBA	18262980,00
CMWPublishStateBA	17129893,00

Evaluation count

Rule ID	Total evaluations
Vino-Lov	1468120,00
PBREsperaDatosWFDENOA	21357,00
ApproveSMS	20808,00
ATG_SMS_Data	20762,00
ATG_Metadata_Attribute	16906,00
EANValidatorSmartSheet	15051,00
ValidatorFormatSmartSheet2	13918,00
InitializeSequential	8575,00
SpecificOnlineNoAlimImport	7655,00

Figure 16: Viewing activity dashboard in STEP – System Administration UI.

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

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

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

10.1.4 Admin Portal business rule tracing

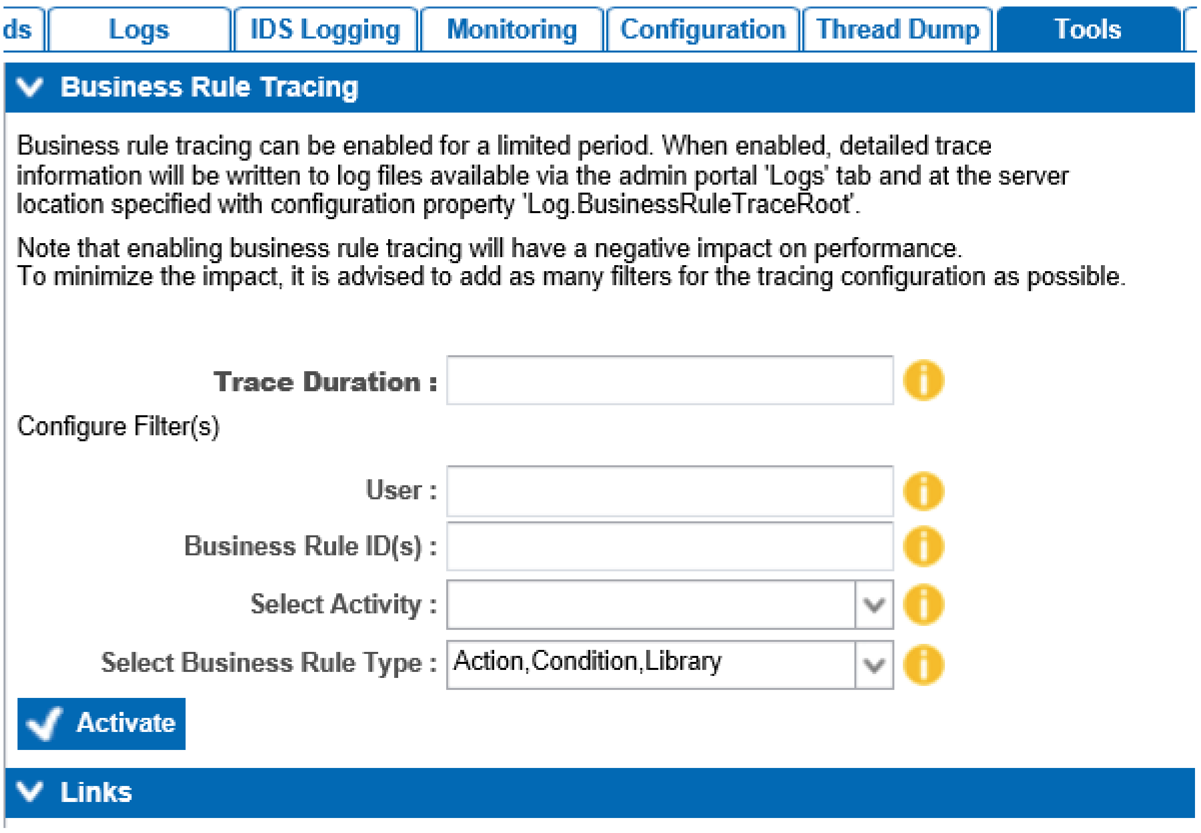
There is an option in the Admin Portal to trace business rules. The functionality of the Business Rule Tracing section of the Tools tab is described within the interface itself.

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

Warning

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

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



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

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

Trace Duration :

Configure Filter(s)

User :

Business Rule ID(s) :

Select Activity :

Select Business Rule Type : Action,Condition,Library

Figure 17: Enabling tracing in STEP – System Administration UI

Note

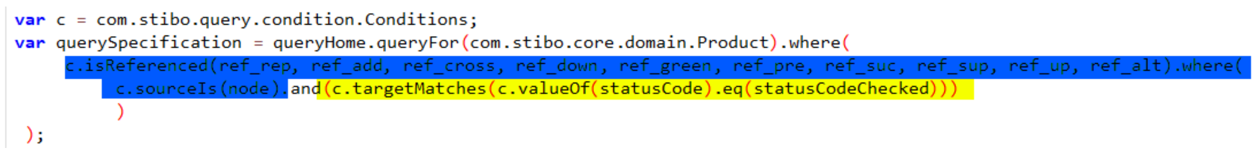
Once tracing has been activated, the relevant business rule(s) must be triggered in STEP within the time frame defined in the Trace Duration parameter so that the rule is active for tracing. Furthermore, if the system is stopped or restarted, any tracing that was in progress will also be stopped.

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

10.1.5 Important considerations for Queries performances

This is especially important to sequence conditions from the “more” precise to the “less” one.

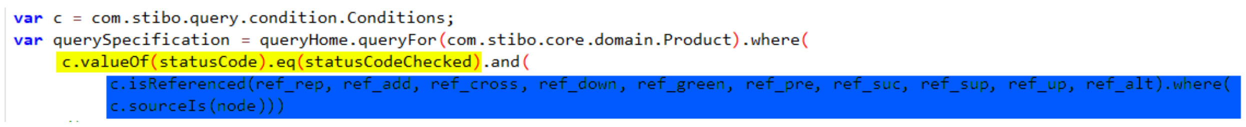
For example, query below uses two conditions, the blue one checking “referenced by” links, and the yellow one looking at an attribute value.



```
var c = com.stibo.query.condition.Conditions;
var querySpecification = queryHome.queryFor(com.stibo.core.domain.Product).where(
      c.isReferenced(ref_rep, ref_add, ref_cross, ref_down, ref_green, ref_pre, ref_suc, ref_sup, ref_up, ref_alt).where(
          c.sourceIs(node).and(c.targetMatches(c.valueOf(statusCode).eq(statusCodeChecked)))
          )
 );
```

Query execution time is around **1500 ms**.

When switching conditions’ sequences:



```
var c = com.stibo.query.condition.Conditions;
var querySpecification = queryHome.queryFor(com.stibo.core.domain.Product).where(
     c.valueOf(statusCode).eq(statusCodeChecked).and(
          c.isReferenced(ref_rep, ref_add, ref_cross, ref_down, ref_green, ref_pre, ref_suc, ref_sup, ref_up, ref_alt).where(
          c.sourceIs(node)))
```

For the same result, Query execution time is less than **10 ms !**

By simply switching conditions sequence, performances are improved by a ratio of **150 !**

Always consider using the more restrictive conditions first. Please also note that in some cases, it is better to have less conditions in the query and then browse query’s results for additional operations.

10.2 Good Practices

10.2.1 Keep business rule transactions small

Business actions have a transaction, which allows you to write data to STEP.

However, business actions with long transactions will degrade the performance. Furthermore, STEP runs with optimistic locking policy. The longer the transaction, the larger is the probability of introducing an optimistic locking failure when running the business action simultaneously.

10.2.2 Avoid the function GetChildren with many nodes

The getChildren method has been replaced by the queryChildren method. It is recommended that all instances of the getChildren method are replaced. The reasoning for this is that business rules using calls "getChildren" on a substantial number of children may cause memory problems. The problem is that the "getChildren" uses an unsafe call that will read all children. It should be changed into using "queryChildren".

10.2.3 Use arrays instead of multiple read calls

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

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

10.2.4 Consider In-Memory for business rules

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

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

11 Working with business rules outside of STEP

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

For more information about the STEP GIT integration, see “Version Control System Integration” in the online documentation.

10.1 Creating business rules outside STEP

Even if it is possible to create a business rule outside of STEP, and load it into STEP, it is easier to setup dependencies, binds, valid object types and so on in STEP, where you can browse for the information, you want to use.

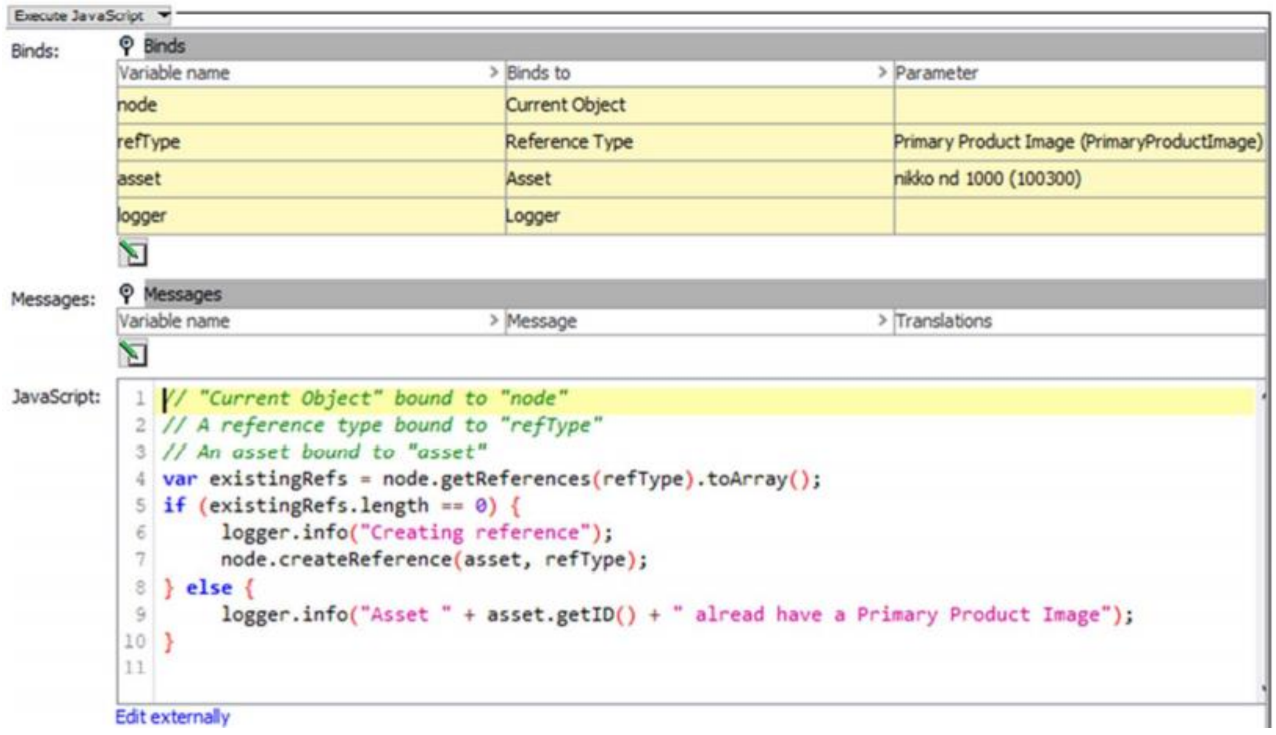
But after the initial creation, then you can easily maintain the java script code externally.

10.2 Maintain business rules outside STEP

Business rules can be exported as *.js files that can be edited outside of STEP and imported back into STEP to update a business rule.

The code part is easy to maintain in an external editor, but the binds are best maintained in STEP, where you have the help of drop-down lists, and selectors to choose the objects.

This business rule called Test1 looks like this in the STEP workbench:



Binds

Variable name	Binds to	Parameter
node	Current Object	
refType	Reference Type	Primary Product Image (PrimaryProductImage)
asset	Asset	nikko nd 1000 (100300)
logger	Logger	

Messages

Variable name	Message	Translations

```
// "Current Object" bound to "node"
// A reference type bound to "refType"
// An asset bound to "asset"
var existingRefs = node.getReferences(refType).toArray();
if (existingRefs.length == 0) {
     logger.info("Creating reference");
     node.createReference(asset, refType);
} else {
     logger.info("Asset " + asset.getID() + " alread have a Primary Product Image");
}
```

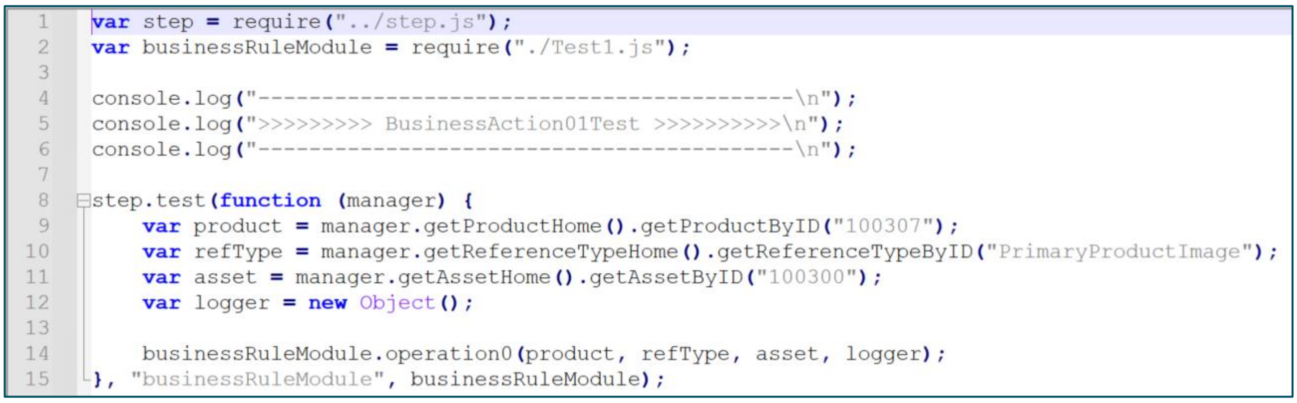
Figure 18: Business rule Test1 in the Workbench.

And it looks like this when exported as a *.js file:

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

10.3 Test business rules outside STEP

The ability to test business rules outside of STEP allows you to do automatic regression testing of your business rules. Within a directory you have your exported business rule as a *.js file together with another *.js file which executes the test.



```
var step = require("../step.js");
var businessRuleModule = require("./Test1.js");

console.log("------------------------------------------\n");
console.log(">>>>>>>>> BusinessAction01Test >>>>>>>>>>\n");
console.log("------------------------------------------\n");

step.test(function (manager) {
    var product = manager.getProductHome().getProductByID("100307");
    var refType = manager.getReferenceTypeHome().getReferenceTypeByID("PrimaryProductImage");
    var asset = manager.getAssetHome().getAssetByID("100300");
    var logger = new Object();

    businessRuleModule.operation0(product, refType, asset, logger);
}, "businessRuleModule", businessRuleModule);
```

Figure 19: Sample test file.

The test file will setup the data used by your business rule, and within the STEP.test method you can test if the business rule created the correct objects in step. All changes on the STEP server will be rolled back when the test completes.

12 Appendix A

This appendix contains tricks and code examples that can help the user with writing STEP business rules. As the format of the examples are not fit for a printed-out format it is suggested to simply copy the code and paste into suitable editor before reading it.

12.1 Comparison of objects

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

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

Note

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

A good example here is the String objects: 2 Strings that from a human perspective reads they cannot be compared equal using the "==" operator IF one String stems from JavaScript and the other from Java.

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

Will always produce false no matter if the value on the product was "xyz".

For String it is therefore recommended to always use the .equals() method available on both JavaScript and Java:

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

11.2 Looping children of an object

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

Example 1: Looping through children using an anonymous inline implementation of the QueryConsumers consume method:

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

Example 2: Looping through children using an explicit implementation of the QueryConsumers consume method (printTitle in this case)

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

var childrenQuery = node.queryChildren();
childrenQuery.forEach(printTitle);
```

11.3 Searches

It is also possible to do searches as part of the scripting API however one should be careful about using these as they can easily lead to exceptionally long running times of the scripts.

Also notice that the below examples are not all valid for all versions of STEP.

11.3.1 Example 1: Perform single attribute search in products (**pre STEP 9**)

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

Note

Please observe that no matter what is chosen as maxResult the result list cannot be more than 100 – the query will simply be cut of at this time.

11.3.2 Example 2: Perform single attribute search in products (**STEP 9 and onwards**)

```
function breakingQueryConsumer(node) {
  logger.info(node.getTitle());
  logger.info("Breaking...");
  return false; // break the "forEach" on the query
}
function continuingQueryConsumer(node) {
  logger.info(node.getTitle());
  logger.info("Continuing...");
  return true; // continue the "forEach" on the query
}
var singleAttributeQueryHome =
manager.getHome(com.stibo.core.domain.singleattributequery.SingleAttributeQueryH
ome);
var conditions = new
com.stibo.core.domain.singleattributequery.SingleAttributeQueryHome.SingleAttrib
uteQuerySpecification(com.stibo.core.domain.Product, descriptionAttribute,
"test");
var query = singleAttributeQueryHome.querySingleAttribute(conditions);
query.forEach(breakingQueryConsumer);
query.forEach(continuingQueryConsumer);
```

11.3.3 Example 3: using the queryAPI (**STEP 9 and onwards**)

Observe that to use the queryAPI the following addon needs to be installed.

spot --apply=to:query/7.0/latest.spr

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

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

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

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

// query where both conditions are met (and).
var querySpecification =
queryHome.queryFor(com.stibo.core.domain.Product).where(isBelowCondition.and(has
ValueTestCondition));

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

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

11.3.4 Example 4: using the queryAPI (**STEP 9 and onwards**)

Observe that to use the queryAPI the following addon needs to be installed.

spot --apply=to:query/7.0/latest.spr

```
function searchProductByAttributeValueAndObjectType(manager, objectTypeID,
attrID, value) {
var conditions = com.stibo.query.condition.Conditions;

var hasValueTestCondition =
conditions.valueOf(manager.getAttributeHome().getAttributeByID("" +
attrID)).eq("" + value);
var hasObjectTypeCondition =
conditions.objectType(manager.getObjectTypeHome().getObjectTypeByID(objectTypeID
));

var queryHome = manager.getHome(com.stibo.query.home.QueryHome);
var querySpecification =
queryHome.queryFor(com.stibo.core.domain.Product).where(hasValueTestCondition.an
d(hasObjectTypeCondition));

var result = querySpecification.execute();
return result;
```

11.4 Date Handling

Comparing dates can be a little tricky thus Java is kindly offering some tools to compare dates against each other so that it for instance can be determined if a date is before or after another date.

11.4.1 Example 1: Finding out if a date has passed

```
function hasISODateBeenExceeded(dateString) {
      return hasDateBeenExceeded(dateString, "yyyy-MM-dd");
}

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

BETTER DATA.
BETTER BUSINESS.
BETTER WORLD.