Red Hat Decision Manager 7.1

Migrating from Red Hat JBoss BRMS 6.4 to Red Hat Decision Manager 7.1

Last Updated: 2020-04-30

Red Hat Decision Manager 7.1 Migrating from Red Hat JBoss BRMS 6.4 to Red Hat Decision Manager 7.1

Red Hat Customer Content Services
brms-docs@redhat.com

Abstract

This document describes how to migrate project data from Red Hat JBoss BRMS 6.4 to Red Hat Decision Manager 7.1.

Table of Contents

PREFACE

As a system administrator or business rules developer, you can migrate your existing project data in Red Hat JBoss BRMS 6.4 to Red Hat Decision Manager 7.1. Red Hat provides migration tools wherever possible to facilitate migration, but in some cases, manual migration or asset re-creation may be required.

Prerequisites

- Red Hat JBoss BRMS 6.4 is installed and contains artifacts that you want to migrate to Red Hat Decision Manager 7.1.
- A clean Red Hat Decision Manager 7.1 installation exists that does not contain a **.niogit** folder. If the Red Hat Decision Manager 7.1 installation contains a **.niogit** folder, the migration will fail. For installation options, see *Planning a Red Hat Decision Manager installation* .

IMPORTANT

If you have already migrated to Red Hat Decision Manager 7.0 and you want to migrate to Red Hat Decision Manager 7.1, follow the migration instructions in *Migrating from Red Hat Decision Manager 7.0 to Red Hat Decision Manager 7.1*.

CHAPTER 1. MIGRATION OVERVIEW

If you use Red Hat JBoss BRMS 6.4 and install Red Hat Decision Manager 7.1, you need to migrate the applications that you created in Red Hat JBoss BRMS 6.4 to the new product. Red Hat provides migration and upgrade tools wherever possible to facilitate migration.

The main change in Red Hat Decision Manager 7.1 that affects product migration is a new repository structure for Decision Central project data. Business assets that you created in Decision Central with Red Hat JBoss BRMS 6.4 must be migrated using the Decision Central migration tool provided with this release to accommodate a new data structure. Project data is restructured in Red Hat Decision Manager 7.1 so that each space (previously known as organizational unit) contains repositories that correspond to individual projects, instead of multiple projects in a single repository as in Red Hat JBoss BRMS 6.4. This improved structure means that you do not need to create and manage repositories in the new Decision Central and can focus on developing your business assets.

Migration paths are available for Decision Central distributions and for environments with Java projects moving to Red Hat Decision Manager 7.1.

IMPORTANT

If you are using a version older than Red Hat JBoss BRMS 6.4, migrate your projects to version 6.4 before migrating to Red Hat Decision Manager 7.1. For migration instructions, see the Red Hat JBoss BPM Suite 6.4 Migration Guide

For more information about updates and new features in this release, see the *Release notes for Red Hat Decision Manager 7.1*.

CHAPTER 2. DECISION CENTRAL DATA MIGRATION

You must migrate your Decision Central data from Red Hat JBoss BRMS 6.4 to Red Hat Decision Manager 7.1 using the Decision Central migration tool provided with this release to accommodate a new data structure in Red Hat Decision Manager 7.1. Additionally, if any of your applications interact with Decision Central spaces (previously known as organizational units), repositories, and projects through Knowledge Store REST API calls (**/decision-central/rest/**), you must update these API references according to the new endpoints supported in Red Hat Decision Manager 7.1.

Prerequisites

- A clean Red Hat Decision Manager 7.1 installation exists that does not contain a **.niogit** folder. If the Red Hat Decision Manager 7.1 installation contains a **.niogit** folder, the migration will fail. For installation options, see *Planning a Red Hat Decision Manager installation* .
- All Decision Central data for Red Hat JBoss BRMS 6.4, including a database used with it (if applicable), has been fully backed up. This is a precaution in case of problems during migration.

2.1. USE THE DECISION CENTRAL MIGRATION TOOL

Red Hat Decision Manager 7.1 contains a modified data structure and other feature changes that do not support a direct data migration from Red Hat JBoss BRMS 6.4. Therefore, Red Hat provides a Decision Central migration tool that enables you to migrate project data and configurations from Red Hat JBoss BRMS 6.4 to Red Hat Decision Manager 7.1. You can run the migration tool on the **.niogit** directory of your Decision Central distribution or on an external Git repository. The migration tool prepares your Decision Central data for the new data structure and features supported in Red Hat Decision Manager 7.1.

2.1.1. Using the migration tool on a .niogit directory

The **.niogit** directory of your Decision Central distribution contains all Decision Central data. You can use the Decision Central migration tool on your **.niogit** directory to prepare project data and system configurations for migration to Red Hat Decision Manager 7.1.

Prerequisites

- Decision Central is not running for either Red Hat JBoss BRMS 6.4 or Red Hat Decision Manager 7.1.
- Java 8 is installed and is available on the path where the project migration tool will be installed.

Procedure

1. Locate the **bin/.niogit** directory in your current Decision Central distribution for Red Hat JBoss BRMS 6.4 and locate the same directory in your new Red Hat Decision Manager 7.1 installation. Example:

   ```
   $ ~/$JBOSS_HOME/bin/.niogit
   ```

   ```
   $ ~/$RHDM_HOME/bin/.niogit
   ```

2. In your command terminal, copy the **.niogit** directory from your current Decision Central distribution for Red Hat JBoss BRMS 6.4 to the **bin** folder of the new Red Hat Decision Manager 7.1 installation directory:

   ```
   cp -r /$JBOSS_HOME/bin/.niogit  /$RHDM_HOME/bin/
   ```

   The copied **.niogit** folder is the directory that the migration tool will prepare for Red Hat Decision Manager 7.1. The original **.niogit** folder remains in your Red Hat JBoss BRMS 6.4 installation and will not be modified by the migration tool in this procedure.

3. In the Red Hat Decision Manager 7.1 **bin** directory, delete the **.index** directory. You must delete this folder because Red Hat Decision Manager 7.1 comes with certain updates that are not compatible with Red Hat JBoss BRMS 6.4. The **bin** directory will be re-indexed after the migration when you start Red Hat Decision Manager 7.1.

4. Note the path to the copied **.niogit** directory for Red Hat Decision Manager 7.1. The path will be required when you run the migration tool.

5. If you have specific **dependencies**, **repositories**, or **pluginRepositories** that you want to add, remove, or update as part of the project migration, create a **pom-migration.json** file containing these elements. When you run the migration tool, the tool will prompt you if you want to include this file in the migration.
   The **pom-migration.json** file requires the following sections:

   - **"dependencies":[]**
   - **"repositories-add":[]**
   - **"repositories-remove":[]**
   - **"repositories-update-urls":[]**
   - **"pluginRepositories-add":[]**
   - **"pluginRepositories-remove":[]**
   - **"pluginRepositories-update-urls":[]**

   If any of these elements do not apply to your project, include them as empty sections to prevent parsing errors.

   Example **pom-migration.json** file:

   ```
   {
     "dependencies":[

       {"groupId":"junit", "artifactId":"junit", "version":"4.12", "scope":"test"},
       {"groupId":"com.thoughtworks.xstream", "artifactId":"xstream", "version":"4.12",
   "scope":"test"}

     ],

     "repositories-add":[
       {
         "id":"jboss-public-repository-group",
         "name":"JBoss Public Repository Group",
   ```

```
      "url":"http://repository.jboss.org/nexus/content/groups/public/",
      "releasesEnabled":true,
      "releasesUpdatePolicy":"never",
      "snapshotEnabled":true,
      "snapshotUpdatePolicy":"never"
    },
    {
      "id":"kie-internal-staging-repository-group",
      "name":"KIE Internal Staging Repositories",
      "url":"https://origin-repository.jboss.org/nexus/content/groups/kie-internal-group",
      "releasesEnabled":true,
      "releasesUpdatePolicy":"never",
      "snapshotEnabled":false,
      "snapshotUpdatePolicy":"never"
    }
  ],

  "repositories-remove":[
    {
      "id":"productization-repository",
"url":"http://download.lab.bos.redhat.com/brewroot/repos/jb-ip-6.1-build/latest/maven/"
    }
  ],

  "repositories-update-urls":[
    {
      "id":"guvnor-m2-repo", "url":"http://localhost:8080/decision-central/maven3/"
    }
  ],

  "pluginRepositories-add":[

    {
      "id":"jboss-public-repository-group",
      "name":"JBoss Public Repository Group",
      "url":"http://repository.jboss.org/nexus/content/groups/public/",
      "releasesEnabled":true,
      "releasesUpdatePolicy":"never",
      "snapshotEnabled":true,
      "snapshotUpdatePolicy":"never"
    },
    {
      "id":"kie-internal-staging-repository-group",
      "name":"KIE Internal Staging Repositories",
      "url":"https://origin-repository.jboss.org/nexus/content/groups/kie-internal-group",
      "releasesEnabled":true,
      "releasesUpdatePolicy":"never",
      "snapshotEnabled":false,
      "snapshotUpdatePolicy":"never"
    }

  ],
  "pluginRepositories-remove":[],
  "pluginRepositories-update-urls":[]
}
```

Example **pom-migration.json** file with **dependencies** only:

```
{
  "dependencies":[

    {"groupId":"junit", "artifactId":"junit", "version":"4.12", "scope":"test"},
    {"groupId":"com.thoughtworks.xstream", "artifactId":"xstream", "version":"4.12",
"scope":"test"}

  ],
  "repositories-add":[],
  "repositories-remove":[],
  "repositories-update-urls":[],
  "pluginRepositories-add":[],
  "pluginRepositories-remove":[],
  "pluginRepositories-update-urls":[]
}
```

6. Navigate to the Software Downloads page in the Red Hat Customer Portal (login required), and select the product and version from the drop-down options:
    - **Product:** Decision Manager
    - **Version:** 7.1
7. Download **Red Hat Decision Manager 7.1.0 Add-Ons** and extract the downloaded **rhdm-7.1.0-add-ons.zip** file to a temporary directory.
8. In the extracted **rhdm-7.1.0-add-ons** folder, extract the **rhdm-7.1-migration-tool.zip** sub-folder. The migration tool is in the **bin** directory.
9. In your command terminal, navigate to the temporary directory where you extracted the **rhdm-7.1-migration-tool** folder and run the migration tool. The **$RHDM_NIOGIT_DIR** portion is the path to the **.niogit** directory that you previously copied to the Red Hat Decision Manager 7.1 installation.
   On Linux or UNIX-based systems:

```
$ cd $INSTALL_DIR/rhdm-7.1-migration-tool/bin
$ ./migration-tool.sh -t $RHDM_NIOGIT_DIR
```

   On Windows:

```
$ cd $INSTALL_DIR\rhdm-7.1-migration-tool\bin
$ migration-tool.bat -t $RHDM_NIOGIT_DIR
```

   In the command prompt that appears, the following options are displayed:

    - **Project structure migration**: Migrates the Red Hat JBoss BRMS 6.4 project repository structure to the new project-oriented structure used in Red Hat Decision Manager 7.1.
    - **System configuration directory structure migration**: Migrates the **system.git** repository structure used in Red Hat JBoss BRMS 6.4 to the new structure used in Red Hat Decision Manager 7.1. This migration option requires the project structure migration to be executed first.

- **POMs migration:** Updates **pom.xml** files with dependencies required for Red Hat Decision Manager 7.1. This migration option requires the project structure migration and system configuration directory structure migration to be executed first.
- **All:** Runs all migration options in sequence.
- **Exit:** Exits the migration tool.

10. Select the option to run **All** migrations in sequence.

    NOTE

    If you prefer to run one migration option at a time, select and run the first individual migration option. After the tool runs, re-run the Decision Central migration tool and select the next individual migration option in the sequence.

11. Enter **yes** each time you are prompted to run a specific migration option.
    For the POMs migration option, if you want to include a path to an external **pom-migration.json** file that you created previously, enter **yes** when prompted and enter the path.
12. After the tool finishes running, enter the option to **Exit** the migration tool.
    The **.niogit** directory structure is now compatible with Decision Central in Red Hat Decision Manager 7.1. Project directories are in separate repositories and all other related configurations have been migrated. You can navigate to the new **.niogit** directory to inspect the restructured contents.

2.1.2. Using the migration tool on an external Git repository

If you store Decision Central project data in a Git repository outside of your **.niogit** directory, you can also use the Decision Central migration tool on the external repository to prepare project data for Red Hat Decision Manager 7.1. Only files in project directories in the Git repository will be restructured by the migration tool.

Prerequisite

Java 8 is installed and is available on the path where the project migration tool will be installed.

Procedure

1. In a local directory, create a clone of the Git repository containing the projects to be migrated (if the repository has not been cloned already).

```
$ cd $REPO_DIR/
$ git clone $GIT_REPO_URL
```

2. Note the path to the cloned Git repository. The path will be required when you run the migration tool.
3. Create an output directory where a copy of the newly migrated repository or repositories will be placed after the migration tool runs. You can also use an existing directory as an output location. Note the path to this output directory. The migration tool operates on a copy of the cloned repository and will prompt you for the new output directory path when you run the tool.
4. If you have specific **dependencies**, **repositories**, or **pluginRepositories** that you want to add, remove, or update as part of the project migration, create a **pom-migration.json** file containing

these elements. When you run the migration tool, the tool will prompt you if you want to include this file in the migration.
The **pom-migration.json** file requires the following sections:

- **"dependencies":[]**
- **"repositories-add":[]**
- **"repositories-remove":[]**
- **"repositories-update-urls":[]**
- **"pluginRepositories-add":[]**
- **"pluginRepositories-remove":[]**
- **"pluginRepositories-update-urls":[]**

If any of these elements do not apply to your project, include them as empty sections to prevent parsing errors.

Example **pom-migration.json** file:

```
{
  "dependencies":[

    {"groupId":"junit", "artifactId":"junit", "version":"4.12", "scope":"test"},
    {"groupId":"com.thoughtworks.xstream", "artifactId":"xstream", "version":"4.12",
"scope":"test"}

  ],

  "repositories-add":[
    {
      "id":"jboss-public-repository-group",
      "name":"JBoss Public Repository Group",
      "url":"http://repository.jboss.org/nexus/content/groups/public/",
      "releasesEnabled":true,
      "releasesUpdatePolicy":"never",
      "snapshotEnabled":true,
      "snapshotUpdatePolicy":"never"
    },
    {
      "id":"kie-internal-staging-repository-group",
      "name":"KIE Internal Staging Repositories",
      "url":"https://origin-repository.jboss.org/nexus/content/groups/kie-internal-group",
      "releasesEnabled":true,
      "releasesUpdatePolicy":"never",
      "snapshotEnabled":false,
      "snapshotUpdatePolicy":"never"
    }
  ],

  "repositories-remove":[
    {
      "id":"productization-repository",
"url":"http://download.lab.bos.redhat.com/brewroot/repos/jb-ip-6.1-build/latest/maven/"
    }
  ],

  "repositories-update-urls":[
    {
      "id":"guvnor-m2-repo", "url":"http://localhost:8080/decision-central/maven3/"
    }
  ],

  "pluginRepositories-add":[

    {
      "id":"jboss-public-repository-group",
      "name":"JBoss Public Repository Group",
      "url":"http://repository.jboss.org/nexus/content/groups/public/",
      "releasesEnabled":true,
      "releasesUpdatePolicy":"never",
      "snapshotEnabled":true,
      "snapshotUpdatePolicy":"never"
    },
    {
      "id":"kie-internal-staging-repository-group",
      "name":"KIE Internal Staging Repositories",
      "url":"https://origin-repository.jboss.org/nexus/content/groups/kie-internal-group",
      "releasesEnabled":true,
      "releasesUpdatePolicy":"never",
      "snapshotEnabled":false,
      "snapshotUpdatePolicy":"never"
    }

  ],
  "pluginRepositories-remove":[],
  "pluginRepositories-update-urls":[]
}
```

Example **pom-migration.json** file with **dependencies** only:

```
{
  "dependencies":[

    {"groupId":"junit", "artifactId":"junit", "version":"4.12", "scope":"test"},
    {"groupId":"com.thoughtworks.xstream", "artifactId":"xstream", "version":"4.12",
"scope":"test"}

  ],
  "repositories-add":[],
  "repositories-remove":[],
  "repositories-update-urls":[],
  "pluginRepositories-add":[],
  "pluginRepositories-remove":[],
  "pluginRepositories-update-urls":[]
}
```

5. Navigate to the Software Downloads page in the Red Hat Customer Portal (login required), and select the product and version from the drop-down options:
   - **Product:** Decision Manager
   - **Version:** 7.1
6. Download **Red Hat Decision Manager 7.1.0 Add-Ons** and extract the downloaded **rhdm-7.1.0-add-ons.zip** file to a temporary directory.
7. In the extracted **rhdm-7.1.0-add-ons** folder, extract the **rhdm-7.1-migration-tool.zip** sub-folder. The migration tool is in the **bin** directory.
8. In your command terminal, navigate to the temporary directory where you extracted the **rhdm-7.1-migration-tool** folder and run the migration tool. The **$GIT_REPO_PATH** portion is the path to the cloned Git repository.
   On Linux or UNIX-based systems:

   ```
   $ cd $INSTALL_DIR/rhdm-7.1-migration-tool/bin
   $ ./migration-tool.sh -t $GIT_REPO_PATH
   ```

   On Windows:

   ```
   $ cd $INSTALL_DIR\rhdm-7.1-migration-tool\bin
   $ migration-tool.bat -t $GIT_REPO_PATH
   ```

9. In the command prompt that appears, enter the path to the output directory where the migrated copy of the repository will be placed. The migration tool operates on a copy of the cloned repository and will place the new repository or repositories in the output location that you specify.
   After you enter the output location, the migration tool prepares the repository copy and restructures all project directories to be compatible with Red Hat Decision Manager 7.1.

   In the command prompt, the following additional migration options are displayed:
   - **POMs migration:** Updates **pom.xml** files with dependencies required for Red Hat Decision Manager 7.1.
   - **All:** Runs all remaining migration options in sequence.
   - **Exit:** Exits the migration tool.

   NOTE

   The **Project structure migration** option is not displayed because that option was run automatically when you entered the output location to initiate the migration tool.

10. Select the option to run **ALL** migrations in sequence.

    NOTE

    If you prefer to run one migration option at a time, select and run the first individual migration option. After the tool runs, re-run the Decision Central migration tool and select the next individual migration option in the sequence.

11. Enter **yes** each time you are prompted to run a specific migration option.
    For the POMs migration option, if you want to include a path to an external **pom-migration.json** file that you created previously, enter **yes** when prompted and enter the path.

12. After the tool finishes running, enter the option to **Exit** the migration tool.
    The project directories in the specified output location are now separated into individual repositories compatible with Decision Central in Red Hat Decision Manager 7.1. The new project repositories are bare repositories with no working directory, and therefore do not show any content files. You can clone each repository to create non-bare repositories and inspect directory contents.

13. Log in to Decision Central for Red Hat Decision Manager 7.1.

14. For each project, create or select the space for the project and click **Import Project**.

15. Enter the **Repository URL** for the newly structured project repository. This URL may be the local file path to the output location if you are importing directly from the workstation where you ran the migration tool, or a GitHub URL or other file hosting location where you have uploaded the repository.
    Example: Local file location

    ```
    file:///$OUTPUT_DIR/loan-application.git
    ```

    Example: GitHub repository location

    ```
    https://github.com/$REPO/loan-application.git
    ```

    NOTE

    If you use Git **hooks** with your project data, you may need to update your **hooks** scripts to accommodate the migration.

2.2. UPDATE API REFERENCES TO DECISION CENTRAL KNOWLEDGE STORE

The Knowledge Store REST API has been deprecated in Red Hat Decision Manager 7.1, but certain endpoints are still supported so that you can continue to manage Decision Central resources. If any of your applications interact with Decision Central spaces (previously known as organizational units), repositories, and projects through Knowledge Store REST API calls (**/decision-central/rest/**), you must update these API references according to the new endpoints supported in Red Hat Decision Manager 7.1.

Procedure

Wherever applicable in your application code, replace any legacy REST API calls to the Decision Central Knowledge Store in the format **http://<SERVER>:<PORT>/decision-central/rest/<ENDPOINT>** with the new corresponding Knowledge Store REST API calls listed in Section 2.2.1, “Supported Knowledge Store REST API endpoints”.

As a result of the restructured content in Red Hat Decision Manager 7.1, the endpoints may differ between the legacy Decision Central REST API calls and the corresponding REST API calls in Red Hat Decision Manager 7.1. Be sure to use the new endpoints to replace your legacy Decision Central REST API calls. If you cannot find a corresponding REST API call in the list to replace a legacy call, then that call is no longer supported in Red Hat Decision Manager 7.1.

The following example is an update to a Knowledge Store REST API call for spaces.

Legacy Knowledge Store REST API call:

```
http://localhost:8080/decision-central/rest/organizational-units
```

New Knowledge Store REST API call:

```
http://localhost:8080/decision-central/rest/spaces
```

2.2.1. Supported Knowledge Store REST API endpoints

The Knowledge Store REST API provides endpoints for managing spaces and projects in Red Hat Decision Manager and for retrieving information about previous Knowledge Store REST API requests, or *jobs*.

2.2.1.1. Spaces

The Knowledge Store REST API supports the following endpoints for managing spaces in Decision Central. The Knowledge Store REST API base URL is **http://SERVER:PORT/decision-central/rest/**. All requests require HTTP Basic authentication or token-based authentication for the **rest-all** user role.

[GET] /spaces

Returns all spaces in Decision Central.

Example server response (JSON)

```
[
  {
    "name": "MySpace",
    "description": null,
    "projects": [
      {
        "name": "Employee_Rostering",
        "spaceName": "MySpace",
        "groupId": "employeerostering",
        "version": "1.0.0-SNAPSHOT",
        "description": "Employee rostering problem optimisation using Planner. Assigns employees to
shifts based on their skill.",
        "publicURIs": [
          {
            "protocol": "git",
            "uri": "git://localhost:9418/MySpace/example-Employee_Rostering"
          },
          {
            "protocol": "ssh",
            "uri": "ssh://localhost:8001/MySpace/example-Employee_Rostering"
          }
        ]
      },
      {
        "name": "Mortgage_Process",
        "spaceName": "MySpace",
        "groupId": "mortgage-process",
        "version": "1.0.0-SNAPSHOT",
        "description": "Getting started loan approval process in BPMN2, decision table, business
rules, and forms.",
        "publicURIs": [
          {
            "protocol": "git",
            "uri": "git://localhost:9418/MySpace/example-Mortgage_Process"
          },
          {
            "protocol": "ssh",
            "uri": "ssh://localhost:8001/MySpace/example-Mortgage_Process"
          }
        ]
      }
    ],
    "owner": "admin",
    "defaultGroupId": "com.myspace"
  },
  {
    "name": "MySpace2",
    "description": null,
    "projects": [
      {
        "name": "IT_Orders",
        "spaceName": "MySpace",
        "groupId": "itorders",
        "version": "1.0.0-SNAPSHOT",
        "description": "Case Management IT Orders project",
        "publicURIs": [
          {
            "protocol": "git",
            "uri": "git://localhost:9418/MySpace/example-IT_Orders-1"
          },
          {
            "protocol": "ssh",
            "uri": "ssh://localhost:8001/MySpace/example-IT_Orders-1"
          }
        ]
      }
    ],
    "owner": "admin",
    "defaultGroupId": "com.myspace"
  }
]
```

[GET] /spaces/{spaceName}

Returns information about a specified space.

Table 2.1. Request parameters

Name	Description	Type	Requirement
spaceName	Name of the space to be retrieved	String	Required

Example server response (JSON)

```
{
  "name": "MySpace",
  "description": null,
  "projects": [
    {
      "name": "Mortgage_Process",
      "spaceName": "MySpace",
      "groupId": "mortgage-process",
      "version": "1.0.0-SNAPSHOT",
      "description": "Getting started loan approval process in BPMN2, decision table, business rules,
and forms.",
      "publicURIs": [
        {
          "protocol": "git",
          "uri": "git://localhost:9418/MySpace/example-Mortgage_Process"
        },
        {
          "protocol": "ssh",
          "uri": "ssh://localhost:8001/MySpace/example-Mortgage_Process"
        }
      ]
    },
    {
      "name": "Employee_Rostering",
      "spaceName": "MySpace",
      "groupId": "employeerostering",
      "version": "1.0.0-SNAPSHOT",
      "description": "Employee rostering problem optimisation using Planner. Assigns employees to
shifts based on their skill.",
      "publicURIs": [
        {
          "protocol": "git",
          "uri": "git://localhost:9418/MySpace/example-Employee_Rostering"
        },
        {
          "protocol": "ssh",
          "uri": "ssh://localhost:8001/MySpace/example-Employee_Rostering"
        }
      ]
    },
    {
      "name": "Evaluation_Process",
      "spaceName": "MySpace",
      "groupId": "evaluation",
      "version": "1.0.0-SNAPSHOT",
      "description": "Getting started Business Process for evaluating employees",
      "publicURIs": [
        {
          "protocol": "git",
          "uri": "git://localhost:9418/MySpace/example-Evaluation_Process"
        },
        {
          "protocol": "ssh",
          "uri": "ssh://localhost:8001/MySpace/example-Evaluation_Process"
          }
        ]
      },
      {
        "name": "IT_Orders",
        "spaceName": "MySpace",
        "groupId": "itorders",
        "version": "1.0.0-SNAPSHOT",
        "description": "Case Management IT Orders project",
        "publicURIs": [
          {
            "protocol": "git",
            "uri": "git://localhost:9418/MySpace/example-IT_Orders"
          },
          {
            "protocol": "ssh",
            "uri": "ssh://localhost:8001/MySpace/example-IT_Orders"
          }
        ]
      }
    ],
    "owner": "admin",
    "defaultGroupId": "com.myspace"
  }
```

[POST] /spaces

Creates a space in Decision Central.

Table 2.2. Request parameters

Name	Description	Type	Requirement
body	The **name**, **description**, **owner**, **defaultGroupId**, and any other components of the new space	Request body	Required

Example request body (JSON)

```
{
  "name": "NewSpace",
  "description": "My new space.",
  "owner": "admin",
  "defaultGroupId": "com.newspace"
}
```

Example server response (JSON)

```
{
  "jobId": "1541016978154-3",
  "status": "APPROVED",
  "spaceName": "NewSpace",
  "owner": "admin",
  "defaultGroupId": "com.newspace",
  "description": "My new space."
}
```

[DELETE] /spaces/{spaceName}

Deletes a specified space from Decision Central.

Table 2.3. Request parameters

Name	Description	Type	Requirement
spaceName	Name of the space to be deleted	String	Required

Example server response (JSON)

```
{
  "jobId": "1541127032997-8",
  "status": "APPROVED",
  "spaceName": "MySpace",
  "owner": "admin",
  "description": "My deleted space.",
  "repositories": null
}
```

2.2.1.2. Projects

The Knowledge Store REST API supports the following endpoints for managing, building, and deploying projects in Decision Central. The Knowledge Store REST API base URL is **http://SERVER:PORT/decision-central/rest/**. All requests require HTTP Basic authentication or token-based authentication for the **rest-all** user role.

[GET] /spaces/{spaceName}/projects

Returns projects in a specified space.

Table 2.4. Request parameters

Name	Description	Type	Requirement
spaceName	Name of the space for which you are retrieving projects	String	Required

Example server response (JSON)

```
[
  {
    "name": "Mortgage_Process",
    "spaceName": "MySpace",
    "groupId": "mortgage-process",
    "version": "1.0.0-SNAPSHOT",
    "description": "Getting started loan approval process in BPMN2, decision table, business rules,
and forms.",
    "publicURIs": [
      {
        "protocol": "git",
        "uri": "git://localhost:9418/MySpace/example-Mortgage_Process"
      },
      {
        "protocol": "ssh",
        "uri": "ssh://localhost:8001/MySpace/example-Mortgage_Process"
      }
    ]
  },
  {
    "name": "Employee_Rostering",
    "spaceName": "MySpace",
    "groupId": "employeerostering",
    "version": "1.0.0-SNAPSHOT",
    "description": "Employee rostering problem optimisation using Planner. Assigns employees to
shifts based on their skill.",
    "publicURIs": [
      {
        "protocol": "git",
        "uri": "git://localhost:9418/MySpace/example-Employee_Rostering"
      },
      {
        "protocol": "ssh",
        "uri": "ssh://localhost:8001/MySpace/example-Employee_Rostering"
      }
    ]
  },
  {
    "name": "Evaluation_Process",
    "spaceName": "MySpace",
    "groupId": "evaluation",
    "version": "1.0.0-SNAPSHOT",
    "description": "Getting started Business Process for evaluating employees",
    "publicURIs": [
      {
        "protocol": "git",
        "uri": "git://localhost:9418/MySpace/example-Evaluation_Process"
      },
      {
        "protocol": "ssh",
        "uri": "ssh://localhost:8001/MySpace/example-Evaluation_Process"
      }
    ]
  },
  {
    "name": "IT_Orders",
    "spaceName": "MySpace",
    "groupId": "itorders",
    "version": "1.0.0-SNAPSHOT",
    "description": "Case Management IT Orders project",
    "publicURIs": [
      {
        "protocol": "git",
        "uri": "git://localhost:9418/MySpace/example-IT_Orders"
      },
      {
        "protocol": "ssh",
        "uri": "ssh://localhost:8001/MySpace/example-IT_Orders"
      }
    ]
  }
]
```

[GET] /spaces/{spaceName}/projects/{projectName}

Returns information about a specified project in a specified space.

Table 2.5. Request parameters

Name	Description	Type	Requirement
spaceName	Name of the space where the project is located	String	Required
projectName	Name of the project to be retrieved	String	Required

Example server response (JSON)

```
{
  "name": "Employee_Rostering",
  "spaceName": "MySpace",
  "groupId": "employeerostering",
  "version": "1.0.0-SNAPSHOT",
  "description": "Employee rostering problem optimisation using Planner. Assigns employees to
shifts based on their skill.",
  "publicURIs": [
    {
      "protocol": "git",
      "uri": "git://localhost:9418/MySpace/example-Employee_Rostering"
    },
    {
      "protocol": "ssh",
      "uri": "ssh://localhost:8001/MySpace/example-Employee_Rostering"
    }
  ]
}
```

[POST] /spaces/{spaceName}/projects

Creates a project in a specified space.

Table 2.6. Request parameters

Name	Description	Type	Requirement
spaceName	Name of the space in which the new project will be created	String	Required
body	The **name**, **groupId**, **version**, **description**, and any other components of the new project	Request body	Required

Example request body (JSON)

```
{
  "name": "Employee_Rostering",
  "groupId": "employeerostering",
  "version": "1.0.0-SNAPSHOT",
  "description": "Employee rostering problem optimisation using Planner. Assigns employees to shifts based on their skill."
}
```

Example server response (JSON)

```
{
  "jobId": "1541017411591-6",
  "status": "APPROVED",
  "spaceName": "MySpace",
  "projectName": "Employee_Rostering",
  "projectGroupId": "employeerostering",
  "projectVersion": "1.0.0-SNAPSHOT",
  "description": "Employee rostering problem optimisation using Planner. Assigns employees to shifts based on their skill."
}
```

[DELETE] /spaces/{spaceName}/projects/{projectName}

Deletes a specified project from a specified space.

Table 2.7. Request parameters

Name	Description	Type	Requirement
spaceName	Name of the space where the project is located	String	Required
projectName	Name of the project to be deleted	String	Required

Example server response (JSON)

```
{
  "jobId": "1541128617727-10",
  "status": "APPROVED",
  "projectName": "Employee_Rostering",
  "spaceName": "MySpace"
}
```

[POST] /spaces/{spaceName}/git/clone

Clones a project into a specified space from a specified Git address.

Table 2.8. Request parameters

Name	Description	Type	Requirement
spaceName	Name of the space to which you are cloning a project	String	Required
body	The **name**, **description**, and Git repository **userName**, **password**, and **gitURL** for the project to be cloned	Request body	Required

Example request body (JSON)

```
{
  "name": "Employee_Rostering",
  "description": "Employee rostering problem optimisation using Planner. Assigns employees to shifts based on their skill.",
  "userName": "baAdmin",
  "password": "password@1",
  "gitURL": "git://localhost:9418/MySpace/example-Employee_Rostering"
}
```

Example server response (JSON)

```
{
  "jobId": "1541129488547-13",
  "status": "APPROVED",
  "cloneProjectRequest": {
    "name": "Employee_Rostering",
    "description": "Employee rostering problem optimisation using Planner. Assigns employees to shifts based on their skill.",
    "userName": "baAdmin",
    "password": "password@1",
    "gitURL": "git://localhost:9418/MySpace/example-Employee_Rostering"
  },
  "spaceName": "MySpace2"
}
```

[POST] /spaces/{spaceName}/projects/{projectName}/maven/compile

Compiles a specified project in a specified space (equivalent to **mvn compile**).

Table 2.9. Request parameters

Name	Description	Type	Requirement
spaceName	Name of the space where the project is located	String	Required
projectName	Name of the project to be compiled	String	Required

Example server response (JSON)

```
{
  "jobId": "1541128617727-10",
  "status": "APPROVED",
  "projectName": "Employee_Rostering",
  "spaceName": "MySpace"
}
```

[POST] /spaces/{spaceName}/projects/{projectName}/maven/test

Tests a specified project in a specified space (equivalent to **mvn test**).

Table 2.10. Request parameters

Name	Description	Type	Requirement
spaceName	Name of the space where the project is located	String	Required
projectName	Name of the project to be tested	String	Required

Example server response (JSON)

```
{
  "jobId": "1541132591595-19",
  "status": "APPROVED",
  "projectName": "Employee_Rostering",
  "spaceName": "MySpace"
}
```

[POST] /spaces/{spaceName}/projects/{projectName}/maven/install

Installs a specified project in a specified space (equivalent to **mvn install**).

Table 2.11. Request parameters

Name	Description	Type	Requirement

Name	Description	Type	Requirement
spaceName	Name of the space where the project is located	String	Required
projectName	Name of the project to be installed	String	Required

Example server response (JSON)

```
{
  "jobId": "1541132668987-20",
  "status": "APPROVED",
  "projectName": "Employee_Rostering",
  "spaceName": "MySpace"
}
```

[POST] /spaces/{spaceName}/projects/{projectName}/maven/deploy

Deploys a specified project in a specified space (equivalent to **mvn deploy**).

Table 2.12. Request parameters

Name	Description	Type	Requirement
spaceName	Name of the space where the project is located	String	Required
projectName	Name of the project to be deployed	String	Required

Example server response (JSON)

```
{
  "jobId": "1541132816435-21",
  "status": "APPROVED",
  "projectName": "Employee_Rostering",
  "spaceName": "MySpace"
}
```

2.2.1.3. Jobs (API requests)

All **POST** and **DELETE** requests in the Knowledge Store REST API return a job ID associated with each request, in addition to the returned request details. You can use a job ID to view the request status or delete a sent request.

Knowledge Store REST API requests, or *jobs*, can have the following statuses:

Table 2.13. Job statuses (API request statuses)

Status	Description
ACCEPTED	The request was accepted and is being processed.
BAD_REQUEST	The request contained incorrect content and was not accepted.
RESOURCE_NOT_EXIST	The requested resource (path) does not exist.
DUPLICATE_RESOURCE	The resource already exists.
SERVER_ERROR	An error occurred in Decision Server.
SUCCESS	The request finished successfully.
FAIL	The request failed.
APPROVED	The request was approved.
DENIED	The request was denied.
GONE	The job ID for the request could not be found due to one of the following reasons: ● The request was explicitly removed. ● The request finished and has been deleted from a status cache. A request is removed from a status cache after the cache has reached its maximum capacity. ● The request never existed.

The Knowledge Store REST API supports the following endpoints for retrieving or deleting sent API requests. The Knowledge Store REST API base URL is **http://SERVER:PORT/decision-central/rest/**. All requests require HTTP Basic authentication or token-based authentication for the **rest-all** user role.

[GET] /jobs/{jobId}

Returns the status of a specified job (a previously sent API request).

Table 2.14. Request parameters

Name	Description	Type	Requirement
jobId	ID of the job to be retrieved (example: **1541010216919-1**)	String	Required

Example server response (JSON)

```
{
  "status": "SUCCESS",
  "jobId": "1541010216919-1",
  "result": null,
  "lastModified": 1541010218352,
  "detailedResult": [
    "level:INFO, path:null, text:Build of module 'Mortgage_Process' (requested by system)
completed.\n Build: SUCCESSFUL"
  ]
}
```

[DELETE] /jobs/{jobId}

Deletes a specified job (a previously sent API request). If the job is not being processed yet, this request removes the job from the job queue. This request does not cancel or stop an ongoing job.

Table 2.15. Request parameters

Name	Description	Type	Requirement
jobId	ID of the job to be deleted (example: **1541010216919-1**)	String	Required

Example server response (JSON)

```
{
  "status": "GONE",
  "jobId": "1541010216919-1",
  "result": null,
  "lastModified": 1541132054916,
  "detailedResult": [
    "level:INFO, path:null, text:Build of module 'Mortgage_Process' (requested by system)
completed.\n Build: SUCCESSFUL"
  ]
}
```

2.3. VERIFY THE MIGRATION IN DECISION CENTRAL

After you have migrated all Decision Central data, verify that the migration was successful in the new Decision Central for Red Hat Decision Manager 7.1.

Prerequisite

Project data from Red Hat JBoss BRMS 6.4 has been migrated using the Decision Central migration tool.

Procedure

1. Start Red Hat Decision Manager 7.1 depending on your installation.
   Example: Installation with Red Hat JBoss EAP

```
$ ~/$EAP_HOME/bin/standalone.sh -c standalone-full.xml
```

2. Log in to Decision Central with your credentials and navigate to **Menu → Design → Projects**.

3. Verify that all projects have been migrated and select each project to verify the migrated project assets.

4. For each project, click **Build** and then **Deploy** in the upper-right corner of the project window to validate the migrated project data and the configured Decision Server.

2.3.1. Troubleshooting Decision Central migration problems

If you encounter problems with your Decision Central migration to Red Hat Decision Manager 7.1, review the following troubleshooting suggestions:

- If any project data is missing from Decision Central, ensure that the **.niogit** directory for Red Hat Decision Manager 7.1 contains the restructured data and is in the correct installation location for Red Hat Decision Manager 7.1 (for example, **~/$EAP_HOME/bin/**).

- If projects fail to build, open the project **pom.xml** file and remove the **http://<SERVER>:<PORT>/decision-central/maven2/** repository, if present. The use of this repository can prevent projects from building properly in Red Hat Decision Manager 7.1. If this repository contains project data that you want to preserve, create a new repository for the data and add the repository as a dependency in the **pom.xml** file.
  Example repository to be removed from **pom.xml** file:

  ```
  <repository>
    <id>guvnor-m2-repo</id>
    <name>Guvnor M2 Repo</name>
    <url>http://localhost:8080/decision-central/maven2/</url>
  </repository>
  ```

- If projects fail to deploy to Decision Server, review your Decision Server installation and configuration. For more information, see *Planning a Red Hat Decision Manager installation* .

- If you cannot resolve migration problems, complete the migration process again on a new Red Hat Decision Manager 7.1 installation.

CHAPTER 3. JAVA PROJECT MIGRATION

Projects that you developed in Java code in Red Hat JBoss BRMS 6.4 (for example, in Eclipse) also require modification for Red Hat Decision Manager 7.1. You must update the dependencies in the **pom.xml** file for each project and rebuild the project. To migrate your Java client applications, you also must update the dependencies in the **pom.xml** file for each project. If your application uses an embedded Red Hat JBoss BRMS 6.4 engine (Drools), this change also updates the engines. If the application calls the Decision Server, the API client library is updated.

3.1. UPDATE JAVA PROJECT DEPENDENCIES

You must update several dependencies in your Java projects to prepare them for migration to Red Hat Decision Manager 7.1.

Prerequisite

The Maven repository for Red Hat Decision Manager 7.1 has been downloaded from the Red Hat Customer Portal and made available to the local Maven installation (as a local or remote repository).

Procedure

1. Open the **pom.xml** file of the project and remove the **<version>** tag from any dependencies under the following groups:
    - **org.kie**
    - **org.drools**
    - **org.jbpm**
    - **org.optaplanner**
2. Add the Red Hat Business Automation bill of materials (BOM) dependency under the **<dependencies>** tag within the **<dependencyManagement>** section:

    ```
    <dependency>
      <groupId>com.redhat.ba</groupId>
      <artifactId>ba-platform-bom</artifactId>
      <version>7.1.0.GA-redhat-00002</version>
      <scope>import</scope>
      <type>pom</type>
    </dependency>
    ```

    The Red Hat Business Automation BOM applies to both Red Hat Decision Manager and Red Hat Process Automation Manager. When you add the BOM files, the correct versions of transitive dependencies from the provided Maven repositories are included in the project. The BOM **<version>** is the version of the **com.redhat.ba:ba-platform-bom** Red Hat Business Automation artifact in the Maven repository. You can view the version of the artifact by entering the repository and navigating to the **maven-repository/com/redhat/bom/ba/ba-platform-bom** directory.
3. If your code uses any Drools CDI annotations (**@KReleaseId** , **@KContainer**, **@KBase**, **@KSession**), also add the following dependency:

    ```
    <dependency>
      <groupId>org.drools</groupId>
    ```

```
  <artifactId>drools-cdi</artifactId>
</dependency>
```

This dependency is necessary because the CDI extension for processing the annotations is now defined using a separate module. Without this dependency, the annotations do not work.

4. Replace the legacy **org.guvnor** dependency (if applicable) with the new **org.uberfire** dependency.
   Legacy **org.guvnor** dependency:

```
<dependency>
  <groupId>org.guvnor</groupId>
  <artifactId>guvnor-rest-client</artifactId>
</dependency>
```

   New **org.uberfire** dependency:

```
<dependency>
  <groupId>org.uberfire</groupId>
  <artifactId>uberfire-rest-client</artifactId>
</dependency>
```

5. Update the version of the project artifact and save the **pom.xml** file.

3.2. VERIFY THE JAVA PROJECT MIGRATION

After you have migrated all Java project data, rebuild the Java project to verify a successful migration.

Prerequisites

- Java project dependencies have been updated.
- Maven 3.x or later is installed.
- Java 8 or later is installed.

Procedure

1. Rebuild each Java project using a regular Maven build.

```
$ mvn clean install
```

2. Review all project data in each build to confirm successful migration.
   If you encounter build errors, verify that the project dependencies have been updated correctly. To review other product changes that may have impacted your Java project, see Chapter 4, *Other migration considerations*.

CHAPTER 4. OTHER MIGRATION CONSIDERATIONS

Red Hat Decision Manager 7.1 contains changes to APIs, rule logic, and Red Hat Business Optimizer features that you should note during your migration. These changes are generally backward compatible between product versions, but in some cases, migration effort is required to resolve build or migration errors.

As part of your migration to Red Hat Decision Manager 7.1, review these changes and address any inconsistencies or errors in your projects that arise.

4.1. RED HAT BUSINESS OPTIMIZER CHANGES IN RED HAT DECISION MANAGER 7.1

Red Hat Business Optimizer is an embeddable planning engine in Red Hat Decision Manager that optimizes planning problems. Red Hat Business Optimizer is based on the community OptaPlanner project that is regularly updated and in some cases requires code changes for the latest Red Hat Business Optimizer features. For an overview of the latest OptaPlanner changes and migration requirements, see the OptaPlanner upgrade recipe archive. OptaPlanner upgrade information for versions 7.0 through 7.11, inclusive, is relevant for upgrading from Red Hat JBoss BRMS 6.4 to Red Hat Decision Manager 7.1.

In Red Hat Decision Manager 7.1, certain Red Hat Business Optimizer configurations in Decision Central must be updated to accommodate recent OptaPlanner changes.

4.1.1. Updating Red Hat Business Optimizer asset configurations in Decision Central

If you have any solver configuration assets (**.solver.xml** files) or solution-related data objects in Decision Central, you must make certain updates to these assets in Red Hat Decision Manager 7.1 to accommodate recent Red Hat Business Optimizer changes.

Prerequisite

Decision Central data has been migrated from Red Hat JBoss BRMS 6.4 to Red Hat Decision Manager 7.1. For migration instructions, see Chapter 2, *Decision Central data migration* .

Procedure

1. Log in to Decision Central for Red Hat Decision Manager 7.1.
2. In Decision Central, go to **Menu** → **Design** → **Projects** and select the project name.
3. Open any solver configuration asset (**.solver.xml** file), if present.
4. In the solver configuration designer, click **Save** without making any changes. This step is required to generate the new code for solver configuration assets in Red Hat Decision Manager 7.1. Do this step with any other solver configuration assets.
5. Under **Data Objects** in the **Project Explorer** (left menu), open any data object (**.java** file) configured as a **Planning Solution**, if applicable.
   To verify if this setting is selected for this data object, click the OptaPlanner icon on the right side of the data objects designer. If **Planning Solution** is not selected, then this procedure does not apply.

6. Under **general properties** in the data objects designer, set the **Superclass** drop-down option to **Nothing selected**. This setting is no longer required by Red Hat Business Optimizer in Red Hat Decision Manager 7.1.

7. On the right side of the data objects designer, click the OptaPlanner icon to reveal the **Planner Settings**. The **Planning Solution** option should be selected (if not, then this procedure does not apply).

8. Select **No selected**, then re-select **Planning Solution** and specify the **Solution Score Type**. This step is required to generate the new code for planning solutions in Red Hat Decision Manager 7.1.

9. Click **Save** in the data objects designer and make the same changes to any other data objects configured as a **Planning Solution**.

4.2. API CHANGES IN RED HAT DECISION MANAGER 7.1

Many of the API changes from Red Hat JBoss BRMS 6.4 to Red Hat Decision Manager 7.1 are backward compatible and do not affect migration. However, the following table lists class changes in APIs that may not be compatible in all cases.

If you encounter build or migration errors related to these changes, review the full List of API changes in the Red Hat Customer Portal and update your code as needed.

Table 4.1. API changes in Red Hat Decision Manager 7.1 (abbreviated)

API	Description	Changed classes
Drools core (**drools-core**)	The decision engine	• **org.drools.core.command.*** • **org.drools.core.common.***
KIE API (**kie-api**)	The main API for all projects from KIE Group	• **org.kie.api.task.*** • **org.kie.api.executor.*** • **org.kie.api.concurrent.*** • **org.kie.api.builder.*** • **org.kie.api.command.*** • **org.kie.api.runtime.***

API	Description	Changed classes
KIE server API (**kie-server-api**)	The general Decision Server API	• **org.kie.server.api.commands.*** • **org.kie.server.api.marshalling.*** • **org.kie.server.api.model.*** • **org.kie.server.api.rest.RestURI** (constants have slightly changed, omitting leading /)
KIE server client API (**kie-server-client**)	The API for Decision Server client	• **org.kie.server.client.SolverServicesClient** • **org.kie.server.client.UIServicesClient** • **org.kie.server.client.admin.ProcessAdminServicesClient** • **org.kie.server.client.ProcessServicesClient** • **org.kie.server.client.QueryServicesClient** • **org.kie.server.client.JobServicesClient** • **org.kie.server.client.UserTaskServicesClient** • **org.kie.server.client.KieServicesClient** • **org.kie.server.client.KieServicesConfiguration** • **org.kie.server.client.CaseServicesClient**
KIE server controller API (**kie-server-controller-api**)	The API for the Decision Manager controller	• **org.kie.server.controller.api.service.***
KIE server controller REST API (**kie-server-controller-rest**)	The REST API for Decision Manager controller	• **org.kie.server.controller.rest.RestSpecManagementServiceImpl** • **org.kie.server.controller.rest.RestKieServerControllerImpl**

4.3. LOGIC CHANGES IN RED HAT DECISION MANAGER 7.1

Note the following logic changes in Red Hat Decision Manager 7.1:

- In Red Hat JBoss BRMS 6.4, when a rule executes the sum function in an accumulate pattern, the result always returns as a double data type regardless of the data type of the inputs. Red Hat Decision Manager 7.1 preserves the data type of the inputs on which the sum is executed. This enhancement provides a more accurate result from the sum function. In the following example, the result type of the accumulate function is **Long** instead of **double**:

  ```
  Long(...) from accumulate(..., sum($p.getLongWeight()))
  ```

  If the rules in your Red Hat JBoss BRMS 6.4 project include the sum function in an accumulate pattern, search for these functions and review them. These functions return a double data type in Red Hat JBoss BRMS 6.4, but will return the data type of the input values in Red Hat Decision Manager 7.1.

- When no fact matches the accumulate pattern, **min** and **max** accumulate functions in Red Hat Decision Manager 7.1 do not return **+/-Integer.MAX_VALUE**, but return **null**. Therefore, unlike in Red Hat JBoss BRMS 6.4, the accumulate in the rule is not matched and the rule does not fire in Red Hat Decision Manager 7.1.

- Business processes that contain business rule tasks with an **implementation=Java** configuration will not be compiled in Red Hat Decision Manager 7.1 due to stricter validation requirements. To resolve compilation errors related to this restriction, set the implementation configuration to **implementation=##unspecified** or remove the **implementation** attribute.

CHAPTER 5. NEXT STEPS

- *Getting started with decision services*
- *Designing a decision service using guided decision tables*

APPENDIX A. VERSIONING INFORMATION

Documentation last updated on Wednesday, March 27, 2019.