

# STEP Business Rules

## Good Practices

VERSION: 1.0

AUTHOR: Stibo Systems Professional Services

CONFIDENTIALITY LEVEL: Confidential

# Content

<b>1</b>	<b>Document Control</b> .....	<b>3</b>
1.1	Document purpose.....	3
1.2	Content of the document.....	3
<b>2</b>	<b>Types of business rules</b> .....	<b>4</b>
<b>3</b>	<b>Readability</b> .....	<b>5</b>
3.1	Business rules objects in STEP .....	5
3.2	Bind variables .....	5
3.3	Variable and function names .....	6
3.4	Indentation .....	6
<b>4</b>	<b>Use of libraries</b> .....	<b>7</b>
4.1	Avoid large business rule libraries .....	7
<b>5</b>	<b>Logging</b> .....	<b>8</b>
<b>6</b>	<b>Exception handling</b> .....	<b>10</b>
<b>7</b>	<b>Performance</b> .....	<b>12</b>
7.1	Investigating business rule performance.....	12
7.2	Performance good practices .....	15
7.2.1	Keep business rule transactions small .....	15
7.2.2	Avoid the function GetChildren with many nodes .....	16
7.2.3	Use arrays instead of multiple read calls .....	16
7.2.4	Consider In-Memory for business rules .....	16

## 1 Document Control

---

### 1.1 Document purpose

This document describes good practice guidelines for writing STEP business rules. 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 big impact on functionality and performance.

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

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

### 1.2 Content of the document

The document will contain 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 business rules.
- 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: Reuse code instead of copy and pasting it.
- Logging: Good log messages makes debugging easier, but make sure that it can be turned on and off, so it doesn't 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 very important. Business rules used occasionally for bulk updates may be less critical in regards to performance.

## 2 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 fourth type of business rule, **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.

## 3 Readability

Code which is easy to read is easy to maintain and support. Readability improves when using self-explaining names and proper indentation.

### 3.1 Business rules objects in STEP

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

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

Example:

The screenshot shows a web interface for a business rule. At the top, there's a title bar with a back arrow, the text 'Check Mandatory Primary Image rev.0.1 - Business Rule', and a right arrow. Below the title bar are several tabs: 'Business Rule' (selected), 'Usage', 'Statistics', 'Log', and 'Status'. The main content area is a table with two columns: 'Name' and 'Value'. The table has two rows: one with 'ID' in the Name column and 'CheckMandatoryPrimaryImage' in the Value column, and another with 'Name' in the Name column and 'Check Mandatory Primary Image' in the Value column. Both rows have a right arrow in the Name column and a right arrow in the Value column.

Name	Value
ID	CheckMandatoryPrimaryImage
Name	Check Mandatory Primary Image

### 3.2 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.

Examples:

Bind variable name to use	For binding to type
<b>approveContext</b>	Approve Context
<b>currentEventQueue</b>	Current Event Queue
<b>currentEventType</b>	Current Event Type
<b>currentObject</b>	Current Object
<b>currentWorkflow</b>	Current Workflow
<b>&lt;derivedEventTypeId&gt;EventType, e.g. webLinkEventType</b>	Event Type
<b>&lt;OIEPID&gt;EventQueue, e.g. eCommEventQueue</b>	Event Queue
<b>gdsnDataMap</b>	GDSN Data Map
<b>currentObjectID</b>	ID

<b>importChangeInfo</b>	Import Change Info
<b>jsonDocument</b>	JSON Document
<b>logger</b>	Logger
<b>lookupTableHome</b>	Lookup Table Home
<b>mailer</b>	Mailer
<b>manager</b>	STEP Manager
<b>mongoDBContext</b>	MongoDB Context
<b>currentObjectName</b>	Name
<b>&lt;attributeID&gt;Value, unless auto-ID is used, e.g. gtinValue</b>	Attribute Value
<b>workflowParameters</b>	Workflow Parameters

### 3.3 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.

### 3.4 Indentation

Use 2 spaces for indentation. Don't use tabs. Keep lines shorter than 80 characters.

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";
}
```

## 4 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

### 4.1 Avoid large business rule libraries

Be aware that JavaScript libraries are compiled each time on execution. Each dependency is stacked into a big script and this one is compiled before each execution of the business rule by using the `ScriptEngine.eval()` method.

For example: Script S is depending on library A and library B. Library A is depending on B. In this case the script is stacked as follows:

- A::script
- A::B::script
- B::script
- S::Bindings
- S::script

STEP caches the scripts instead of reloading them from the database. By default, 100 business rules are cached. Still, as a rule of thumb, it takes about 500 milliseconds to compile about 8,500 lines of code at each business rule execution. Therefore, be careful with huge libraries, especially if these are dependent on each other and are used in many business rules, since these libraries will be compiled every time the business rule is executed.

Even if the large library is broken down into multiple libraries, but the libraries still maintain to be dependent on each other, then still it doesn't resolve the issue.

It's recommended to analyse which performance increase can be achieved by making the library functions local to the business rule itself, in case large libraries are used and the business rule suffers from bad performance.

## 5 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's therefore recommended to configure the business rule logging properly to avoid unnecessary logging of business rule details.

- Use the `BusinessRule.Warning.Threshold` configuration property in `sharedconfig.properties` to specify a threshold in milliseconds for business action execution. If it takes longer to execute or test a given business action, a warning is posted in the main STEP log file.

The value is in milliseconds. So, `BusinessRule.Warning.Threshold=10000` means write warning to the STEP log file whenever a business rule execution takes longer than 10 seconds.

### BusinessRule

`BusinessRule.Warning.Threshold=10000`

- Set in: `/opt/stibo/step/sharedconfig.properties`
- Using default value
- Must be an Integer.

Time before logging slow business rules - in milliseconds

- Use the `Log.Level.com.stibo.scripting.StepScriptEngineManager` configuration property in `sharedconfig.properties` to specify the level of detail for logging business rules results.

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)/`
- (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`

- For the logging of business rules, it's good practise 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 analysed by analysing 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' DO NOT use unless you develop or test

//When doing tests DO NOT test on large amount of products

//REMEMBER to turn 'false' when you are done

var isDebug = false;

//Function to handle whatever logging of debug information should occur or not

function logDebug(message) {

    if (isDebug) {logger.info(message) }

}

...

logDebug("Here's a message for the log file")

...
```

It's recommended to avoid unnecessary business rule logging by configure the business rules logging settings in the `sharedconfig.properties` configuration file and set a logging debug flag in the business rules code itself.

## 6 Exception handling

---

If an error occurs during approval, an exception is thrown from the domain layer. If this exception is caught in business rules 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 behaviour also has a negative effect on the performance of the business rule.

You need to be careful when doing exception handling to avoid this behaviour when using “try-catch” in business rules.

- What is NOT recommended

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) {  
  
}
```

- What IS recommended

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

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

For example:

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

## 7 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, in order to avoid optimistic locking errors.

### 7.1 Investigating business rule performance

There're several ways to analyse and monitor business rules.

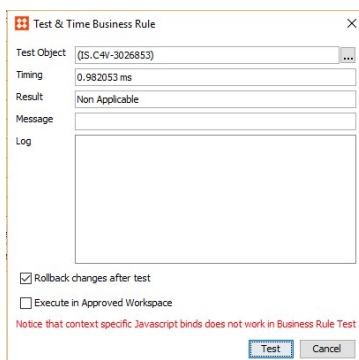
- Workbench business rule test menu

The Workbench business rule test menu is typically used during development

- 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're certain the business rule will fail or pass.
- Then analyse the `timing` of the business rule to see its performance.

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

For example: The following screenshot shows that the business rule took about 0.98 milliseconds to complete the business rule for item `IS.C4V-3026853`. Nevertheless, be aware that it might take longer or shorter for other Items.



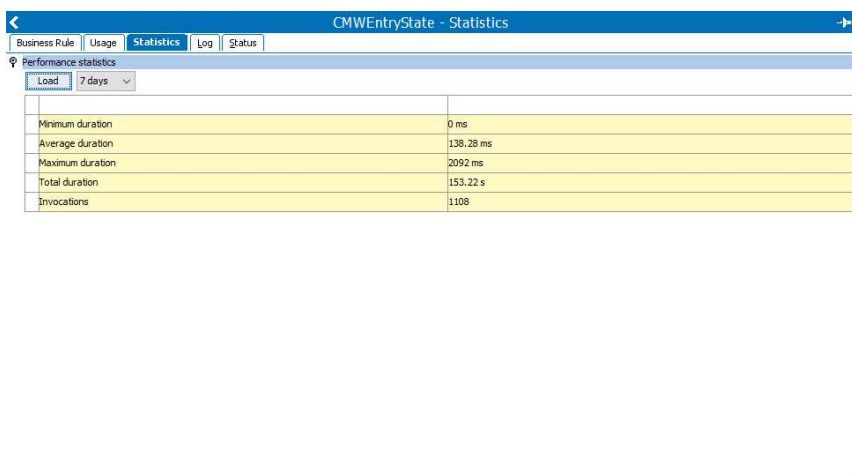
This method of business rule analysis gives a first indication of the performance of the business rule for a certain item.

- 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.



The screenshot shows a window titled "CMWEntryState - Statistics" with tabs for "Business Rule", "Usage", "Statistics", "Log", and "Status". The "Statistics" tab is active, showing "Performance statistics" for a "Load" of "7 days". A table displays the following data:

Metric	Value
Minimum duration	0 ms
Average duration	138,28 ms
Maximum duration	2092 ms
Total duration	153,22 s
Invocations	1108

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

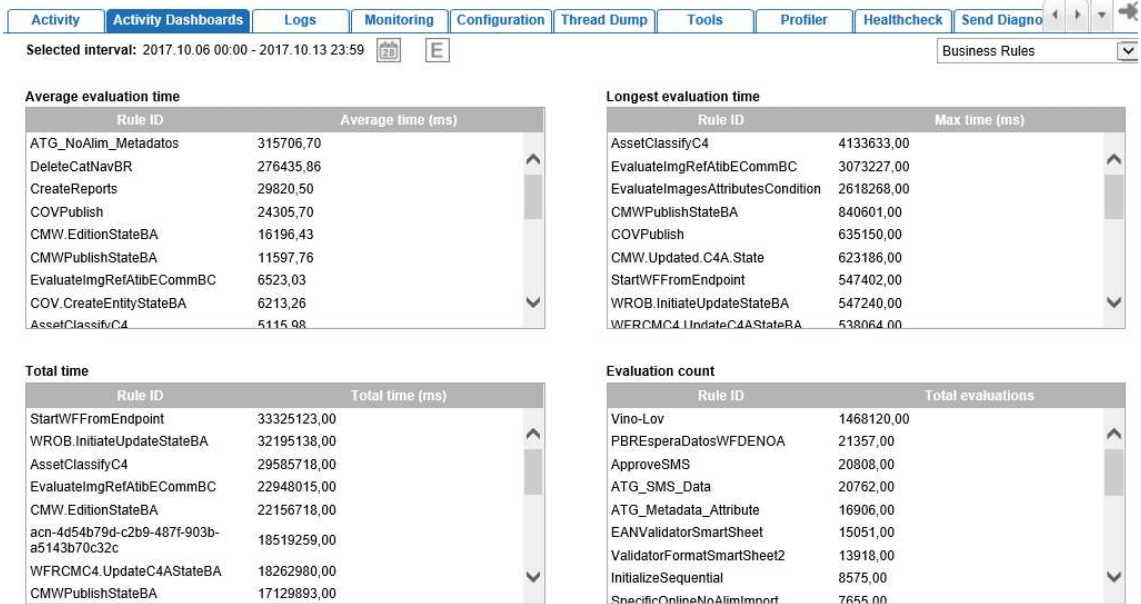
This method of business rule analysis gives an indication of the performance of the business rule over a period of time.

- 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:



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

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

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

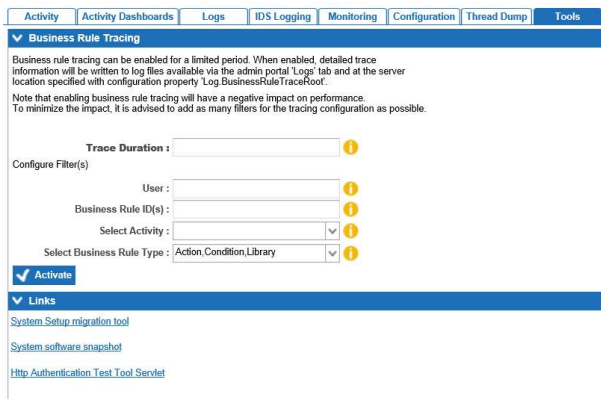
- Admin Portal business rule tracing

There's an option in the Admin Portal of version STEP 8.1 and higher 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 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.

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



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

Note: Once tracing has been activated, the relevant business rule(s) must be triggered in 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.

The 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`.

## 7.2 Performance good practices

### 7.2.1 Keep business rule transactions small

Business actions have a transaction, which allows you to manipulate data in 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.

### 7.2.2 Avoid the function GetChildren with many nodes

Business rules using calls "getChildren" on a huge number of children may cause memory problems. It should not be possible to expand a node having more than 10.000 children.

The problem that the "getChildren" uses an unsafe call that will read all children. It should be changed into using "queryChildren".

```

TreeNavigatorServiceBean

    public List getChildren(final String internalId, final Class wantedChildType,
                           final Manager manager) {
    ...
        } else if (obj instanceof Product) {
            Product product = (Product) obj;
            children = product.getChildren(); // also works for product overrides <-- Unsafe!!!
    }
    }

```

It's recommended to analyse the business rules and determine if "getChildren" is not used on a selection with more than 10.000 children. Otherwise, change it to "queryChildren".

### 7.2.3 Use arrays instead of multiple read calls

Business rules using calls to the database for large sets of data again and again, will perform significantly worse than using one call to get the data, push it into arrays and work from there, because the number calls to the database will be minimized

Also in situations where multiple business rules are executed sequentially (e.g. as part of an approval process) and these business rule fetch the same data from the database multiple times again and again, it's worth to rewrite these business rules to fetch the data once, push the data into (multi-dimensional) arrays, and use the arrays, instead of reading the data from the database again and again.

It's recommended to analyse the business rules and determine if data is read efficiently from the database (once instead of many times) and arrays are used. Change the business rule to read data from the database preferably once and use arrays to work with the data in the business rule.

### 7.2.4 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.