Red Hat JBoss BPM Suite 6.2

Migration Guide

Migrating from earlier versions to Red Hat JBoss BPM Suite 6.2

Last Updated: 2017-11-13

Red Hat JBoss BPM Suite 6.2 Migration Guide

Migrating from earlier versions to Red Hat JBoss BPM Suite 6.2

Doug Hoffman

Eva Kopalova

Gemma Sheldon
Red Hat Engineering Content Services
gsheldon@redhat.com

Christian Huffman
Red Hat Engineering Content Services
chuffman@redhat.com

Legal Notice

Abstract

The Guide will help you migrate from JBoss Enterprise BRMS Platform 5.3.1, and earlier versions of Red Hat JBoss BPM Suite, to Red Hat JBoss BPM Suite 6.2

Table of Contents

CHAPTER 1. WHY MIGRATE?

Migrating to JBoss BRMS/BPM Suite 6 from BRMS 5.x branch means that you get the following new features:

- A higher performance rule engine based on the Drools 6 community project.
- Improved rule authoring tools and an enhanced, integrated user interface.
- A common defined methodology for building and deployment using Maven as the basis for repository management.
- A heuristic planning engine based on the OptaPlanner community project.
- Use of the PHREAK algorithm to handle a larger number of rules and facts.
- New Data Modeler that replaces the declarative Fact Model Editor
- And many more stability, usability and functional improvements.

Note that Red Hat JBoss BPM Suite 6 is now the superset that includes a version of the Red Hat JBoss BRMS 6. Red Hat JBoss BRMS 6 by itself does not include business process management capabilities.

Being backwards compatible with JBoss BRMS 5.x branch, migrating to Red Hat JBoss BRMS 6 means that your rules created with BRMS 5.x will still execute on BRMS 6 without changes. A migration tool is also provided with the 6 branch to facilitate migration of the content of a JBoss BRMS 5.x repository to Red Hat JBoss BRMS/BPM Suite 6. Note, however, that BPMN2 process models created with JBoss BRMS 5.x will not execute on Red Hat JBoss BRMS 6.

CHAPTER 2. MIGRATING FROM JBOSS ENTERPRISE BRMS PLATFORM 5.3.1 TO RED HAT JBOSS BPM SUITE 6.X

2.1. WHAT HAS CHANGED?

Before starting the migration process, let's find out what exactly has changed between the 5 and 6 branches of Red Hat JBoss BRMS/BPM Suite.

Naming Changes

First and foremost it is important to understand how older components in the 5 branch that you were familiar with relate to in the 6 branch. The biggest change, of course, is the name of the product and it's structure.

JBoss Enterprise BRMS is now called Red Hat JBoss BRMS Red Hat JBoss BRMS, or JBoss BRMS, is a standalone product with its own workbench and is used solely to create and manage your business rules.

A new product, called Red Hat JBoss BPM Suite, or JBoss BPM Suite encapsulates the JBoss BRMS product and on top of that provides Business Process Management which allows you to create, manage, validate, and deploy Business Processes *and* Rules.

Component Name Migration

Several components from the 5 branch to 6 branch have been renamed or merged with other components to provide a better working environment. The figure below captures these changes.

API Name Changes

Table 2.1.

5.x API (deprecated in 6.x)	Replaced in 6.x by
org.drools.KnowledgeBase	org.kie.api.KieBase
org.drools.runtime.StatefulKnowledgeSession	org.kie.api.runtime.KieSession
org.drools.runtime.StatelessKnowledgeSession	org.kie.api.runtime.KieSession
org.drools.builder.KnowledgeBuilderFactory	org.kie.internal.builder.KnowledgeBuilderFactory
org.drools.io.ResourceFactory	org.kie.internal.io.ResourceFactory
org.drools.io.ResourceType	org.kie.internal.io.ResourceType
org.drools.runtime.Environment	org.kie.api.runtime.Environment
org.drools.runtime.EnvironmentName	org.kie.api.runtime.EnvironmentName
org.drools.runtime.KnowledgeSessionConfiguration	org.kie.api.runtime.KieSessionConfiguration
org.drools.event.AgendaEventListener	org.kie.api.event.rule.AgendaEventListener

5.x API (deprecated in 6.x)	Replaced in 6.x by
org.drools.event.rule.AgendaEventListener	org.kie.api.event.rule.AgendaEventListener
org.drools.event.DefaultAgendaEventListener	org.kie.drools.core.event.DefaultAgendaEventListener
org.drools.event.rule.DefaultAgendaEventListener	org.kie.api.event.rule.DefaultAgendaEventListener
org.drools.event.process.ProcessEventListener	org.kie.api.event.process.ProcessEventListener
org.drools.event.process.DefaultProcessEventListener	org.kie.api.event.process.DefaultProcessEventListener
org.drools.logger.KnowledgeRuntimeLogger	org.kie.api.logger.KieRuntimeLogger
org.drools.logger.KnowledgeRuntimeLoggerFactory	org.kie.api.logger.KieLoggers

Repository Change

Red Hat JBoss BRMS and JBoss BPM Suite 6 are backed by a GIT repository for source management while JBoss BRMS 5 was backed by a JCR/JackRabbit implementation. A tool to migrate the repository to GIT is provided and discussed later in this guide.

Changes to the REST API

The base url for working with the REST API has changed from **`http://SERVER_ADDRESS:PORT/jboss-brms/rest/`** to **`http://SERVER_ADDRESS:PORT/business-central/rest/`**

Migrating Task Service

JBoss BPMS 6 provides support for a locally running task server only. This means that you don't need to setup any messaging service in your project. This differs from JBoss BRMS 5 because it provided a task server that was bridged from the core engine by using, most commonly, the messaging system provided by HornetQ.

To help you bridge the gap until you can migrate this in your current architecture, there is a helper or utility method, **`LocalHTWorkItemHandler`** located in **`org.jbpm.services.task.wih`**.

Since the TaskService API is part of the public API you will now need to refactor your imports because of package changes and refactor your methods due to API changes themselves.

Logging factory

Besides the API name change, logging is now implemented through the org.kie.api.logger package which contains the factory class **`KieLoggers`** that can be used to create instances of the **`KieRuntimeLogger`**.

2.2. HOW-TO MIGRATE

Migrating your projects from Red Hat JBoss BPM Suite 5 to Red Hat JBoss BPM Suite 6 requires careful planning and step by step evaluation of the various issues. You can plan for migration either manually, or by using automatic processes. Most real world migration will require a combination of these two processes.

Because JBoss BPM Suite 6 uses GIT for storing assets, artifacts and code repositories including processes and rules, you should start by creating an empty project in JBoss BPM Suite 6 as the basis for your migration with dummy files as placeholders for the various assets and artifacts. Running a GIT clone of this empty project into your favorite IDE will initiate the migration process.

Based on the placeholder files in your cloned project, you can start adding assets at the correct locations. The JBoss BPM Suite 6 system is smart enough to pick these changes and apply them correctly. Ensure that when you are importing old rule files that they are imported with the right package name structure.

Since Maven is used for building projects, the projects assets like the rules, processes and models are accessible as a simple jar file.

This section lists the generally accepted step by step ways to migrate your project. These are just guidelines though, and actual migration may vary a lot from this.

In general, you should...

1. Migrate the data first: These are your business assets.
2. Next, migrate your runtime processes.
3. Finally, convert old API calls to new ones one by one.

Let's look at these steps in more detail in the next few sections.

2.2.1. Data Migration

To migrate data from Red Hat JBoss BPM Suite 5, do the following:

1. Download the migration tool by logging in at the Red Hat Customer Portal and then navigating to Red Hat JBoss BPM Suite Software Downloads section. Click on **`Red Hat JBoss BPM Suite Migration Tool`** to download the zip archive.
2. Unzip the downloaded zip archive in a directory of your choice and navigate to this directory in a command prompt. This directory contains four folders:
   - **`bin`** - contains the launch scripts.
   - **`jcr-exporter-libs`** - contains the libs specific to the **`export-from-JCR`** part of the migration.
   - **`vfs-importer-libs`** - contains the libs specific to the **`import-into-Git`** part of the migration.
   - **`conf`** - contains global migration tool configuration.
3. For production databases, copy the JDBC driver for the database that is used by the JCR repository into the **`jcr-exporter-libs`** directory of the migration tool.
4. Execute the following command:

```
./bin/runMigration.sh -i <source-path> -o <destination-path> -r
<repository-name>
```

Where:

- **<source-path>** is a path to a source JCR repository.
- **<desintation-path>** is a path to a destination GIT VFS. This folder must not exist already.
- **<repository-name>** an arbitrary name for the new repository.

The repository is migrated at the specified destination.

Besides the **-i** command, you can also use **-h** to print out a help message and **-f** which forces an overwrite of the output directory, thus eliminating the need for manual deletion of this directory.

Importing the repository in Business Central
The repository can be imported in business central by cloning it. In the Administration perspective, click on the **Repositories** menu and then click on **Clone Repository** menu to start the process.

NOTE

Assets can also be migrated manually. After all, they are all just text files. The BPMN2 specification and the DRL syntax did not change between the different versions.

Importing the repository in JBDS
To import the repository in JBoss Developer Studio, do the following

1. Start JBoss Developer Studio.
2. Start the Red Hat JBoss BPM Suite server (if not already running) by selecting the server from the server tab and click the start icon.
3. Select **File** → **Import...** and navigate to the Git folder. Open the Git folder to select **Projects from Git** and click next.
4. Select the repository source as **Existing local repository** and click next.
5. Select the repository that is to be configured from the list of available repositories.
6. Import the project as a general project in the next window and click next. Name this project and click Finish.

2.2.2. Runtime Migration

To run Red Hat JBoss BPM Suite 5 processes in Red Hat JBoss BPM Suite 6, do the following:

1. Set the system property **jbpm.v5.id.strategy** to true in the JBoss BPM Suite **standalone.xml** file:

```
<property name="jbpm.v5.id.strategy" value="true"/>
```

2. Load the KieSession as shown here:

```
KieSession ksession =
JPAKnowledgeService.loadStatefulKnowledgeSession(sessionID, kbase,
sessionConf, env);
```

3. Continue the normal execution of the process using KieSession methods:

```
ksession.signalEvent("SomeEvent", null);
```

2.2.3. API and Backwards Compatibility

Migrating to Version 6.1 and later

In version 6.1, 5.x APIs are no longer officially supported.

Red Hat JBoss BPM Suite no longer provides backward compatibility with the rule, event, and process application programming interface (API) from JBoss BRMS 5. The content of the **knowledge-api JAR** file is no longer supported in version 6.1 and onward, and is replaced by APIs contained in the **kie-api JAR** file that were introduced in JBoss BPM Suite 6.0.

If you used the legacy 5.x API (located in **knowledge-api.jar**), please migrate (rewrite) the API calls to the new KIE API. Please be aware that several other APIs have changed between JBoss BRMS 5.x and JBoss BPM Suite 6.x, namely the task service API and the REST API.

Migrating to Version 6.0

The JBoss BPM Suite 6 system provides backward compatibility with the rule, event and process interactions from JBoss BRMS 5. You should eventually migrate (rewrite) these interactions to the all new revamped core API because this backward compatibility is likely to be deprecated.

If you cannot migrate your code to use the new API, then you can use the API provided by the purpose built **knowledge-api** jar for backwards compatible code. This API is the public interface for working with JBoss BPM Suite and JBoss BRMS and is backwards compatible.

If you are instead using the REST API in JBoss BPM Suite 5, note that this has changed as well and there is no mechanism in it for backwards compatibility.

2.3. ADVANCED MIGRATION

2.3.1. REST API Migration

As you know, there are two ways you can use Red Hat JBoss BRMS/JBoss BPM Suite: in an embedded mode or in a remote mode. In the embedded mode, you package the runtime libraries with your application and execute the BPMN processes in the same JVM. In the remote mode, the business assets are created, deployed and executed in the same server, but they are accessed via a remote client application using the REST or the JMS API.

Both of these modes presents different challenges when it comes to the migration. In this section we will focus on the migration for the remote usage of JBoss BPM Suite with the help of a REST example.

REST API Example

In this example, let's assume that you want to:

- start a process which is already deployed in the JBoss BPM Suite server.

- pass some parameters to this process during its creation.

The client side for JBoss BRMS 5 can be created using the following code:

```
import org.apache.commons.httpclient.HttpClient;
import org.apache.commons.httpclient.HttpMethod;
import org.apache.commons.httpclient.NameValuePair;
import org.apache.commons.httpclient.methods.GetMethod;
```

```
import org.apache.commons.httpclient.methods.PostMethod;
import
org.apache.commons.httpclient.methods.multipart.MultipartRequestEntity;
import org.apache.commons.httpclient.methods.multipart.Part;
import org.apache.commons.httpclient.methods.multipart.StringPart;
import org.jboss.bpm.console.client.model.ProcessDefinitionRef;
import org.jboss.bpm.console.client.model.ProcessDefinitionRefWrapper;
import org.jboss.bpm.console.client.model.ProcessInstanceRef;


 public class RestClientStartWithParam {
    private static final String BASE_URL =
"http://localhost:8080/business-central-server/rs/";
    private static final String AUTH_URL = BASE_URL +
"identity/secure/j_security_check";
    private final String username;
    private final String password;

    private static final String PROCESS_ID = "defaultPackage.hello";

    public RestClientStartWithParam(final String u, final String p) {
        this.username = u;
        this.password = p;
    }

    public static void main(String[] args) throws Exception {

        RestClientStartWithParam client = new
RestClientStartWithParam("admin", "admin");

        // get process definitions
        ProcessDefinitionRefWrapper processDefinitionWrapper =
client.getProcessDefinitions(client);

        // pick up "com.sample.bpmn.hello"
        ProcessDefinitionRef definitionRef = null;
        for (ProcessDefinitionRef processDefinitionRef :
processDefinitionWrapper.getDefinitions()) {
            if (processDefinitionRef.getId().equals(PROCESS_ID)) {
                definitionRef = processDefinitionRef;
                break;
            }
        }
        if (definitionRef == null) {
            System.out.println(PROCESS_ID + " doesn't exist");
            return;
        }

        // start a process instance with parameters
        Map<String, String> params = new HashMap<String, String>();
        params.put("employee", "thomas");
        params.put("reason", "theReason");
        client.startProcessWithParameters(client, definitionRef, params);
    }

        private void startProcessWithParameters(RestClientStartWithParam
```

```
client, ProcessDefinitionRef def,
            Map<String, String> params) throws Exception {
        String newInstanceUrl = BASE_URL + "form/process/" + def.getId() +
"/complete";
        String dataFromService = client.getDataFromService(newInstanceUrl,
"POST", params, true);

        System.out.println(dataFromService);
    }

    // get DataFromService method can be implemented like this

    private String getDataFromService(String urlpath, String method,
Map<String, String> params, boolean multipart)
            throws Exception {
        HttpClient httpclient = new HttpClient();

        HttpMethod theMethod = null;
        StringBuffer sb = new StringBuffer();

        if ("GET".equalsIgnoreCase(method)) {
            theMethod = new GetMethod(urlpath);
        } else if ("POST".equalsIgnoreCase(method)) {
            theMethod = new PostMethod(urlpath);

            if (params != null) {

                if (multipart) {
                    List<Part> parts = new ArrayList<Part>();
                    for (String key : params.keySet()) {
                        StringPart stringPart = new StringPart(key,
params.get(key));
                        stringPart.setCharSet("UTF-8");
                        parts.add(stringPart);
                    }
                    ((PostMethod) theMethod).setRequestEntity(new
MultipartRequestEntity(parts.toArray(new Part[0]),
                            theMethod.getParams()));
                } else {

                    List<NameValuePair> NameValuePairList = new
ArrayList<NameValuePair>();
                    for (String key : params.keySet()) {
                        NameValuePairList.add(new NameValuePair(key,
params.get(key)));
                    }
                    ((PostMethod)
theMethod).setRequestBody(NameValuePairList.toArray(new
NameValuePair[0]));

                }
            }

        }

        if (username != null && password != null) {
```

```
            try {
                int result = httpclient.executeMethod(theMethod);
                System.out.println("result = " + result);
                //
System.out.println(theMethod.getResponseBodyAsString());
            } catch (IOException e) {
                e.printStackTrace();
            } finally {
                theMethod.releaseConnection();
            }
            PostMethod authMethod = new PostMethod(AUTH_URL);
            NameValuePair[] data = { new NameValuePair("j_username",
username),
                    new NameValuePair("j_password", password) };
            authMethod.setRequestBody(data);
            try {
                int result = httpclient.executeMethod(authMethod);
                System.out.println("result = " + result);
                //
System.out.println(theMethod.getResponseBodyAsString());
            } catch (IOException e) {
                e.printStackTrace();
            } finally {
                authMethod.releaseConnection();
            }
        }

        try {
            int result = httpclient.executeMethod(theMethod);
            System.out.println("result = " + result);
            sb.append(theMethod.getResponseBodyAsString());
            String rawResult = sb.toString();
            return rawResult;

        } catch (Exception e) {
            throw e;
        } finally {
            theMethod.releaseConnection();
        }
    }
```

The JBoss BRMS 5 endpoints are documented here and here.

As you can see, even this very simple example looks rather complex when implemented. The reason for this is partially that there is no native client for JBoss BRMS 5 server. You can however choose the optional web client - Apache HttpClient, RestEasy or even use just plain java.net libraries. This applies in JBoss BPM Suite/BRMS 6 as well - you can still choose the web client - however, there is also a native java client provided for remote communication with version 6 which is much simpler to use.

Migrate to JBoss BRMS/JBoss BPM Suite 6

So let's migrate the same use case to JBoss BPM Suite 6:

- process is already deployed in the JBoss BPM Suite server

- we want to start it with some parameters

- this time, there are some human tasks in this process, so we want to complete those.

All of the available REST endpoints for JBoss BRMS/JBoss BPM Suite are documented here. The code for the requirements above in the standalone application will look like this:

```
import java.net.MalformedURLException;
import java.net.URL;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.kie.api.runtime.KieSession;
import org.kie.api.task.TaskService;
import org.kie.api.task.model.TaskSummary;
import org.kie.services.client.api.RemoteRestRuntimeEngineFactory;
import org.kie.services.client.api.command.RemoteRuntimeEngine;

 public class Main {

 public static void main(String[] args) throws MalformedURLException {

  /*
   * Set the parameters according to your installation
   */

  String APP_URL = "http://localhost:8080/business-central/";

  URL url = new URL(APP_URL);
  String USER = "anton";
  String PASSWORD = "password1!";

  RemoteRestRuntimeEngineFactory factory =
RemoteRestRuntimeEngineFactory.newBuilder()
    .addDeploymentId("org.redhat.gss:remote-test-
project:3.0").addUrl(url).addUserName(USER)
    .addPassword(PASSWORD).build();

  RemoteRuntimeEngine engine = factory.newRuntimeEngine();
  KieSession kSession = engine.getKieSession();
  TaskService taskService = engine.getTaskService();

   // start a process instance with parameters
     Map<String, Object> params = new HashMap<String, Object>();
     params.put("employee", "thomas");
     params.put("reason", "theReason");
     kSession.startProcess("com.sample", params);

     taskService.claim(taskSummaryList.get(0).getId(), "anton");
     taskService.start(taskSummaryList.get(0).getId(), "anton");
     taskService.complete(taskSummaryList.get(0).getId(), "anton", null);
// not passing any data = null
  }
}
```

As you can see, this example is much more simple and readable than the one for JBoss BRMS 5. The **`RemoteRuntimeEngine`** gives us direct access to the **`TaskService`** / **`KieSession`** and **`AuditLogService`** API.

However, it is still possible to use your arbitrary Java web client and achieve the same scenario by sending GET/POST requests to the appropriate endpoints.

NOTE

While the basic functionality is provided by both APIs - JBoss BRMS 5 and JBoss BRMS/JBoss BPM Suite 6, (starting the process, completing the tasks and so on) not all endpoints from BRMS 5 have their replacement in JBoss BRMS/JBoss BPM Suite 6. If in doubt, consult the corresponding documentation of the REST API.

2.3.2. KnowledgeAgent to KieScanner Migration

KnowledgeAgent is a component of JBoss BRMS 5 which allows you to obtain Knowledge Bases dynamically as it gets updated. If you correctly configure an instance of **`KnowledgeAgent`** and then you try to obtain the **`KnowledgeBase`** from the agent, you will be able to receive the latest version of the **`KnowledgeBase`** including updated resources - whether it is a freshly built package (*.PKG) in Business Central or BPMN process definition updated via the Eclipse designer tool.

See a working example of how this works in version 5 here: KnowlegeAgent Example.

In JBoss BRMS and JBoss BPM Suite 6 it is also possible to obtain **`KieBase`** (instead of **`KnowledgeBase`**) dynamically as it gets updated. However, the migration is not so straightforward, because of a few things:

- In JBoss BRMS 5, the native storage for packages was Guvnor - which used JackRabbit repository underneath. You could also point to a single resource (drl, bpmn..) with any valid URL (i.e. file://, http://, ...).
- The API is completely different as there is no direct mapping between KnowledgeAgent API in JBoss BRMS/JBoss BPM Suite 6.

The component which replaces the **`KnowledgeAgent`** in BRMS 6 is called **`KieScanner`** and therefore you need to include **`kie-ci`** library on classspath if you want to use it.

To see an example of how this works in version 6: KieScanner Example.

In version 6, you no longer refer to **`*.PKG`** files or specific business resources such as drl, bpmn. Instead you configure your **`KieScanner`** with a specific KJAR, which is a Maven artifact including your resources, identified by GAV. **`KieScanner`** uses the Maven Repository to figure out where to look for these built KJARs. If not specified otherwise, it will look into your local Maven repository (by default stored under ~/.m2/ directory on your filesystem).

A typical scenario will be where you set GAV so it identifies projects created in Business Central. **`KieScanner`** is now bound to this project, and once you make changes to this project in Business Central and build the project, it's latest build will be stored into the local Maven repository (this is the default). **`KieScanner`** scans the local Maven repository and picks up the changes. If you want to configure KieScanner in a way that it scans other repositories besides your local one you can do so by setting a system property: kie.maven.settings.custom which can point to the custom settings.xml (a standard Maven configuration file where you include all repositories which should be taken into consideration).

KieScanner invokes Maven under the hood for artifact lookup by following known Maven conventions and rules. For example:

- If the remote repository requires authentication, you need to configure this authentication in a Maven way by updating **settings.xml**.
- If you point your **KieScanner** to a KJar with GAV **org.my:project:1.0** your KieBase will never get updated even if you build the project with the same GAV again. This is because Maven will resolve the artifact to a fixed version.
- If you point your **KieScanner** to a KJar with GAV **org.my:project:1.0-SNAPSHOT** your KieBase will get updated for any new build of the project with that GAV - it will be resolved to the LATEST build of that GAV, identified by the timestamp.

A KCS article which discuss various scenarios and configurations is available here:
https://access.redhat.com/solutions/710763

2.3.3. Database Migration

The default underlying database in JBoss BPM Suite is an instance of H2. This is fine for most test systems, but production systems are generally based around MySQL, PostgreSQL, Oracle or others databases. This section lists some of the tips and tricks related to databases when migrating from BRMS 5 to JBoss BPM Suite 6.x.

- **Include hbm.xml for PostgreSQL**
  If the underlying database on which the migration is being performed is PostgreSQL, you will need to include an additional configuration file, called **hbm.xml** inside the **META-INF** directory, next to **persistence.xml**, with the following contents:

  ```
  <?xml version="1.0"?>
  <!DOCTYPE hibernate-mapping PUBLIC
      "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
      "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
  <hibernate-mapping>
    <typedef name="materialized_clob"
  class="org.hibernate.type.TextType" />
  </hibernate-mapping>
  ```

  This file instructs Hibernate to use **TextType** for materialized CLOBS and solves an issue where Hibernate incorrectly tries to interpret the type of a parameter as Long when it should be String based.

- **Avoid ID constraint violations in PostgresSQL and Oracle**
  NodeInstanceLog in BRMS 5.2.x doesn't have a sequence generator associated with it and was added to have more consistent behavior with multiple databases. Since not all databases initialize the id sequence correctly on migration it is necessary to update the NODE_INST_LOG_ID_SEQ id manually. The two databases that are affected are: PostgreSQL and Oracle.

  - PostgreSQL: In PostgreSQL two actions are required:
    - Find the **id** with the biggest value in the **NodeInstanceLog** table:

      ```
      SELECT MAX(id) FROM nodeinstancelog;
      ```

- Restart sequence NODE_INST_LOG_ID_SEQ using the result from the previous step, increased by 1. So if the command in the previous step returned the number **10** you will use **11** in the following command.

```
ALTER SEQUENCE node_inst_log_id_seq RESTART WITH 11;
```

The reason to increase the result from the first step by **1** is that restarting the sequence sets the is_called flag to **false**, which tells the system that the sequence was not yet used.

- Oracle: In Oracle, the following steps are required:
  - Find the **id** with the biggest value in the **NodeInstanceLog** table:

```
SELECT MAX(id) FROM nodeinstancelog;
```

  - Execute the following commands in SQL:

```
-- Re-create the sequence by first dropping it and then
creating a new one.
DROP SEQUENCE NODE_INST_LOG_ID_SEQ;
CREATE SEQUENCE NODE_INST_LOG_ID_SEQ START WITH 11 INCREMENT
BY 1 NOCYCLE;

-- Increase the sequence (the result must be greater then the
result obtained in step 1)
ALTER SEQUENCE NODE_INST_LOG_ID_SEQ INCREMENT BY 100;
```

CHAPTER 3. MIGRATING FROM RED HAT JBOSS BPM SUITE 6.X TO A LATER VERSION

3.1. MIGRATING FROM 6.1.X TO 6.2

3.1.1. Migrating Client Application APIs from 6.1.x to 6.2

In this section we will discuss the migration steps that are specific to migrating from Red Hat JBoss BPM Suite 6.1.x to 6.2. While the majority of settings and API have remained the same the differences are highlighted below.

jBPM Executor (embedded mode)
The jBPM Executor has now been exposed as a public API available in the KIE API. In the event that the **`jbpm-executor`** service was in use previously, with embedded environments, it is recommended to now create the jBPM Executor using the API.

Deprecated methods
The **`TaskSummary.getPotentialOwners()`** is now deprecated and slated for removal.

Update IDs from primitive types
Many IDs were changed from the primtive **`long`** to the Object **`Long`**. While Java can perform auto-casting of these primitive types it is recommended to update any uses of the following methods to avoid potential issues:

```
org.kie.api.task.model.Content.getId();
org.kie.api.task.model.TaskData.getOutputContentId();
org.kie.api.task.TaskService.addComment();
```

3.1.2. Migrating Business Central Project, Repository, and Artifacts from 6.1.x to 6.2

The recommended method of upgrading the Red Hat JBoss BPM Suite 6.1.x git projects (**`.niogit`** folder) and maven local dependencies (**`bin/repositories`** is to copy over these directories to the new installation. This process will copy over all projects and artifacts.

Procedure 3.1. Migrate the Git Repository

1. Turn off JBoss BPM Suite 6.1.x and 6.2 instances.
2. Navigate to the **`.niogit`** directory, found in the home directory of the JBoss BPM Suite 6.1 installation.
3. Copy this directory, and any subdirectories, to the **`.niogit`** directory of the JBoss BPM Suite 6.2 installation:

```
cp -R ./ $BPM_6.2_INSTALLATION/.niogit/
```

Procedure 3.2. Migrate the Maven Repository

1. Turn off JBoss BPM Suite 6.1.x and 6.2 instances.
2. Navigate to the **`repositories`** directory, found in the home directory of the JBoss BPM Suite 6.1 installation.

3. Copy this directory, and any subdirectories, to the **repositories** directory of the JBoss BPM Suite 6.2 installation:

```
cp -R ./ $BPM_6.2_INSTALLATION/repositories/
```

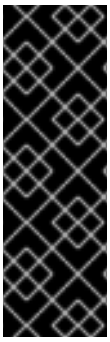
3.1.3. Upgrading the Database from 6.1.x to 6.2

Ensure the database has been upgraded as documented in Section 3.4, “Database Migration for 6.x Instances”.

3.2. MIGRATING FROM 6.0.X TO 6.1

3.2.1. Migrating Client Application APIs from 6.0.x to 6.1

In this section, we will discuss the migration steps that are specific to migrating from Red Hat JBoss BPM Suite 6.0.x to 6.1.



IMPORTANT

When using Red Hat JBoss BRMS/BPM Suite prior to 6.1 deployed on Red Hat JBoss EAP, it was possible to load JBoss BRMS/BPM Suite libraries from the JBoss EAP modules by correctly configuring **jboss-deployment-structure.xml**. In JBoss BRMS and JBoss BPM Suite 6.1, there are no modules including JBoss BRMS/BPM Suite libraries, hence the application developer is responsible to ensure all dependencies will be resolvable during runtime. For more information, see the relevant section in the *Red Hat JBoss BPM Suite Development Guide*.

Remove Old API

As noted in section Section 2.2.3, “API and Backwards Compatibility” , 6.1 is *not* backwards compatible with the 5.x API, while 6.0.x was. Therefore, when migrating from 6.0.x to 6.1 version, you will need to remove all references to the old API.

Modify Your (Custom) persistence.xml

There have been several changes in the classes present in the **persistence.xml** file. If you have been using a custom file for persistence, then you need to be aware of these changes and implement them:

- The Task Audit classes have changed from:

```
<class>org.jbpm.services.task.audit.impl.model.GroupAuditTaskImpl</class>
<class>org.jbpm.services.task.audit.impl.model.HistoryAuditTaskImpl</class>
<class>org.jbpm.services.task.audit.impl.model.UserAuditTaskImpl</class>
```

  to:

```
<class>org.jbpm.services.task.audit.impl.model.AuditTaskImpl</class>
```

- The Event class has changed from:

```
<class>org.jbpm.services.task.audit.TaskEventImpl</class>
```

to:

```
<class>org.jbpm.services.task.audit.impl.model.TaskEventImpl</class>
```

- There is a new entry for the deployment store:

```
<class>org.jbpm.kie.services.impl.store.DeploymentStoreEntry</class>
```

- There are some new mapping files that need adding:

```
<mapping-file>META-INF/Taskorm.xml</mapping-file>
<mapping-file>META-INF/Servicesorm.xml</mapping-file>
<mapping-file>META-INF/TaskAuditorm.xml</mapping-file>
```

Migrate Classes

6.1 has migrated some API classes to the Services API (and access to those classes has been removed). If you were using these API's because no alternative was available earlier, migrate to these new classes:

```
old --> new
org.kie.internal.deployment.DeployedUnit; -->
org.jbpm.services.api.model.DeployedUnit;
org.kie.internal.deployment.DeploymentService; -->
org.jbpm.services.api.DeploymentService;
org.kie.internal.deployment.DeploymentUnit; -->
org.jbpm.services.api.model.DeploymentUnit;
org.jbpm.kie.services.api.IdentityProvider; -->
org.kie.internal.identity.IdentityProvider;
```

Make Changes to Module Ids in Your POMs

The following ids have changed.

```
old --> new
kie-services-jaxb --> kie-remote-jaxb
kie-services-client --> kie-remote-client
kie-services-remote --> kie-remote-services
```

Review JMS Port Changes

In 6.1 4447 is the JNDI port to discover JMS **`ConnectionFactory`** connection and to get queues. Port 5445 is used by default for unsecured JMS communication. Port 5446 is used by default for SSL secured JMS communication.

In 6.0.x only port 4447 was necessary to get the JMS **`ConnectionFactory`**. This factory provided JMS queues and no other port setting was necessary.

Verify Location of Deployment Units

In 6.1, deployment units are located inside the database. In 6.0, these were stored in the GIT repository. If you want to continue storing your deployment units within the GIT repo, you must use the system property org.kie.git.deployments.enabled and set it to true.

This change is backward compatible. The first time a deployment is run on a new setup it will read up from **system.git** and deploy into runtime (in database storage). At the same time it will remove it from system GIT to clean it up.

Modify Remote Class API Code

Between 6.0.x and 6.1, the Remote Class API has changed considerably. For example, in 6.0.x, you would use the following code for creating a JMS based runtime engine:

```
import org.kie.services.client.api.command.RemoteRuntimeEngine;
...
RemoteJmsRuntimeEngineBuilder rjmsreBuilder =
RemoteJmsRuntimeEngineFactory.newBuilder().addJbossServerHostName(host).ad
dHostName(host).addUserName(username).addPassword(password).addTimeout(tim
eout).addDeploymentId(deploymentId).addExtraJaxbClasses(classes);

if( useSsl ) {

rjmsreBuilder.addKeystoreLocation("client0.keystore.jks").addKeystorePassw
ord(storePassword).addTruststorePassword(storePassword).useKeystoreAsTrust
store().addJmsConnectorPort(sslPort);
  } else {
    rjmsreBuilder.useSsl(false).addJmsConnectorPort(port);
  }

RemoteRuntimeEngine remoteRuntimeEngine = rjmsreBuilder.build();
```

or the following code for creating a REST based runtime engine:

```
remoteRuntimeEngine =
RemoteRestRuntimeEngineFactory.newBuilder().addUrl(url).addUserName(userna
me).addPassword(password).addDeploymentId(deploymentId).addExtraJaxbClasse
s(classes).build();
```

This code should be modified as the Remote Class API defines new generalized interface for remote runtime engine: org.kie.api.runtime.manager.RuntimeEngine. To create the runtime engines in 6.1, use the following code (JMS):

```
// for JMS
import org.kie.api.runtime.manager.RuntimeEngine;
...

RemoteJmsRuntimeEngineBuilder rjmsreBuilder =
RemoteRuntimeEngineFactory.newJmsBuilder().addJbossServerHostName(host).ad
dHostName(host).addUserName(username).addPassword(password).addTimeout(tim
eout).addDeploymentId(deploymentId).addExtraJaxbClasses(classes);

// 5446 is the default secured jms port, 5445 the default unsecured jms
port
if ( port == 5446 ) {

rjmsreBuilder.addKeystoreLocation("client0.keystore.jks").addKeystorePassw
ord(storePassword).addTruststorePassword(storePassword).useKeystoreAsTrust
store().addJmsConnectorPort(port);

  sslStatus = "enabled";
```

```
} else if (port == 5445) {
  rjmsreBuilder
    .useSsl(false)
    .disableTaskSecurity()
    .addJmsConnectorPort(port);
  sslStatus = "disabled";
  } else {
    throw new IllegalArgumentException("Unsupported jms port, valid ones
are secured 5446 and unsecured 5445.");
  }

  RuntimeEngine remoteRuntimeEngine = rjmsreBuilder.build();
```

and the following code for REST:

```
remoteRuntimeEngine =
remoteRuntimeEngineFactory.newRestBuilder().addUrl(url).addUserName(userna
me).addPassword(password).addDeploymentId(deploymentId).addExtraJaxbClasse
s(classes).build();
```

Rename the AuditLogService Class
If you were using the **`AuditLogService`** in 6.0.x branch, migrate that to the renamed class: **`AuditService`**. Method names have changed accordingly: **`remoteRuntimeEngine.getAuditLogService()`** should be changed to **`remoteRuntimeEngine.getAuditService()`**.

3.2.2. Migrating Business Central Project, Repository and Artifacts from 6.0.x to 6.1

Use the following procedure to move your existing Red Hat JBoss BPM Suite 6.0.x git projects (**`.niogit`** folder) and maven local dependencies (**`bin/repositories`**) to a new JBoss BPM Suite 6.1 installation.

Note, the example commands used in the following procedures are based on migrating from JBoss BPM Suite 6.0.3 to 6.1.

Procedure 3.3. Migrate a Single Project

1. Turn off JBoss BPM Suite 6.0.x and JBoss BPM Suite 6.1 instances.
2. Navigate to **`.niogit`** folder of JBoss BPM Suite 6.0.x installation.
3. Clone the repository where desired project is located.

   ```
   $ git clone repository603.git
   ```

4. Navigate to the JBoss BPM Suite 6.1.0 **`niogit`** folder.
5. Clone the repository where you want to migrate the 6.0.x project.

   ```
   $ git clone repository610.git
   ```

6. Copy the project from 6.0.x cloned repository to 6.1.x cloned repository.

```
$ cp -R /path/to/6.0.3/project /path/to/6.1.0/repository
```

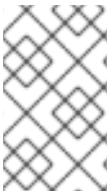
7. Navigate to the 6.1.0 cloned repository.

```
$ cd /path/to/6.1.0/repository
```

8. Commit the newly added 6.0.x project to your new 6.1.0 repository.

```
$ git add ./copied-6.0.3-project/*
$ git commit -m "migrating 6.0.3 project to 6.1.0 repository"
$ git push
```

9. Start JBoss BPM Suite 6.1.0. The 6.0.3 project should be successfully migrated and visible under the specified repository.



NOTE

The outcome of the above procedure may also be achieved by using the eGit plugin for Red Hat JBoss Developer Studio. See the relevant section in the *Red Hat JBoss BPM Suite Development Guide*.

Migrating a Repository

The following procedure demonstrates how to migrate a selected JBoss BPM Suite 6.0.x repository to JBoss BPM Suite 6.1.0 installation.

Procedure 3.4. Clone and Migrate a Repository

1. Turn on JBoss BPM Suite 6.1.0.
2. Log in to Business Central and navigate to **Authoring** → **Administration** → **Repositories** → **Clone Repository**.
3. Fill in the form. For example:

```
Repository Name - MyOld603Repo
Organizational Unit - example
Git URL - file:///path/to/old/603/.niogit/repository.git
```

and press **Confirm**.

4. The repository should be now available for Authoring.

For more information about cloning repositories, see the *Red Hat JBoss BPM Suite Administration and Configuration Guide*.

Migrating a Maven Artifact

The Artifact Repository is an internal maven repository for JBoss BPM Suite. The default internal maven repository is created in a working directory of JBoss BPM Suite 6.1.0 installation, with the folder name `repositories/kie`.

Maven artifacts can be migrated using the GUI.

Procedure 3.5. Migrating Maven Artifacts using the GUI

1. Turn on JBoss BPM Suite 6.1.
2. Navigate to **Authoring** → **Artifact Repository**.
3. Upload the Artifact from your old 6.0.3 installation.

Alternatively, following procedure demonstrates how to migrate selected maven artifact from JBoss BPM Suite 6.0.x Artifact Repository to a JBoss BPM Suite 6.1.0 Artifact Repository, and assumes that the two Artifact Repositories are located on the same physical system.

Procedure 3.6. Migrating a Particular Artifact

1. Consider following KJAR, which was installed into JBoss BPM Suite 6.0.x Artifact Repository by Business Central.

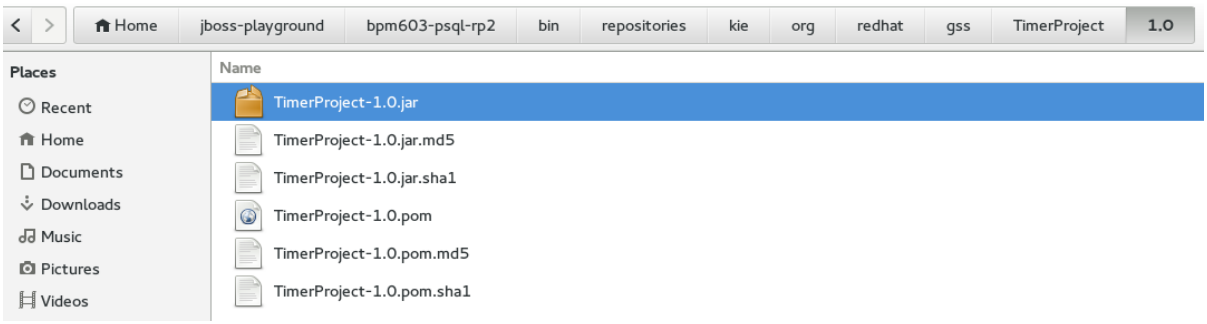


Figure 3.1. Installed KJAR

2. Copy this artifact to the JBoss BPM Suite 6.1.0 Artifact Repository. For example:

```
$ cp -R --parents /path/to/603/kjar/kie/org/redhat/gss/TimerProject/
/path/to/bpms-610-psql/bin/repositories/
```

The **`--parents`** argument will ensure that all the necessary folders (if missing) will be created in 6.1.0 too. In this case, it will honor the **`/org/redhat/gss/TimerProject`** path.

After copying, the 6.1.0 maven repository should look appear as follows:

Figure 3.2. Copied Artifact

3. Start the JBoss BPM Suite 6.1.0 installation and navigate to Artifact Repository. The copied artifact should be present as shown.

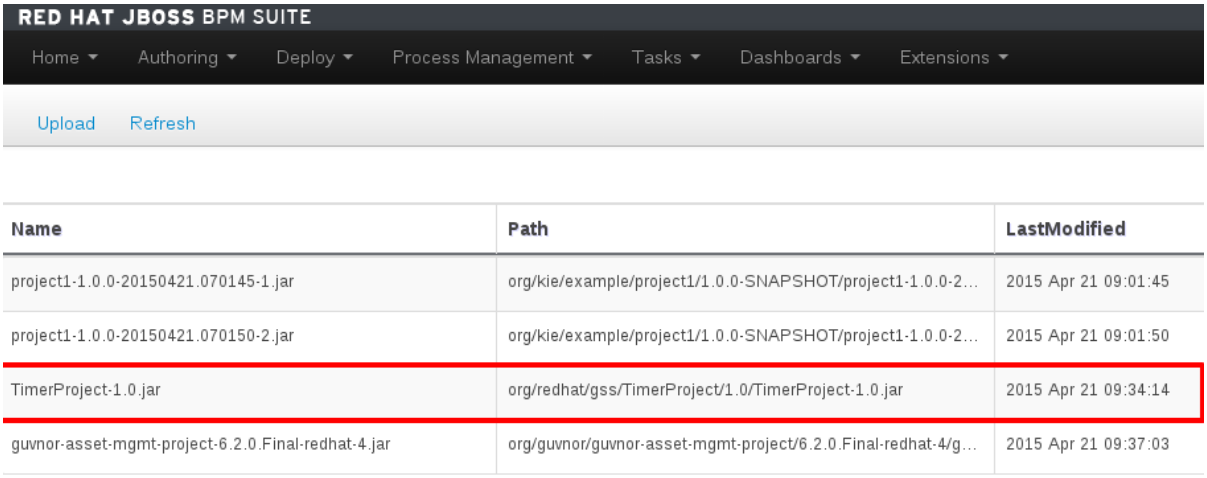


Figure 3.3. Artifact Repository

Migrating the `.niogit` Folder

In order to migrate the whole **`.niogit`** from 6.0.x to 6.1.0, set the org.uberfire.nio.git.dir property in 6.1.0 as follows.

```
$ ./standalone.sh -Dorg.uberfire.nio.git.dir=/path/to/6.0.3/.niogit
```

3.2.3. Upgrading the Database from 6.0.x to 6.1

Ensure that the database has been upgraded as documented in Section 3.4, “Database Migration for 6.x Instances”.

3.3. MIGRATING FROM 6.0.X TO 6.2

3.3.1. Migrating Client Application APIs from 6.0.x to 6.2

To migrate client applictions from JBoss BPM Suite 6.0.x to 6.2 the applications must incorporate all of the changes mentioned in both Section 3.2.1, “Migrating Client Application APIs from 6.0.x to 6.1” and Section 3.1.1, “Migrating Client Application APIs from 6.1.x to 6.2” .

3.3.2. Migrating Business Central Project, Repository, and Artifacts from 6.0.x to 6.2

Use the following procedure to move your existing Red Hat JBoss BPM Suite 6.0.x git projects (**`.niogit`** folder) and maven local dependencies (**`bin/repositories`**) to a new JBoss BPM Suite 6.2 installation.

Note, the example commands used in the following procedures are based on migrating from JBoss BPM Suite 6.0.3 to 6.2.

Procedure 3.7. Migrate a Single Project

1. Turn off JBoss BPM Suite 6.0.x and JBoss BPM Suite 6.2 instances.
2. Navigate to **`.niogit`** folder of JBoss BPM Suite 6.0.x installation.
3. Clone the repository where desired project is located.

```
$ git clone repository603.git
```

4. Navigate to the JBoss BPM Suite 6.2.0 **niogit** folder.
5. Clone the repository where you want to migrate the 6.0.x project.

```
$ git clone repository620.git
```

6. Copy the project from 6.0.x cloned repository to 6.2.x cloned repository.

```
$ cp -R /path/to/6.0.3/project /path/to/6.2.0/repository
```

7. Navigate to the 6.2.0 cloned repository.

```
$ cd /path/to/6.2.0/repository
```

8. Commit the newly added 6.0.x project to your new 6.2.0 repository.

```
$ git add ./copied-6.0.3-project/*
$ git commit -m "migrating 6.0.3 project to 6.2.0 repository"
$ git push
```

9. Start JBoss BPM Suite 6.2.0. The 6.0.3 project should be successfully migrated and visible under the specified repository.

NOTE

The outcome of the above procedure may also be achieved by using the eGit plugin for Red Hat JBoss Developer Studio. See the relevant section in the *Red Hat JBoss BPM Suite Development Guide*.

Migrating a Repository

The following procedure demonstrates how to migrate a selected JBoss BPM Suite 6.0.x repository to JBoss BPM Suite 6.2.0 installation.

Procedure 3.8. Clone and Migrate a Repository

1. Turn on JBoss BPM Suite 6.2.0.
2. Log in to Business Central and navigate to **Authoring** → **Administration** → **Repositories** → **Clone Repository**.
3. Fill in the form. For example:

```
Repository Name - MyOld603Repo
Organizational Unit - example
Git URL - file:///path/to/old/603/.niogit/repository.git
```

and press **Confirm**.

4. The repository should be now available for Authoring.

For more information about cloning repositories, see the *Red Hat JBoss BPM Suite Administration and Configuration Guide*.

Migrating a Maven Artifact

The Artifact Repository is an internal maven repository for JBoss BPM Suite. The default internal maven repository is created in a working directory of JBoss BPM Suite 6.2.0 installation, with the folder name `repositories/kie`.

Maven artifacts can be migrated using the GUI.

Procedure 3.9. Migrating Maven Artifacts using the GUI

1. Turn on JBoss BPM Suite 6.2.
2. Navigate to **Authoring** → **Artifact Repository**.
3. Upload the Artifact from your old 6.0.3 installation.

Alternatively, following procedure demonstrates how to migrate selected maven artifact from JBoss BPM Suite 6.0.x Artifact Repository to a JBoss BPM Suite 6.2.0 Artifact Repository, and assumes that the two Artifact Repositories are located on the same physical system.

Procedure 3.10. Migrating a Particular Artifact

1. Consider following KJAR, which was installed into JBoss BPM Suite 6.0.x Artifact Repository by Business Central.

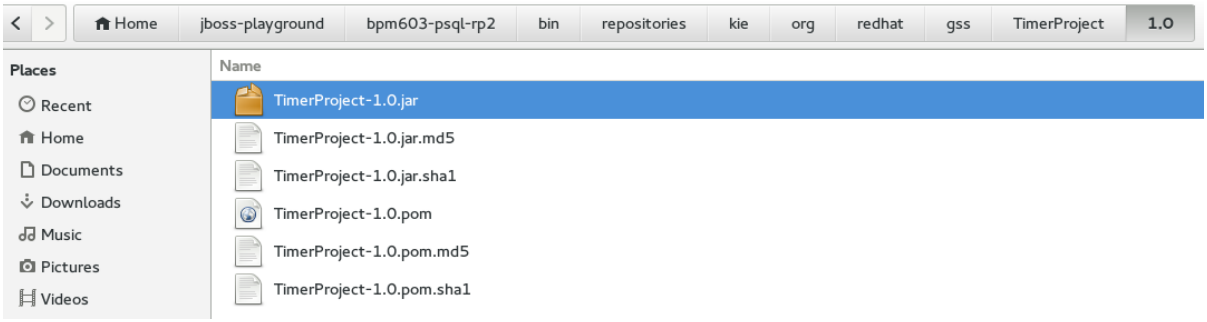


Figure 3.4. Installed KJAR

2. Copy this artifact to the JBoss BPM Suite 6.2.0 Artifact Repository. For example:

```
$ cp -R --parents /path/to/603/kjar/kie/org/redhat/gss/TimerProject/
/path/to/bpms-610-psql/bin/repositories/
```

The `--parents` argument will ensure that all the necessary folders (if missing) will be created in 6.2.0 too. In this case, it will honor the `/org/redhat/gss/TimerProject` path.

3. Start the JBoss BPM Suite 6.2.0 installation and navigate to Artifact Repository. The copied artifact should be present as shown.

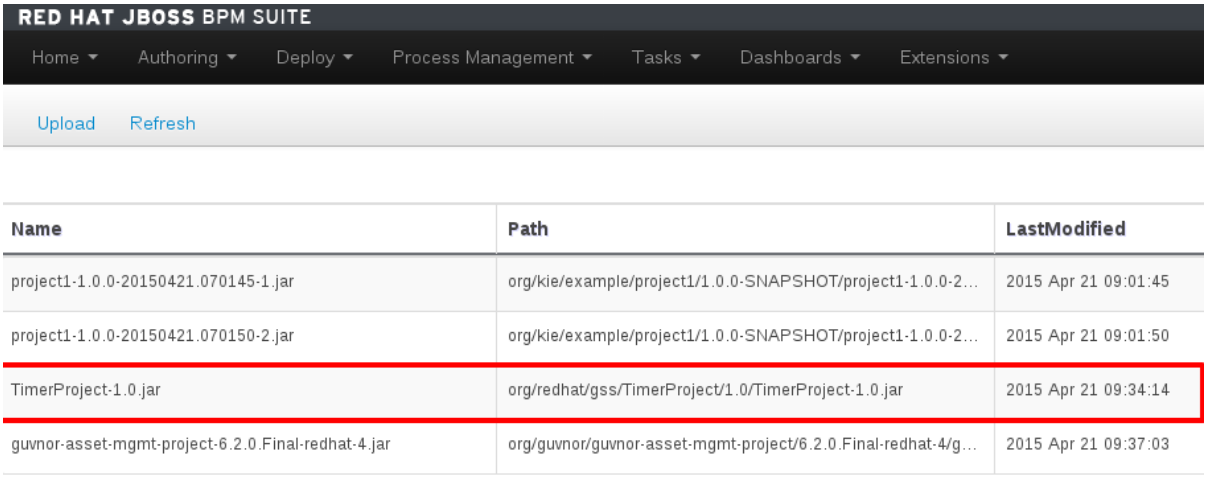


Figure 3.5. Artifact Repository

Migrating the `.niogit` Folder

In order to migrate the whole **`.niogit`** from 6.0.x to 6.2.0, set the org.uberfire.nio.git.dir property in 6.2.0 as follows.

```
$ ./standalone.sh -Dorg.uberfire.nio.git.dir=/path/to/6.0.3/.niogit
```

3.3.3. Upgrading the Database from 6.0.x to 6.2

Ensure that the database has been upgraded as documented in Section 3.4, “Database Migration for 6.x Instances”.

3.4. DATABASE MIGRATION FOR 6.X INSTANCES

Due to data model changes the database schema has also changed between minor versions of JBoss BPM Suite. A set of scripts has been included to ease this migration process; these scripts will generate the new tables and columns necessary, in addition to populating these columns where appropriate.

IMPORTANT

It is strongly recommended to backup the database before attempting any update, as this will provide a recoverable state should any issues arise during the update process.

1. Download the **`Red Hat JBoss BPM Suite Supplementary Tools`** from the Customer Portal.
2. Shutdown any Red Hat JBoss BPM Suite servers communicating with the database.
3. Navigate to the subdirectory corresponding to the type of database used. For instance, if a h2 database is in use (the default), then the scripts in the **h2** directory will be used in the subsequent steps.
4. Examine the contents of each script to be executed. While many of the scripts contain only database changes, there are others that require commands to be executed from a client with an open connection to the database.
5. Execute the scripts, in order, corresponding to the following table.

Table 3.1. Database Scripts by Version

Original JBoss BPM Suite Version	New JBoss BPM Suite Version	Script(s) to execute
6.0	6.2	`bpmsuite-6.0-to-6.1.sql`, `bpmsuite-6.1-to-6.2.sql`
6.1	6.2	`bpmsuite-6.1-to-6.2.sql`
6.0	6.1	`bpmsuite-6.0-to-6.1.sql`

6. Start the new JBoss BPM Suite server.

3.5. MIGRATING RUNNING PROCESSES

To migrate these processes follow the below steps:

1. Shutdown any active BPM Suite servers using the older version.

IMPORTANT

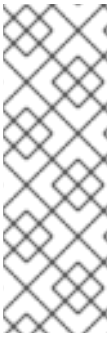
When the server is shutdown all processes will be terminated, and only the processes that have been persisted to the database will migrate smoothly. Before shutting down the BPM Suite server ensure that all rules are either waiting on a human task, so that they will be persisted, or not actively executing.

2. Ensure the instructions in Section 3.4, “Database Migration for 6.x Instances” have been followed to upgrade the database to the new version.

3. Start the new BPM Suite server, pointing to the same database that was previously in use.

4. Before executing any processes on the new server follow the instructions for copying the maven and git repositories that pertain to the migration. These are discussed in Section 3.1.2, “Migrating Business Central Project, Repository, and Artifacts from 6.1.x to 6.2” and Section 3.2.2, “Migrating Business Central Project, Repository and Artifacts from 6.0.x to 6.1” .

5. At this point the processes will be retrieved from the backing database and may be used as normal.



NOTE

Due to the database changes the server must be offline temporarily while the database is updated, and as such a true migration with no downtime does not exist. To minimize downtime it is recommended to use a load balancer in front of the BPM Suite servers and delegate requests to older instances until the full migration has completed. Once the database upgrade has completed, and the new JBoss BPM Suite servers started successfully, requests may be sent to these servers which will retrieve the persisted process.

CHAPTER 4. MIGRATION EXAMPLES

4.1. HELLO WORLD PROJECT MIGRATION

Let's start with the simplest migration example, a simple Hello World kind of process. This is just to get you started and make you comfortable with the migration process.

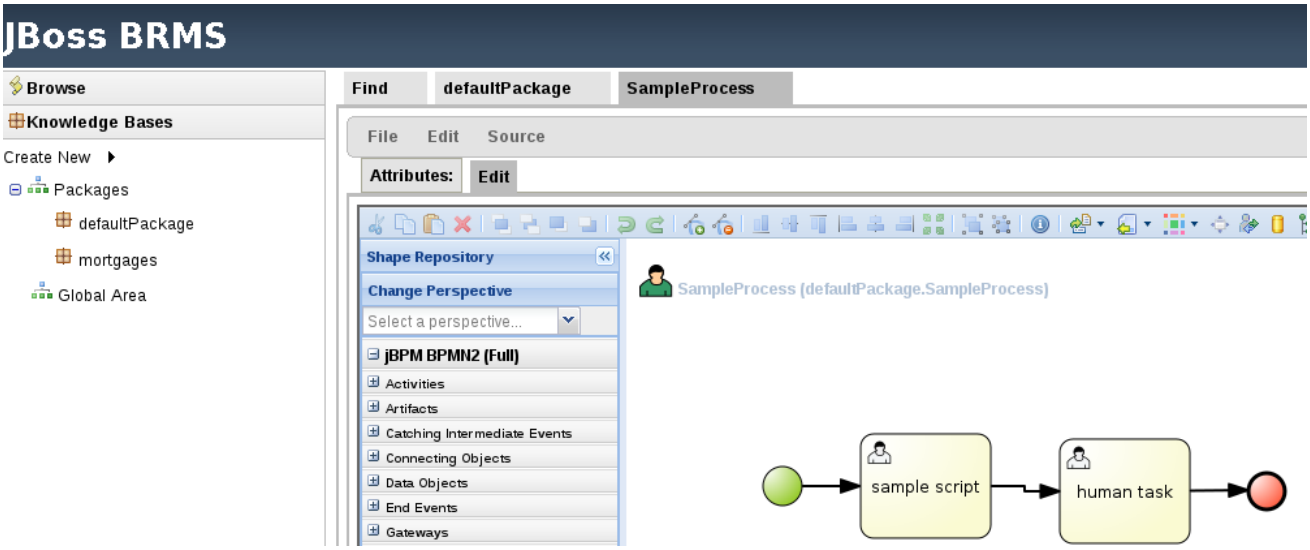
Data Migration

Start JBoss BRMS 5 with a *clean* repository and install the built-in samples. To make it more interesting, add one BPMN process with following structure:

```
START->SCRIPT_Task->Human_Task->End
```

.

Save the process and build the package. This is what it looks like in JBoss BRMS 5:



As discussed earlier, in order to automatically migrate assets from JBoss BRMS 5 to JBoss BPM Suite 6, you need to convert the JCR repository, storage for business assets in 5, to a GIT repository - storage for business assets in 6. Run the migration tool:

```
$ ./runMigration.sh -i /home/osiris/jboss-playground/jboss-eap-6.0-brms/bin/repository -o /home/osiris/Downloads/myGitRepo -r MyMigratedRepo
```

This example assumes that the JBoss BRMS 5 instance was started from the **bin** directory, using **./standalone.sh**, so by default, the **repository** directory and **repository.xml** will be created in this directory.

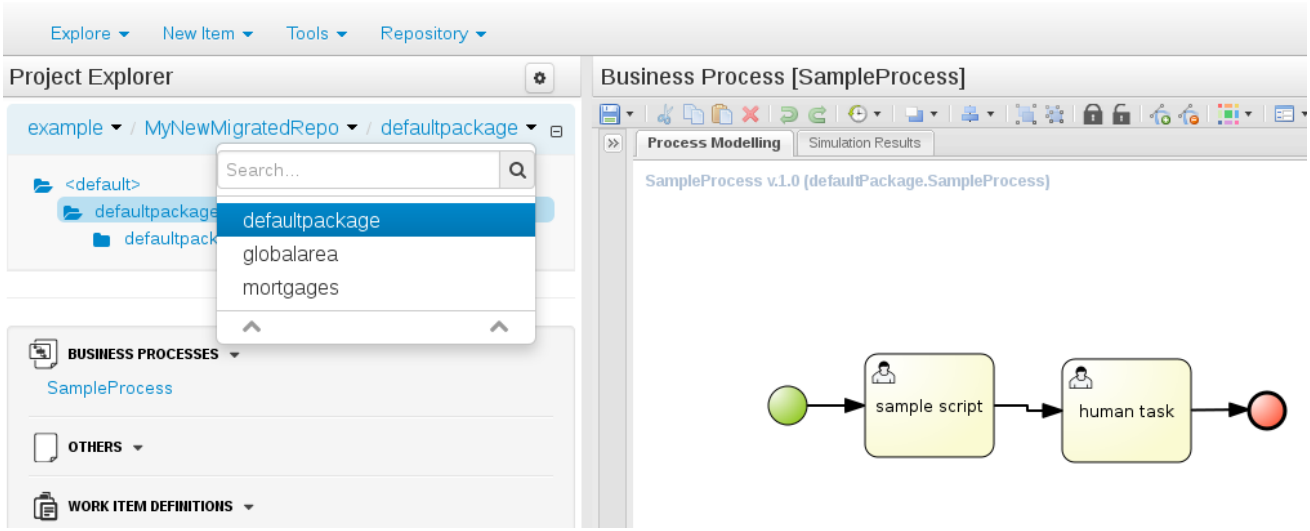
The destination (specified by the **-o** option) can be whatever dir you want it to be. The GIT repository is created under **myGitRepo** folder once the migration tool finishes its work.

NOTE

The destination folder must not exist when you run this command otherwise you will get an error.

Once done, clone this repository into JBoss BPM Suite 6.x via **Administration** perspective and clicking on **Repositories** and then **Clone Repository** links. Point it to **`file:///home/osiris/Downloads/myGitRepo/MyMigratedRepo.git`** to start the import.

It is now possible to build and deploy project in JBoss BPM Suite 6 and easily instantiate the process. This is what it should look like once successfully imported.



4.2. COOL STORE PROJECT MIGRATION

The 'Hello World' scenario is a useful migration demo to get your hands dirty. Let's make it more interesting by trying on a real-life project, such as the Cool Store Demo by Eric Schabell which is a reasonably complex project mimicking an online web shopping cart example.

This project provides both Red Hat JBoss BRMS 5 and Red Hat JBoss BPM Suite 6 repositories.

The JBoss BRMS 5 repository with included business-assets can be downloaded from the following location: https://github.com/jbossdemocentral/brms-coolstore-demo/tree/v1.0

Let's migrate this repository to JBoss BPM Suite 6 step by step.

1. Navigate to **`bin`** folder of your BRMS 5 installation - make sure there is no repository folder and no repository.xml file in this folder as we want to start with a clean repository for this example.

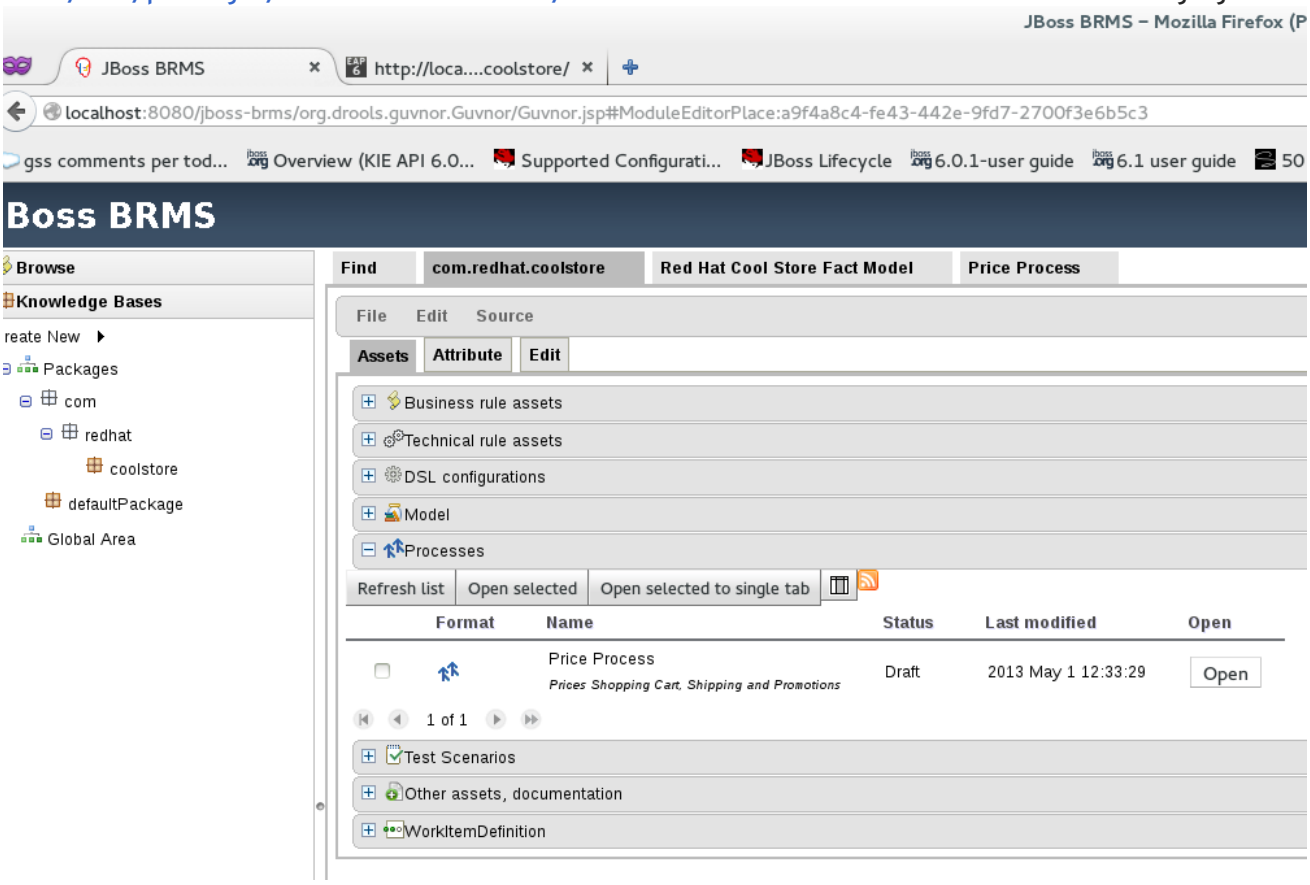
2. Start the server using **`./standalone.sh`** command. On JBoss EAP 5, start the server by executing the **`run.sh`** script.

3. In your browser, navigate to and login to http://localhost:8080/jboss-brms. If this is the first time you are running this, you will be prompted to install the default examples. Select **`No Thanks`** to installing these.

4. To start the migration of the business assets, let's first import the CoolStore repo into BRMS 5. Open a terminal window and execute the following commands: **`// get the repo $ git clone https://github.com/jbossdemocentral/brms-coolstore-demo.git $ cd brms-coolstore-demo // switch to the version 5 quickstart $ git checkout jdf-quickstart $ cd support // this unzips the 5.x repo in an XML format which we will now import $ unzip repository_export.zip`**

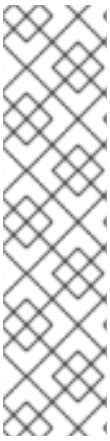
You now have the 5.x repository of the CoolStore demo in the file **`repository_export.xml`** which you can import in JBoss BRMS 5 via the Administration section of the Guvnor (Business Central in 5.x

branch of BRMS). Do this now, and once finished, there will be numerous assets present in the repository. You can view them at http://localhost:8080/jboss-brms/rest/packages/com.redhat.coolstore/. Your Guvnor should look similar to the following figure:



Build the package and the repository using the Guvnor interface and then migrate it using the migration tool.

Run the migration tool like this, from the **bin** folder of the JBoss BRMS 5 EAP install. **$ ./runMigration.sh -i /home/osiris/jboss-playground/jboss-eap-6.0-brms/bin/repository -o /home/osiris/Downloads/MigrateRepository -r "MyMigratedRepo"**



NOTE

1. JBoss BRMS 5 uses the JackRabbit repository for storage of its business assets. Its location is specified via the -i parameter.
2. JBoss BRMS 6/JBoss BPM Suite 6 uses the GIT repository for the storage of business assets. Its location is specified via the -o parameter
3. Name of the git repository is specified via the -r parameter.
4. Make sure that the destination folder **MigrateRepository** doesn't already exist.

After running the previous command successfully, your file structure should resemble: **$ pwd /home/osiris/jboss-playground/jboss-eap-6.0-brms/bin $ tree | grep repository |-- repository | |-- repository | |-- repository.xml |-- repository.xml**

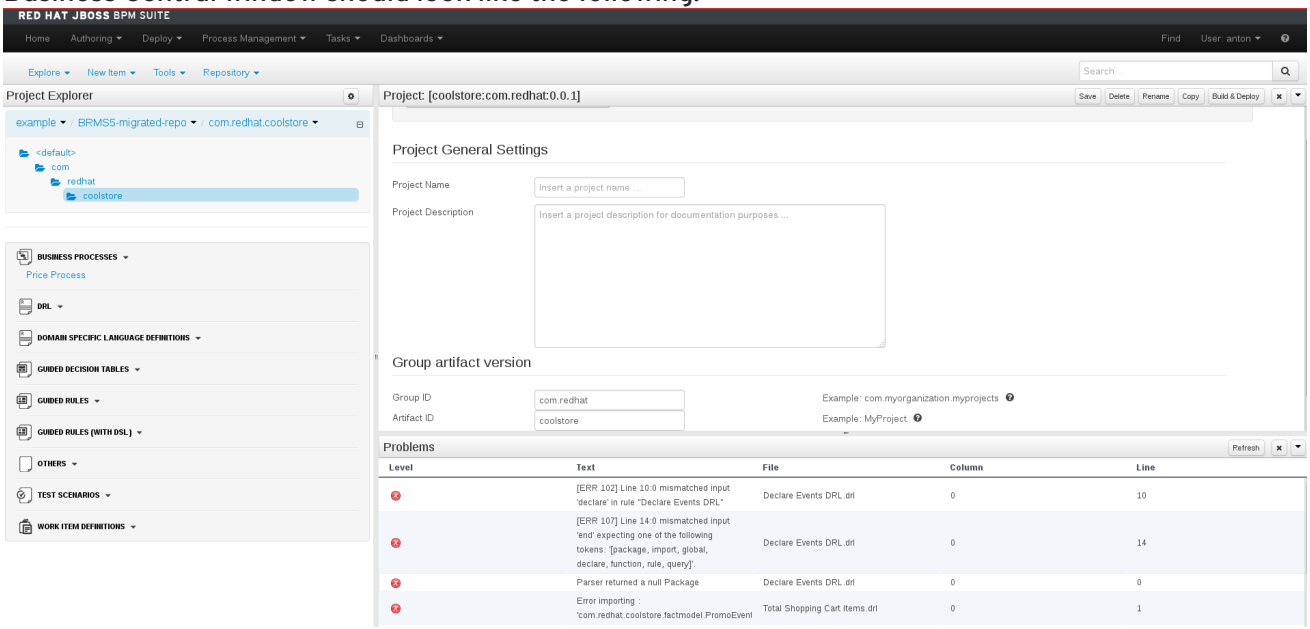
The **/bin/repository/repository.xml** file has been placed there manually - it is a copy of **/bin/repository.xml** file.

The output of this migration command will be the GIT repository located here: **/home/osiris/Downloads/MigrateRepository/MyMigratedRepo.git** This GIT repository is fully compatible with JBoss BPM Suite 6 and you will import it to Business Central later.

Migrating POJOs

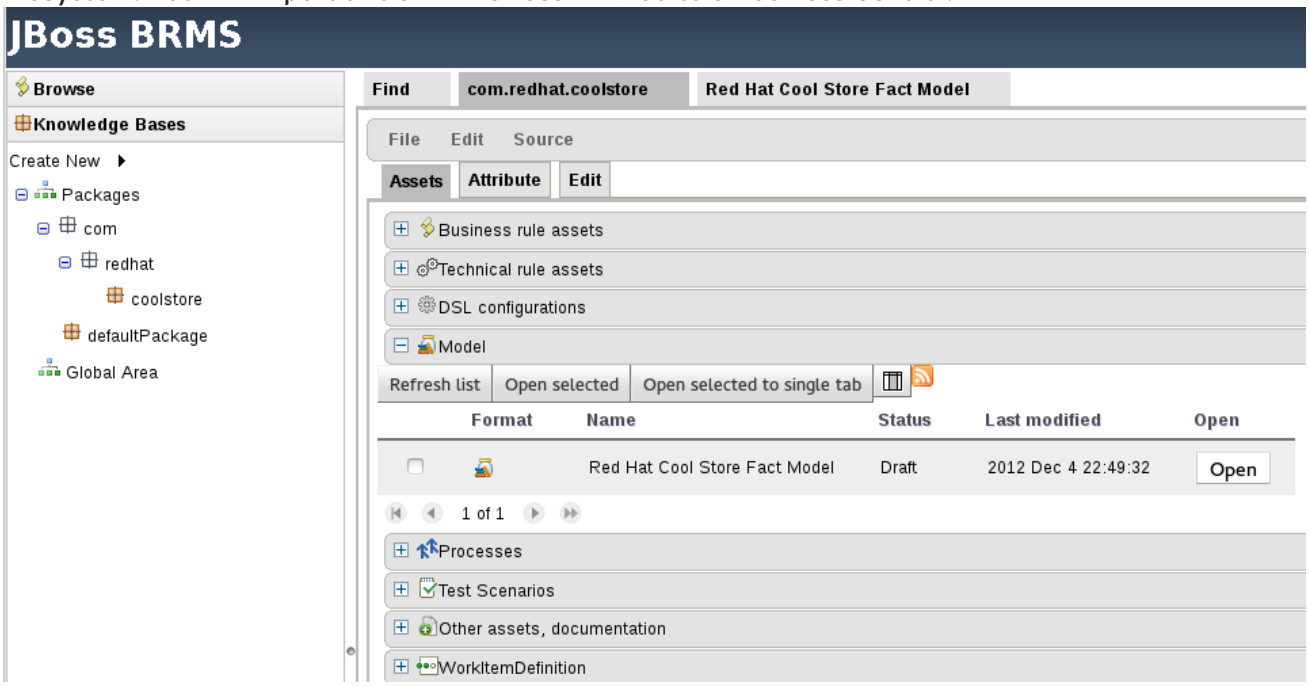
Although the migration tool migrates most assets, it doesn't migrate POJO model jars. These have to be migrated manually and your project won't build till this is done due to unresolved dependencies.

Open up your JBoss BPM Suite 6 instance, and import the GIT repository into it. Once imported, your Business Central window should look like the following:



As you can see, there are unresolved dependencies errors which will prevent this project from being built. We haven't imported the JAR with POJO model classes yet. We need to do this manually.

Log back into JBoss BRMS 5 Guvnor and locate the Fact Model JAR and download it to your local filesystem. You will import this JAR in JBoss BPM Suite 6 Business Central.



In Business Central, log into Authoring -> Artifact Repository. Press Upload button and locate the jar that you downloaded from JBoss BRMS 5.

NOTE

This JAR is not a Maven project, and it is a requirement that all JARs used by Business Central are Maven based. If they are not, you need to provide at least some Maven info in the form of Group, Artifact and Version (GAV). Fill in these values for the uploaded Fact Model JAR (these can be anything you want).

Include JAR as a dependency

You now need to place this JAR on the project's classpath so it will be available for every business asset located within the project. From Tools->Project Editor->Dependencies, select **`Add from repository`** and then select the freshly download JAR. Press **`Save`** and then Build & Deploy.

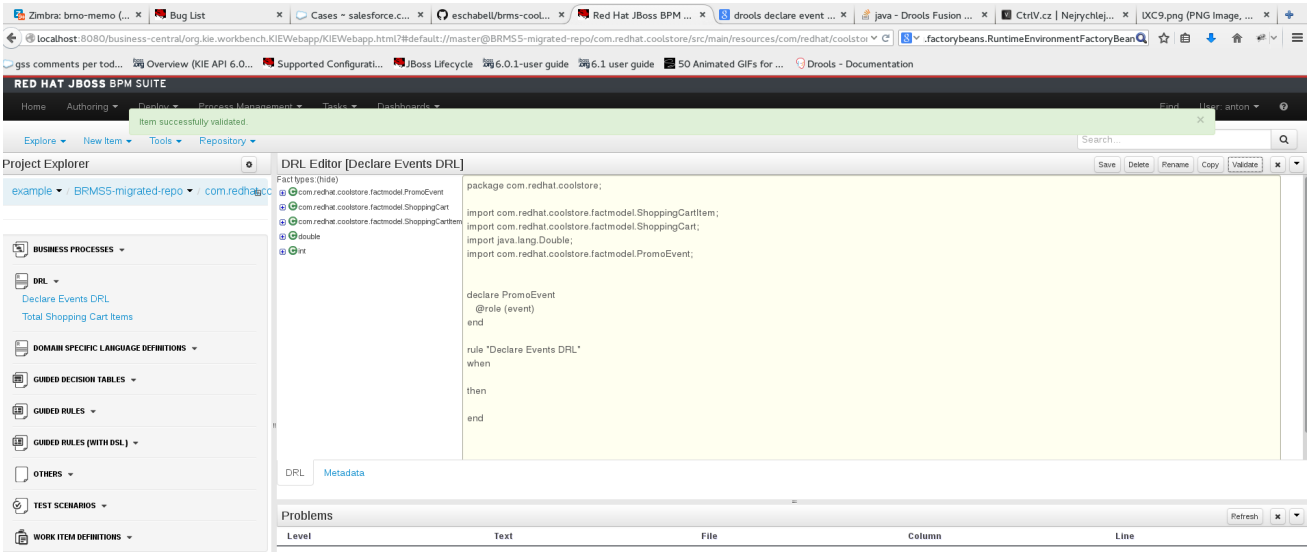
The unresolved dependencies errors will now go away. But you will have another issue.

Fixing syntax issues

If you look in the **`Problems`** section of Business Central, you will notice that the Build hasn't been successful due to an issue with **`Declare Events DRL`**. Looks like this rule has a syntax issue that needs fixing.

Although the BPMN2 and the DRL syntax didn't change between JBoss BRMS 5 and JBoss BPM Suite 6, what did change was stricter enforcing of syntax rules in version 6 than in version 5. This should normally not be an issue, and the system will always tell you where there is a mistake.

In version 6, you always need to include the basic keywords when defining a rule. Therefore, in the error you saw earlier, the missing keywords are: WHEN, THEN and END. It is easy to fix this!



Make these changes in the rule file and then press the **`Validate`** button. Your project should validate successfully, without any errors.

NOTE

The **`Validate`** button is a great tool when migrating content. With it, you are able to see if an asset can be built, before you actually try to build or deploy a whole project. The interface should tell you the root cause of any issue. More detailed information is also usually displayed in the server.log.

Now let's get back to Tools->Project Editor. Press the **`Build & Deploy`** - the project will build successfully. The migration is successful and all assets are validated and transferred in this process.

Migrating Selected Assets Manually

In some projects, you might not need to migrate the complete repository. The manual approach to transferring assets is fairly simple:

1. Create a new GIT repository in Business Central.
2. For DRL, create an empty rule and copy paste the content of the rule from JBoss BRMS 5.

   For BPMN processes, create a new process and use **`Import from BPMN2`** to import process definition from JBoss BRMS 5.

APPENDIX A. REVISION HISTORY

Note that revision numbers relate to the edition of this manual, not to version numbers of Red Hat JBoss BPM Suite.

Revision 6.2.0-4 **Thu Apr 28 2016** **Tomas Radej**

Updated with latest fixes.

Revision 6.2.0-3 **Tue Mar 29 2016** **Tomas Radej**

Build for release update 2 of JBoss BPM Suite.

Revision 6.2.0-2 **Mon Nov 30 2015** **Tomas Radej**

Added note about versions in Revision History, fixed changelog dates.

Revision 6.2.0-1 **Mon Nov 20 2015** **Tomas Radej**

Initial build for release 6.2.0 of JBoss BPM Suite.