Red Hat JBoss BPM Suite 6.1

Guide d'administration et de configuration

Le guide d'administration et de configuration de Red Hat JBoss BPM Suite

Last Updated: 2017-11-21

Red Hat JBoss BPM Suite 6.1 Guide d'administration et de configuration

Le guide d'administration et de configuration de Red Hat JBoss BPM Suite

Kanchan Desai
kadesai@redhat.com

Doug Hoffman

Eva Kopalova

Red Hat Content Services

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

Joshua Wulf
jwulf@redhat.com

Notice légale

Résumé

Guide pour les administrateurs et les utilisateurs avancés qui s'occupent des installations, configurations et utilisations avancées de Red Hat JBoss BPM Suite.

Table des matières

PARTIE I. INTRODUCTION **5**

CHAPITRE 1. BPMN (DE L'ANGLAIS BUSINESS PROCESS MODEL AND NOTATION) **6**
1.1. COMPOSANTS 6
1.2. PROJET 7
1.3. CRÉATION D'UN PROJET 7
1.4. AJOUTER DES DÉPENDANCES 8

PARTIE II. CONFIGURATION **10**

CHAPITRE 2. CONFIGURATION DE BUSINESS CENTRAL **11**
2.1. CONRÔLE DE L'ACCÈS 11
2.2. CONFIGURATION D'UN PROFIL DE BUSINESS CENTRAL 12
2.3. PERSONNALISATION DE L'APPLICATION BUSINESS CENTRAL 13
2.4. DESCRIPTEURS DE DÉPLOIEMENT 16
2.5. GESTION DE LA POLITIQUE DE REMPLACEMENT DE DÉPLOIEMENT 20
2.6. ÉTENDRE BUSINESS CENTRAL 21
2.7. CONFIGURER DES COLONNES SUR UN TABLEAU 28

CHAPITRE 3. CONFIGURATION EN LIGNE DE COMMANDE **30**
3.1. DÉMARRER L'OUTIL KIE-CONFIG-CLI TOOL EN MODE EN LIGNE 30
3.2. DÉMARRER L'OUTIL KIE-CONFIG-CLI TOOL EN MODE HORS LIGNE 30
3.3. COMMANDES DISPONIBLES POUR L'OUTIL KIE-CONFIG-CLI 31

CHAPITRE 4. MIGRATION **33**
4.1. MIGRATION DES DONNÉES 33
4.2. MIGRATION DE RUNTIME 34
4.3. API ET COMPATIBILITÉ RÉTROACTIVE 35
4.4. MIGRATION DU SERVICE DE TÂCHES 35

CHAPITRE 5. GESTION DES DONNÉES **37**
5.1. SAUVEGARDE DES DONNÉES 37
5.2. INSTALLATION DES INDEXES 37
5.3. INSTALLER LA BASE DE DONNÉES 37
5.4. MODIFIER LA BASE DE DONNÉES 38
5.5. SCRIPTS DDL 39

CHAPITRE 6. RÉFÉRENTIEL DE RESSOURCES **40**
6.1. CRÉATION D'UN RÉFÉRENTIEL 40
6.2. CLONAGE D'UN RÉFÉRENTIEL 41
6.3. SUPPRIMER UN RÉFÉRENTIEL 43
6.4. GESTION DES RESSOURCES 44
6.5. RÉFÉRENTIEL MAVEN 50
6.6. CONFIGURER UN DÉPLOIEMENT POUR UN RÉFÉRENTIEL NEXUS DISTANT 51
6.7. CONFIGURATION SYSTÈME 52

CHAPITRE 7. EXPORTATION ET IMPORTATION D'UN PROCESSUS **54**
7.1. CRÉER UNE DÉFINITION DE PROCESSUS 54
7.2. IMPORTATION D'UNE DÉFINITION DE PROCESSUS 54
7.3. IMPORTER JPDL 3.2 DANS BPMN2 55
7.4. EXPORTATION D'UN PROCESSUS 56

PARTIE III. INTÉGRATION **57**

CHAPITRE 8. LE DÉPLOIEMENT D'ARTEFACTS DE RED HAT JBOSS BPM SUITE DANS S-RAMP (DE L'ANGLAIS REPOSITORY ARTIFACT MODEL AND PROTOCOL) 58
8.1. LE DÉPLOIEMENT D'ARTEFACTS DE RED HAT JBOSS BPM SUITE DANS SOA REPOSITORY ARTIFACT MODEL AND PROTOCOL (S-RAMP) EN UTILISANT MAVEN 58
8.2. LE DÉPLOIEMENT D'ARTEFACTS DE RED HAT JBOSS BPM SUITE DANS SOA REPOSITORY ARTIFACT MODEL AND PROTOCOL (S-RAMP) EN UTILISANT L'INTERFACE GRAPHIQUE (GUI). 60

CHAPITRE 9. INTÉGRER RED HAT JBOSS BPM SUITE DANS RED HAT JBOSS FUSE 61
9.1. INSTALLER / METTRE À JOUR LES FONCTIONNALITÉS D'INTÉGRATION PRINCIPALES 63
9.2. INSTALLER DES FONCTIONNALITÉS D'INTÉGRATION SUPPLÉMENTAIRES 64
9.3. INSTALLER LES APPLICATIONS QUICKSTART DEJBOSS FUSE INTEGRATION 65

CHAPITRE 10. INTÉGRATION AVEC SPRING 68
10.1. CONFIGURER RED HAT JBOSS BPM SUITE AVEC SPRING 68

CHAPITRE 11. INTÉGRATION CDI 70
11.1. INTÉGRATION CDI 70

CHAPITRE 12. PERSISTANCE 72
12.1. SESSION 72
12.2. INSTANCE DE PROCESSUS 73
12.3. ÉLÉMENT DE TRAVAIL 74
12.4. CONFIGURATION DE LA PERSISTANCE 75

CHAPITRE 13. TRANSACTIONS 79
13.1. TRANSACTIONS 79
13.2. DÉFINIR DES TRANSACTIONS 79
13.3. CMT (CONTAINER MANAGED TRANSACTIONS) 80

CHAPITRE 14. JOURNALISATION 82
14.1. JOURNALISATION DES ÉVÉNEMENTS DANS LA BASE DE DONNÉES 84
14.2. FONCTIONNALITÉ LOGBACK 85
14.3. CONFIGURER LA JOURNALISATION 86

CHAPITRE 15. LOCALISATION ET PERSONNALISATION 88
15.1. LANGUES DISPONIBLES 88
15.2. MODIFIER LES PARAMÈTRES DE LANGUE 88
15.3. EXÉCUTER UNE JVM AVEC LA CODIFICATION UTF-8 89

PARTIE IV. EXÉCUTION 90

CHAPITRE 16. SERVEUR D'EXÉCUTION 91
16.1. RÈGLES D'ATTRIBUTION 91
16.2. SESSION MAIL 92

CHAPITRE 17. PLUG-IN DE RED HAT JBOSS DEVELOPER STUDIO 94
17.1. PLUG-IN 94

PARTIE V. MONITORING 95

CHAPITRE 18. MONITORING DES PROCESSUS 96
18.1. RÉSEAU JBOSS OPERATIONS NETWORK 96
18.2. INSTALLER LE PLUG-IN JBOSS BRMS DANS JBOSS ON 96
18.3. CONTRÔLE DES KIEBASES ET DES KIESESSIONS 97

CHAPITRE 19. GESTION DE LA SÉCURITÉ DANS RED HAT JBOSS BPM SUITE DASHBUILDER 98
19.1. ACCÉDER À RED HAT JBOSS BPM SUITE DASHBUILDER 98

19.2. GESTION DE LA SÉCURITÉ 98
19.3. PERMISSIONS D'ESPACE DE TRAVAIL 98
19.4. PERMISSIONS DE PAGE 100
19.5. PANNEAU DE PERMISSIONS 101

ANNEXE A. HISTORIQUE DES VERSIONS 103

PARTIE I. INTRODUCTION

CHAPITRE 1. BPMN (DE L'ANGLAIS BUSINESS PROCESS MODEL AND NOTATION)

BPMN (Business Process Model and Notation) est un système d'annotations standards de modélisation des processus métier. BPMS aspire à réduire l'écart entre les analystes et les programmeurs en procurant un langage de flux de travail qui puisse être clairement compris par les deux.

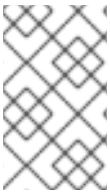
1.1. COMPOSANTS

Red Hat JBoss BPM Suite intègre plusieurs composantes pour soutenir les processus métier tout au long de leur cycle de vie et pour fournir des fonctionnalités de gestion des processus et des outils pour les analystes, les développeurs et les utilisateurs professionnels. Le produit peut être déployé sur plusieurs serveurs JEE compatibles ; l'option recommandée est Red Hat JBoss Enterprise Application Platform 6.

Red Hat JBoss BPM Suite contient les composantes principales suivantes :

- **`Execution Engine`** - fournit l'environnement d'exécution des processus et des règles d'entreprise. Il comprend une bibliothèque de flux de travail qui peut être incorporée dans une application web de l'utilisateur. Le Runtime Manager est l'objet root et il contient les composantes suivants :
  - **`Runtime Engine`** - implémente le comportement principal du langage informatique de la machine et il est fourni par le Runtime Manager.
    - **`Process Engine`** - est l'environnement d'exécution du modèle de processus d'entreprise.
    - **`Task Service`** - gère les cycles de vie des tâches humaines.
  - **`Rule Engine`** - peut être utilisé avec le moteur de processus ou tout seul.
    - **`Rules Evaluation`** - exécute les règles métier sur la base d'un ensemble de faits donnés.
    - **`Complex Event Processing`** - applique les règles métier sur des flux d'événements entrants.
- **`Business Central`** - une application basée-web qui accommode des outils de création de ressources, de gestion et de contrôle en procurant un environnement web intégré.
  - **`Référentiel Ressources`** - c'est l'emplacement central de partage des connaissances (Knowledge Store) des ressources d'entreprise, processus, règles, formulaires, etc.. Sur la page Project Explorer de Business Central via **Création** → **Création Projet**, les utilisateurs peuvent accéder à ce référentiel. Par défaut, le produit initialise un référentiel GIT local comme son référentiel de ressources. Toutefois, d'autres référentiels peuvent être ajoutés ou retirés si nécessaires.
  - **`Référentiel d'artefacts`** - c'est un référentiel de stockage d'artefacts de jars de projets basé Maven
  - **`Serveur d'exécution`** - fournit un environnement d'exécution pour les tâches et les instances de processus d'entreprise.

- **`Surveillance Activité`** - fournit un affichage personnalisable sur la performance de l'entreprise.



NOTE

Red Hat JBoss BRMS est fourni avec sa propre application Business Central qui représente un sous-ensemble de l'application Business Central de Red Hat JBoss BPM Suite.

1.2. PROJET

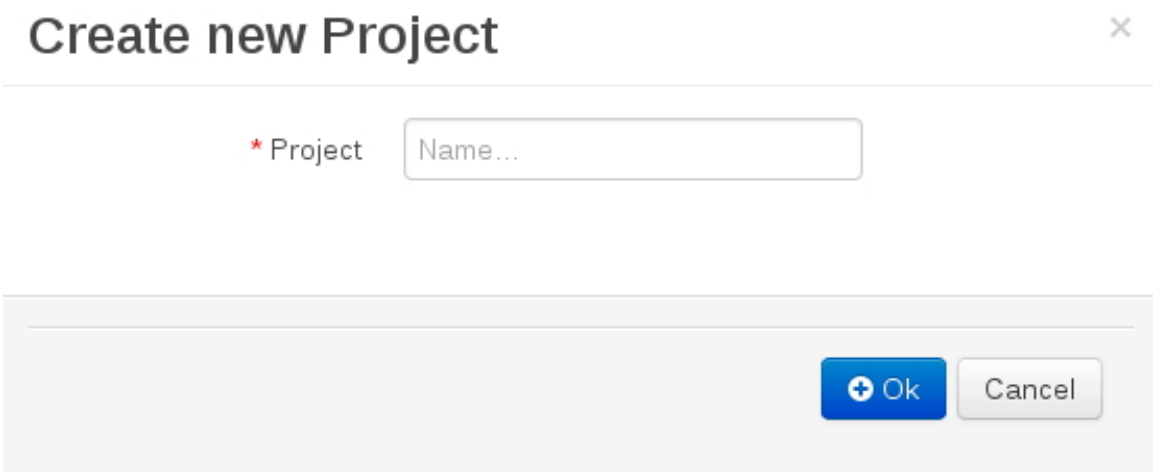
Un projet est un conteneur pour les paquets de ressources (processus d'entreprise, règles, définitions de travaux, tables de décision, modèles de faits, modèles de données et des DSL) qui se trouvent dans le référentiel de connaissances. C'est ce conteneur qui définit les propriétés de KIE Base et de KIE Session, qui sont appliquées à son contenu. Dans l'interface graphique, vous pouvez modifier ces entités dans l'éditeur de projet.

Un projet étant un projet Maven, il contient le fichier de Project Object Model (**`pom.xml`**) avec des informations sur la façon de construire l'artefact sortant. Il contient également le fichier de descripteur de module, **`kmodule.xml`**, qui contient la configuration de KIE Base et de KIE Session pour les ressources du projet.

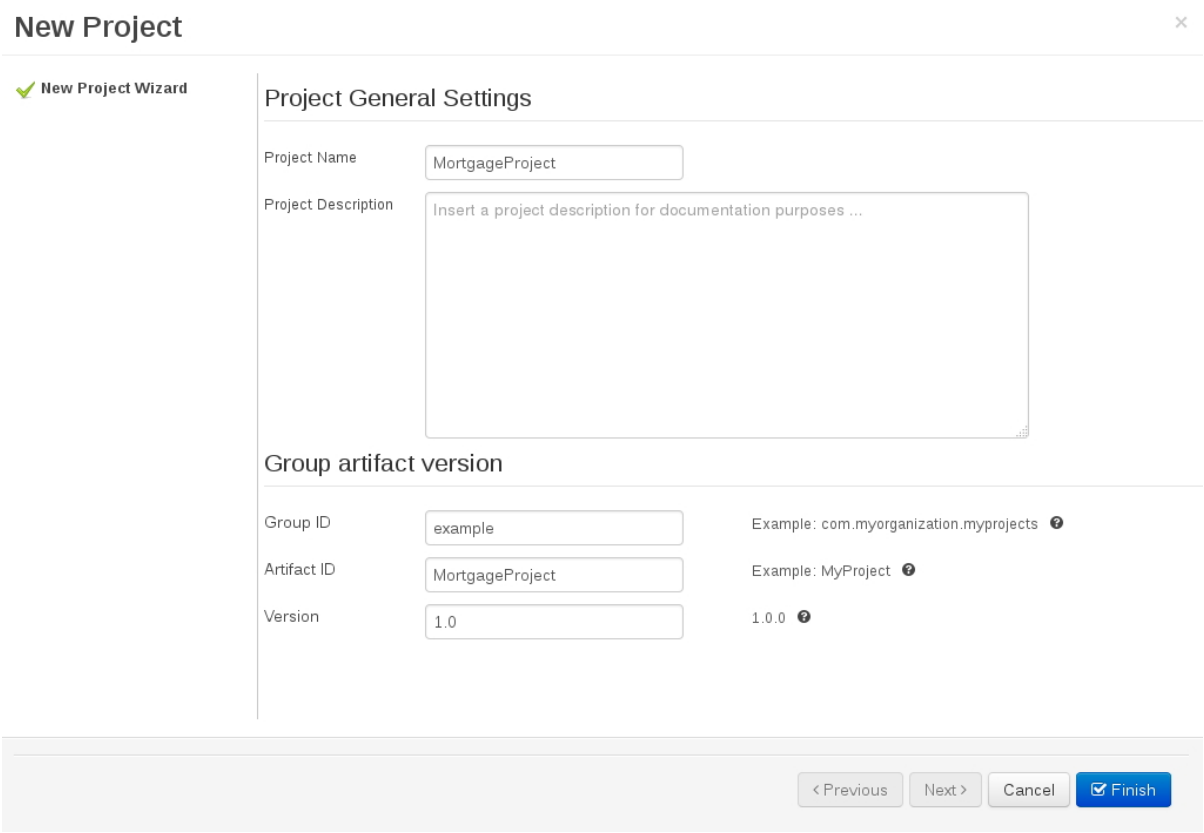
1.3. CRÉATION D'UN PROJET

Pour créer un projet, procéder ainsi :

1. Ouvrir la perspective **`Création Projet`** : à partir du menu principal, cliquer sur **Création** → **Création Projet**.
2. Dans **`Project Explorer`**, sélectionner l'unité organisationnelle et le référentiel où vous souhaitez créer le projet.
3. Dans le menu perspective, cliquer sur **Nouvel élément** → **Projet**.
4. Dans la boîte de dialogue **`Créer Nouveau projet`**, définir les informations sur le projet :
   a. Dans la case **`Projet`**, saisir le nom du projet.



5. Le Project Explorer se réactualise pour afficher une fenêtre **`Assistant Nouveau projet`**.



6. Définir les informations **`Configuration générale Projet`** et **`Version Artefact Groupe`** de ce nouveau projet. Ces paramètres se trouvent dans le fichier de configuration **`pom.xml`** de Maven.

   - **`Nom du projet`** : le nom du projet; par exemple **`MortgageProject`**
   - **`Description du projet`** : la description du projet qui peut être utile à des buts de documentation du projet.
   - **`ID Groupe`** : ID de groupe du projet ; par exemple **`org.mycompany.commons`**
   - **`ID Artefact`** : ID d'artefact unique au groupe ; par exemple **`myframework`**. Évitez d'utiliser un espace ou un caractère spécial qui risque de vous mener à un nom non valide.
   - **`ID Version`** : version du projet ; par exemple **`2.1.1`**

   L'affichage **`Écran Projet`** est mis à jour avec les nouveaux détails du projet, tel que défini dans le fichier **`pom.xml`**. Notez que vous pouvez basculer entre les fichiers de descripteur du projet dans le menu déroulant via **`Configuration Projet`** et **`Configuration Base de connaissances`** et modifier leur contenu.

1.4. AJOUTER DES DÉPENDANCES

Pour ajouter des dépendances à votre projet, procéder ainsi :

1. Ouvrir le Project Editor pour le projet donné :

   a. Sur la page **`Project Explorer`** de la perspective **`Création Projet`**, ouvrir le répertoire du projet.

b. Cliquer sur le bouton 

pour ouvrir la vue du projet.

2. Dans la vue **`Écran du projet`**, sélectionner dans la boîte de dialogue **`Configuration Projet`** l'élément **`Dépendances`**.

3. Sur l'écran **`Écran du projet`** mis à jour, cliquer sur le bouton **`Ajouter`** pour ajouter une dépendance Maven ou cliquer sur le bouton **`Ajouter à partir du référentiel`** pour ajouter une dépendance du Knowledge Store (référentiel d'artefacts) :

   - Quand il ajoute une dépendance Maven, un utilisateur doit définir les **`ID Groupe`**, **`ID Artefact`** et **`ID Version`** sur la nouvelle rangée créée dans la table de dépendances.

   - Quand on ajoute une dépendance du Knowledge Store, il faut sélectionner la dépendance dans la boîte de dialogue qui s'affiche : la dépendance sera ajoutée au tableau des dépendances.

4. Pour pouvoir appliquer les différents changements, les dépendances doivent être sauvegardées.

AVERTISSEMENT

Si vous travaillez avec des artéfacts modifiés, ne chargez pas à nouveau des artefacts non liés à des instantanés modifiés car Maven ne reconnaîtra pas ces artefacts mis à jour, et il ne fonctionnera pas si c'est déployé ainsi.

PARTIE II. CONFIGURATION

CHAPITRE 2. CONFIGURATION DE BUSINESS CENTRAL

Business Central est une application web, tous les paramètres de configuration sont chargés à partir du ***DEPLOY_DIRECTORY*/business-central.war/WEB-INF/web.xml** et des fichiers référencés, et si déployé sur Red Hat JBoss EAP 6, également dans **jboss-web.xml** et **jboss-deployment-structure.xml**.

Notez que l'application totale peut être exécutée dans des profis différents (voir *Red Hat JBoss BPM Suite Installation Guide*).

2.1. CONRÔLE DE L'ACCÈS

Le mécanisme de contrôle de l'accès inclut l'autorisation et l'authentification. Dans l'environnement unifié de Red Hat JBoss BPM Suite , les utilisateurs peuvent mettre à jour les rôles d'utilisateur situés dans ***$JBOSS_HOME*/standalone/deployments/business-central.war/WEB-INF/classes/userinfo.properties**.

Pour donner accès à JBoss BPM Suite à un utilisateur, l'utilisateur devra avoir le rôle suivant assigné :

- **admin** : administre le système JBoss BPM Suite et possède des droits pour effectuer les changements nécessaires, y compris la possibilité d'ajouter ou de supprimer des utilisateurs du système.
- **developer** : implémente le code requis pour que les processus fonctionnent et a accès à tout sauf aux tâches administratives.
- **analyst** : crée et conceptualise des processus et des formulaires, instancie les artefacts de processus et de déploiement. Ce rôle est similaire à celui d'un développeur, sans l'accès au référentiel des ressources, ni aux déploiements.
- **user** : réclame, effectue et fait appel à d'autres actions (par exemple, escalade, rejet, etc.) sur les tâches affectées et n'a pas accès aux fonctions de création (authoring).
- **manager** : contrôle le système et ses statistiques, et a uniquement accès au tableau de bord.
- **business user** : effectue les tâches d'entreprise requises pour que les processus puissent continuer. Travaille surtout sur la liste de tâches.

Si vous utilisez Red Hat JBoss EAP, pour créer un utilisateur avec des rôles particuliers, exécutez le script ***$JBOSS_HOME*/add-user.sh** et créez un utilisateur d'application dans **ApplicationRealm** avec les rôles respectifs.

Configuration de workbench

Dans Red Hat JBoss Suite BPM, les utilisateurs peuvent créer des rôles à l'aide de LDAP pour modifier les rôles existants. Les utilisateurs peuvent modifier les rôles dans la configuration workbench pour s'assurer que les rôles uniques basés LDAP soient conformes aux standards de l'entreprise en modifiant le répertoire de déploiement situé dans ***$JBOSS_HOME*/standalone/deployments/business-central.war/WEB-INF/classes/workbench-policy.propeties**.

Si vous authentifiez un utilisateur via LDAP via GIT, les administrateurs doivent définir la propriété système org.uberfire.domain au nom du module de connexion qu'il doit utiliser pour authentifier les utilisateurs par le service GIT. Cela doit être défini dans le fichier **standalone.xml** dans EAP.

Authentification pour les tâches humaines

Chaque tâche qui doit être exécutée est assignée à un ou plusieurs rôles ou groupes, de façon à ce que chaque rôle donné ou que le groupe donné assigné puisse réclamer l'instance de la tâche et l'exécuter. Les tâches peuvent être assignées directement à un ou à plusieurs utilisateurs. JBoss BPM Suite utilise l'interface **`UserGroupCallback`** pour assigner des tâches aux utilisateurs.

AVERTISSEMENT

Un groupe de tâche humaine ne doit pas être nommé d'aprés un utilisateur existant dans le système. Cela causerait des problèmes intermittents.

2.2. CONFIGURATION D'UN PROFIL DE BUSINESS CENTRAL

Le serveur de Red Hat JBoss BPM Suite 6 (ou versions supérieures) est capable de démarrer l'application Business Central de trois façons différentes :

- Full profile - profil par défaut actif sans besoin de configuration supplémentaire (IU et services distants, par ex. REST).
- Execution server profile - désactive les composants IU de l'application complètement et permet un accès éloigné uniquement, par ex. via l'interface REST.
- UI server profile - désactive les services distants comme REST et permet uniquement l'accès à l'IU à l'application.

Pour modifier le profil, utiliser les étapes de configuration suivantes.

Procédure 2.1. Configuration des profils de Business Central

1. Sélectionner le **`web.xml`** désiré dans **`$BPMS_HOME/standalone/deployments/business-central.war/WEB-INF/`**. Les fichiers suivants sont fournis.
   - **`web.xml`** (valeur par défaut) pour le profil complet
   - **`web-exec-server.xml`** pour le profil de serveur d'exécution
   - **`web-ui-server.xml`** pour le profil de serveur de l'IU
2. Pour activer un profil autre que le profil complet (full profil) par défaut, le fichier web-<PROFILE>.xml doit être renommé **`web.xml`**. Les étapes suivantes montrent une façon d'exécuter le profil du serveur.
   a. Sauvegarder le fichier **`web.xml`** du profil complet (full profile).

   ```
   $ mv web.xml web-full.xml
   ```

   b. Renommer le fichier **`web-exec-server.xml`** :

   ```
   $ mv web-exec-server.xml web.xml
   ```

3. Démarrer le serveur d'applications avec une propriété système supplémentaire pour instruire le gestionnaire de profils d'activer le profil donné.
    - **`Dorg.kie.active.profile=full`** - pour activer le profil complet (full profil) ou éviter la propriété complètement.
    - **`Dorg.kie.active.profile=exec-server`** - pour activer le profil de serveur d'exécution
    - **`Dorg.kie.active.profile=ui-server`** - pour activer le serveur de l'IU

2.3. PERSONNALISATION DE L'APPLICATION BUSINESS CENTRAL

L'application web Business Central permet de personnaliser son apparence en vous permettant de substituer certains de ses styles qui existent par défaut. La possibilité de personnaliser Business Central vous permet d'obtenir une apparence cohérente à travers toutes vos applications, ce qui améliore votre expérience utilisateur. C'est également utile lorsque plusieurs équipes utilisent l'application. Chaque équipe peut développer sa propre interface utilisateur personnalisée. Les éléments personnalisables sont construits en utilisant des feuilles de style en cascade (CSS), des images et des fichiers HTML, fournissant ainsi une approche flexible et facile à personnaliser, sans besoin de recompiler le code.

Vous pouvez modifier les éléments suivants dans l'application Business Central pour rester en phase avec l'image de marque de votre entreprise :

- Écran de connexion

  Vous pouvez personnaliser les attributs de l'écran de connexion de Business Central suivants :

    - L'image d'arrière-plan
    - Le logo de la société
    - Le logo de l'application

- L'en-tête de l'application

  Vous pouvez personnaliser les attributs de l'en-tête de l'application de Business Central suivants :

    - L'en-tête de Business Central contient le titre et le logo de la bannière

- Fenêtres contextuelles d'assistance

  Vous pouvez personnaliser les attributs suivants des fenêtres contextuelles d'assistance de l'écran de démarrage :

    - Les images d'assistance de l'écran de démarrage
    - Texte de l'étiquette

2.3.1. Personnalisation de la page de connexion de Business Central

Procédure 2.2. Modifier l'image d'arrière-plan de la page de connexion de Business Central

1. Démarrer le serveur EAP et ouvrir http://localhost:8080/business-central dans un navigateur web.
2. Copier la nouvelle image d'arrière-plan dans le répertoire **`$EAP_HOME/standalone/deployments/business-central.war/images`** de votre installation JBoss BPM Suite.
3. Naviguez dans le répertoire **`$EAP_HOME/standalone/deployments/business-central.war/styles`** et ouvrir le fichier **`login-screen.css`** dans l'éditeur de texte.
4. Dans le fichier **`login-screen.css`**, indiquez l'emplacement de votre nouvelle image d'arrière-plan dans l'attribut ***`background-image`*** suivant.

   ```
   background-image: url("../images/login-screen-background.jpg");
   ```

   L'attribut ***`background-image`*** pointe vers l'image **`login-screen-background.jpg`** par défaut.

   En plus de l'image d'arrière-plan, vous pouvez modifier les autres attributs comme la taille de l'image, sa position, la couleur du fond dans le fichier **`login-screen.css`**.

Réactualiser la page de connexion de Business Central pour apercevoir les changements.

Procédure 2.3. Modifier les logos de la société et du projet dans Business Central

1. Démarrer le serveur EAP et ouvrir http://localhost:8080/business-central dans un navigateur web.
2. Naviguez dans le répertoire **`$EAP_HOME/standalone/deployments/business-central.war/images`** de votre installation JBoss BPM Suite.
3. Remplacer l'image par défaut **`login-screen-logo.png`** par une nouvelle image. Ce sera le logo de la société, qui apparaîtra dans le coin de droite de votre page de connexion.
4. Remplacer l'image par défaut **`RH_JBoss_BPMS_Logo.pngRH_JBoss_BRMS_Logo.png`** par une nouvelle image. Ce sera le logo du projet qui apparaîtra au centre légèrement à gauche de la page de connexion.

Réactualiser la page de connexion de Business Central pour apercevoir les changements.

2.3.2. Personnalisation de l'en-tête de l'application Business Central

Procédure 2.4. Modifier l'en-tête de l'application Business Central (bannière)

1. Démarrer le serveur EAP et ouvrir http://localhost:8080/business-central dans un navigateur web.
2. Connectez-vous à l'application Business Central par vos identifiants d'utilisateur.
3. Copier la nouvelle image d'en-tête de votre application dans le répertoire **`$EAP_HOME/standalone/deployments/business-central.war/images`** de votre installation JBoss BPM Suite.
4. Ouvrir le fichier **`$EAP_HOME/standalone/deployments/business-central.war/banner/banner.html`** dans un éditeur de texte.

5. Dans le fichier **banner.html**, modifier la balise <img> suivante pour donner un nom à votre nouvelle image d'en-tête :

```
<img src="banner/logo.png"/>
```

L'image par défaut est **logo.png**.

Réactualiser la page d'accueil de Business Central pour apercevoir les changements.

2.3.3. Personnalisation des fenêtres d'assistance de l'écran de démarrage de Business Central

Le répertoire **$EAP_HOME/standalone/deployments/business-central.war/plugins** contient les pages d'écran de démarrage et les fichiers html correspondants. Chaque page d'écran de démarrage (splash page) contient le nom du fichier html qui contient des informations sur le ou les image(s) à afficher. Par exemple, la page d'écran de démarrage **authoring_perspective.splash.js** pointe vers le fichier **authoring_perspective.splash.html**. Le fichier **authoring_perspective.splash.html** contient les noms et emplacements de tous les fichiers d'images qui apparaissent sur la page d'assistance de démarrage d'Authoring Perspective, ainsi que leurs légendes.

Procédure 2.5. Modifier les images et légendes des fenêtres contextuelles d'assistance de Business Central

1. Démarrer le serveur EAP et ouvrir http://localhost:8080/business-central dans un navigateur web.

2. Connectez-vous à l'application Business Central par vos identifiants d'utilisateur.

3. Copier la ou les nouvelle(s) image(s) d'assistance de la page de démarrage dans le répertoire **$EAP_HOME/standalone/deployments/business-central.war/images** de votre installation JBoss BPM Suite.

4. Ouvrir le fichier html correspondant du répertoire **$EAP_HOME/standalone/deployments/business-central.war/plugins** dans l'éditeur de texte.

5. Modifiez le fichier html pour qu'il pointe vers votre nouvelle image d'aide de l'écran de démarrage. Par exemple, pour modifier la première image qui apparaît dans l'aide de Perspective Authoring, modifiez la balise < img > suivante dans le fichier **authoring_perspective.splash.html** pour ajouter votre nouvelle image :

```
<img src="images/authoring_perspective1.png" alt="">
```

L'image par défaut est **authoring_perspective1.png**, qui apparaît dans l'assistance de la première page d'Authoring Perspective (perspective de création).

6. Pour modifier la légende de l'image qui apparaît sur la page d'aide de l'écran de démarrage, modifier les contenus des balises <h4> et <p> sous la balise <img> :

```
<h4>Authoring</h4>
<p>Modularized and customizable workbench</p>
```

Réactualiser la page d'accueil de Business Central et accéder aux fenêtre contextuelles d'assistance de la page de démarrage pour apercevoir les changements.

2.4. DESCRIPTEURS DE DÉPLOIEMENT

Les processus et les règles de Red Hat JBoss BPM Suite 6 sont stockés dans un empacketage basé Apache Maven, et sont appelés kjar (de l'anglais knowledge archives). Les règles, les processus, les ressources, etc. font partie d'un built de fichier jar, et sont gérés par Maven. Un fichier conservé dans le répertoire **`META-INF`** du kjar appelé **`kmodule.xml`** peut être utilisé pour définir les bases de connaissances et de sessions. Par défaut, ce fichier **`kmodule.xml`** est vide.

Quand un composant de runtime comme Business Central est sur le point de traiter un kjar, il consulte **`kmodule.xml`** pour construire la représentation de runtime.

Deployment Descriptors, les descripteurs de déploiement font partie d'une nouvelle fonctionnalité introduite dans la branche 6.1 de Red Hat JBoss BPM Suite, qui vous permet un contrôle affiné de votre déploiement et qui complémente le fichier **`kmodule.xml`**. La présence de ces descripteurs est optionnelle et votre déploiement pourra avoir lieu sans qu'ils soient présents. Les propriétés qui vous pouvez définir en utilisant ces descripteurs sont purement techniques de nature, et incluent des meta valeurs comme la stratégie de runtime, l'auditing et la persistance.

Ces descipteurs vous permettent de configurer le serveur d'exécution sur plusieurs niveaux (niveau serveur par défaut, différents descripteurs de déploiement par kjar etc.). Cela vous permet de faire des personnalisations simples à la configuration out-of-the-box du serveur d'exécution (probablement par kjar).

Vous pouvez définir ces descripteurs dans un fichier nommé **`kie-deployment-descriptor.xml`** et mettre ce fichier à côté de votre fichier **`kmodule.xml`** dans le dossier **`META-INF`**. Vous pouvez changer cet emplacement par défaut (et le nom du fichier) en le spécifiant comme paramètre système :

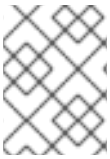
```
-Dorg.kie.deployment.desc.location=file:/path/to/file/company-deployment-descriptor.xml
```

2.4.1. Configuration du descripteur de déploiement

Les descripteurs de déploiement permettent à l'utilisateur de configurer le serveur d'exécution sur plusieurs niveaux :

- niveau serveur : le niveau principal, s'applique à tous les kjars déployés sur le serveur.
- niveau kjar : vous permet de configurer les descripteurs sur la base d'un kjar.
- niveau temps de déploiement : descripteurs qui s'appliquent quand un kjar est en cours de déploiement.

Les éléments de configuration granulaire spécifiés par les descripteurs de déploiement ont préséance sur ceux niveau serveur, sauf dans le cas d'éléments de configuration basés collection, qui sont fusionnés. La hiérarchie fonctionne comme ceci : *configuration du temps de déploiement* > *configuration kjar* > *configuration du serveur*.



NOTE

La configuration du temps de déploiement s'applique à des déploiements effectués via l'API REST.

Ainsi, si le mode de persistance (l'un des éléments que vous pouvez configurer) défini au niveau du serveur est ***NONE***, mais que le même mode est spécifié comme ***JPA*** au niveau kjar, le mode réel sera ***JPA*** pour cette kjar. Si rien n'est spécifié pour le mode de persistance dans le descripteur de déploiement pour ce kjar (ou s'il n'y a aucun descripteur de déploiement), cela renverra à la configuration niveau serveur, qui est dans ce cas ***NONE*** (aucun) (ou ***JPA*** s'il n'y a aucun descripteur de déploiement au niveau du serveur).

Pouvez-vous substituer ce comportement de mode de fusionnement par hiérarchie ?
Oui. Dans le mode par défaut, s'il existe des descripteurs de déploiement présents à plusieurs niveaux, les propriétés de configuration seront fusionnées avec les propriétés granulaires se substituant aux valeurs brutes, et avec les éléments de configuration manquants au niveau granulaire fournis par les valeurs présentes aux niveaux supérieurs. Le résultat final est une configuration fusionnée de descripteur de déploiement. Ce mode de fusion par défaut s'appelle le mode ***MERGE_COLLECTIONS***. Mais vous pouvez en changer (Section 2.4.2, « Gestion des descripteurs de déploiement ») si cela ne convient pas à votre environnement :

- KEEP_ALL: dans ce mode, toutes les valeurs de niveau supérieur remplacent toutes les valeurs de niveau inférieur (les valeurs de niveau serveur remplacent les valeurs kjar)
- OVERRIDE_ALL: dans ce mode, toutes les valeurs de niveau inférieur remplacent toutes les valeurs de niveau supérieur (les valeurs kjar remplacent les valeurs de niveau serveur)
- OVERRIDE_EMPTY: dans ce mode, tous les éléments de configuration *non vides* des niveaux inférieurs remplacent ceux des niveaux supérieurs, y compris les éléments qui sont représentés sous forme de collections.
- MERGE_COLLECTIONS (DEFAULT): dans ce mode, tous les éléments de configuration non vides de niveau inférieur remplacent les éléments de niveaux supérieurs (comme OVERRIDE_EMPTY), mais les propriétés de collection sont mergées (combinées).

Les descripteurs de déploiement des kjars dépendants sont mis à un niveau inférieur à celui auquel le kjar est déployé, mais ils sont toujours situés à un niveau supérieur par rapport au serveur.

Est-ce que je dois fournir un descripteur de déploiement complet pour tous les kjars ?
Non. Et c'est ici précisément que la fusion entre différents fichiers peut vous être utile. Il est possible et recommandé de fournir des descripteurs de déploiement partiels. Par exemple, si vous souhaitez substituer le mode auditing dans un kjar, vous aurez juste besoin de fournir ceci et le reste des valeurs sera fusionné du niveau serveur aux kjars de niveaux supérieurs.

Il est important de noter que quand on utilise le mode OVERRIDE_ALL, tous les éléments de configuration doivent être spécifiés car le kjar pertinent les utilisera toujours et ne mergera avec aucun autre descripteur de déploiement dans la hiérarchie.

Peut-on configurer ?
Les informations de configuration technique de haut niveau peuvent être configurées par les descripteurs de déploiement. Le tableau suivant en donne la liste, ainsi que des valeurs autorisées et par défaut pour chacun.

Tableau 2.1. Descripteurs de déploiement

Configuration	Entrée XML	Valeurs autorisées	Valeur par défaut
Nom d'unité de persistance pour les données de runtime	persistence-unit	N'importe quel nom valide de package de persistance	org.jbpm.domain

Configuration	Entrée XML	Valeurs autorisées	Valeur par défaut
Nom d'unité de persistance pour les données d'auditing	audit-persistence-unit	N'importe quel nom valide de package de persistance	org.jbpm.domain
Mode de persistance	persistence-mode	JPA, NONE	JPA
Mode d'auditing	audit-mode	JPA, JMS ou NONE	JPA
Stratégie de runtime	runtime-strategy	SINGLETON, PER_REQUEST ou PER_PROCESS_INSTANCE	SINGLETON
Liste des listeners d'événements à enregistrer	event-listeners	Noms de classe de listener comme **`ObjectModel`**	Pas de valeur par défaut
Liste des listeners d'événements de tâches à enregistrer	task-event-listeners	Noms de classe de listener comme **`ObjectModel`**	Pas de valeur par défaut
Liste de Work Item Handlers à enregistrer	work-item-handlers	Classes de Work Item Handlers données comme **`NamedObjectHandler`**	Pas de valeur par défaut
Liste des variables globales à enregistrer	globals	Variables globales valides données comme **`NamedObjectModel`**	Pas de valeur par défaut
Stratégies de marshalling à enregistrer (pour une persistance variable enfichable)	marshalling-strategies	Classes **`ObjectModel`** valides	Pas de valeur par défaut
Rôles exigés pour pouvoir avoir accès aux ressources du kjar	required-roles	Noms de rôles de string	Pas de valeur par défaut
Entrées environnementales supplémentaires pour les sessions de Knowledge	environment-entries	**`NamedObjectModel`** valide	Pas de valeur par défaut

Configuration	Entrée XML	Valeurs autorisées	Valeur par défaut
Options de configuration supplémentaires pour les sessions de Knowledge	configurations	`NamedObjectModel` valide	Pas de valeur par défaut

Comment procurer des valeurs aux éléments de configuration basés collections ?
Dans le tableau d'éléments de configuration valide vu plus haut, vous aurez sans doute remarqué que les valeurs valides pour les éléments basés collection sont soit `ObjectModel`, soit `NamedObjectModel`. Les deux sont similaires et fournissent une définition de l'objet à construire ou créer pendant le runtime, sauf que les informations sur l'objet `NamedObjectModel` nomment l'objet à observer. Ces deux types sont définis à l'aide d'un identificateur, de paramètres facultatifs et d'un résolveur (qui résoudre l'objet).

- identifier - définit toutes les informations sur l'objet, comme le nom de classe complet, l'id du bean Spring ou une expression MVEL.
- parameters - paramètres optionnels qui doivent être utilisés quand on crée des instances d'objets à partir de ce modèle.
- resolver - identificateur du programme de résolution qui sera utilisé pour créer des instances d'objet à partir du modèle - (reflection, mvel ou Spring).

Ainsi, si vous avez construit une stratégie de marshaling personnalisée et que vous souhaitez que vos déploiements utilisent cette stratégie plutôt que de la valeur par défaut, vous devrez fournir cette stratégie comme `ObjectModel`, avec pour identificateur `com.mycompany.MyStrategy`, le programme de résolution étant reflection (le plus simple et la valeur par défaut) et tous les paramètres requis pour que votre stratégie fonctionne. Reflexion servira ensuite pour créer une instance de cette stratégie en utilisant le nom de classe qualifié complet que vous avez fourni comme identificateur.

```
<marshalling-strategy>
 <resolver>reflection</resolver>
 <identifier>com.myCompany.MyStrategy</identifier>
 <parameters>
    <parameter xsi:type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
      param
    </parameter>
  </parameters>
</marshalling-strategy>
```

Dans le cas où un programme de résolution basé reflection ne suffise pas (comme c'est le cas dans l'exemple précédent), vous pourrez utiliser un programme de résolution basé MVEL comme identificateur du modèle d'objet. Quand vous évaluez les expressions, vous pouvez substituer les paramètres out-of-the-box. Exemple :

```
<marshalling-strategy>
  <resolver>mvel</resolver>
  <identifier>new com.myCompany.CustomStrategy(runtimeManager)
</identifier>
</marshalling-strategy>
```

Le programme de résolution Spring vous permet de chercher un bean par son identificateur à partir d'un contexte d'application Spring. Chaque fois que JBoss BPM Suite est utilisée avec Spring, ce programme de résolution aide à déployer les kjars dans l'environnement de runtime. A titre d'exemple (à noter que l'identificateur est, dans ce cas, un bean nommé en contexte Spring) :

```
<marshalling-strategy>
 <resolver>spring</resolver>
 <identifier>customStrategy</identifier>
</marshalling-strategy>
```

2.4.2. Gestion des descripteurs de déploiement

Les descripteurs de déploiement peuvent être modifiés via Business Central d'une des deux façons suivantes. Soit graphiquement (en cliquant sur **Création** → **Création de projet**, puis en sélectionnant **Outils** → **Descripteur de déploiement**) ou soit en cliquant sur le menu **Création** → **Administration**, puis en cliquant sur le dossier **`META-INF`** de File Explorer. Cliquer sur le fichier **`kie-deployment-descriptor.xml`** pour pouvoir le modifier manuellement.

À chaque fois qu'un projet est créé, un fichier **`kie-deployment-descriptor.xml`** de stockage est généré avec les valeurs par défaut décrites plus haut.

Remplacer le comportement de mode de fusionnement par hiérarhie

Pour changer le mode par défaut de MERGE_COLLECTIONS à KEEP_ALL, OVERRIDE_ALL ou OVERRIDE_EMPTY, vous pouvez utiliser une des méthodes suivantes, selon les besoins.

- Définir la propriété système org.kie.dd.mergemode à l'une des valeurs suivantes. Ce mode de fusion deviendra le mode par défaut de tous les kjars déployés dans le système, à moins que vous ne les remplaciez au niveau kjar par la prochaine méthode.
- Quand vous déployez une nouvelle unité de déploiement via Business Central (**Déployer** → **Déploiements**), vous pouvez sélectionner le mode de fusion à utiliser pour ce kjar particulier.
- Quand vous déployez via l'API REST, vous pouvez ajouter le paramètre de recherche mergemode à l'un de ces modes pour définir le mode de fusion pour ce déploiement.

Limiter l'accès à Runtime Engine

Un des éléments de configuration discutés plus tôt, required-roles, peut être modifié par les descripteurs de déploiement. Cette propriété limite l'accès au moteur d'exécution sur la base d'un kjar ou par niveau de serveur, en veillant à ce que l'accès à certains processus ne soit accordé qu'aux utilisateurs qui appartiennent à des groupes définis par cette propriété.

Le rôle de sécurité peut être utilisé pour limiter l'accès aux définitions de processus ou pour restreindre l'accès pendant le runtime.

Le comportement par défaut consiste à ajouter des rôles requis à cette propriété sur la base des restrictions du référentiel. Vous pouvez, biensûr, modifier ces propriétés manuellement si nécessaire, comme décrit ci-dessus, en fournissant des rôles qui correspondent aux rôles définis dans le domaine de sécurité.

2.5. GESTION DE LA POLITIQUE DE REMPLACEMENT DE DÉPLOIEMENT

Si un utilisateur essaie de déployer un artefact par un GAV (Group-Id, Artifact-Id et Version) qui existe déjà dans le système, le déploiement échouera et un message d'erreur s'affichera dans le panneau **`Messages`**.

Cette fonctionnalité empêche l'utilisateur de remplacer un déploiement existant par erreur.

Cette fonctionnalité est activée par défaut, c'est à dire que, par défaut, le système empêchera l'utilisateur de remplacer une installation existante par le même GAV.

Toutefois, il peut exister des cas quand l'utilisateur *souhaite* remplacer des déploiements existants ayant le même GAV. Bien que vous ne puissiez pas activer les remplacements par déploiement, vous pouvez installer cette fonctionnalité pour le système dans son ensemble en utilisant la configuration système org.kie.override.deploy.enabled. Ce paramètre est **`false`** par défaut. Modifiez-la à **`true`** pour activer le remplacement des déploiements ayant le même GAV, que vous procurez au moment du démarrage de votre serveur (**`-Dorg.kie.override.deploy.enabled=true`**).

2.6. ÉTENDRE BUSINESS CENTRAL

Depuis la version 6.1 de JBoss BPM Suite , Business Central peut être configuré pour ajouter de nouveaux écrans, des menus, des éditeurs, des splashscreens et des perspectives par l'administrateur. Ces éléments peuvent étendre les fonctionnalités de Business Central et sont accessibles via le menu **Extensions** et sont classifiées **`Gestion de plugins`**

Vous pouvez maintenant définir vos propres plugins basés Javascript ou HTML pour étendre Business Central et les ajouter sans vous soucier de copier les fichiers dans le système de fichiers sous-jacent. Ajouter un nouvel écran au système pour afficher les options de base de cette fonctionnalité.

2.6.1. Gestion des plugins

Vous pouvez accéder a l'écran **`Gestion Plugin`** en cliquant sur **Extensions** → **Gestion Plugin**. Vous afficherez ainsi l'écran **`Plugin Explorer`** qui donne la liste de tous les plugins : **`Plugin Perspective`**, **`Plugin Écran`**, **`Editor Plugin`**, **`Plugin Splashscreen`** et **`Menu dynamique`**. Ouvrez-en une et vous verrez tous les plugins existants pour chaque catégorie, y compris ceux qui ne sont as modifiables et qui sont générés par le système.

Nous allons créer un nouveau plugin qui fait écho à « Hello World » lorsque les utilisateurs visitent l'écran de ce plugin. En général, les étapes pour créer un nouveau plugin sont les suivantes :

- Créer un nouvel écran
- Créer une nouvelle perspective (et y ajouter le nouvel écran)
- Créer un nouveau menu (et y ajouter le nouvel écran)
- Apps (en option)

Ajouter un nouvel écran

Cliquer sur le bouton **`Nouvel ...`** et sélectionner **Nouvel écran**. On vous invitera à saisir le nom de ce nouvel écran. Saisir "HelloWorldJS" et appuyer sur le bouton **OK**. L'éditeur du plugin d'écran apparaîtra divisé en 4 sections : Modèle, CSS, JavaScript et Media.

NOTE

Tous les éléments créés manuellement entrent dans des catégories respectives, au cas où vous souhaitez les modifier plus tard. Dans ce cas, pour ouvrir l'éditeur de plugin d'écran à nouveau si vous le fermez, ouvrez la catégorie **`Plugin Écran`** et passez les écrans générés par le système pour aller vers le plugin créé manuellement, et cliquer dessus pour ouvrir l'éditeur de plugins d'écran à nouveau.

Le modèle est l'endroit où votre HTML ira, CSS est pour le style, JavaScript pour vos fonctions et Media pour télécharger et charger des images.

Comme nous faisons un simple plugin de Hello World, saisir le code suivant dans la section Template (Modèle) : **`< div >Mon ecran Helloworld< / div >`**. Il peut s'agir de n'importe quel code HTML, et vous pouvez utiliser les exemples **`Angular`** et **`Knockout`** fournis. Pour les besoins de cet exemple, nous n'utilisons pas un de ces exemples, mais vous pouvez en choisir un en le sélectionnant dans le menu déroulant au bas dans la section Template.

Saisir votre code JavaScript dans la section JavaScript. Certaines méthodes et propriétés sont définies pour vous, y compris **`main`**, **`on_close`** et **`on_open`**. Pour cette démo, sélectionnez **`on_open`** et saisir ce qui suit : **`function () { alert('Hello World'); }`**

Cliquer sur le bouton **`Enregistrer`** pour terminer la création de l'écran.

Ajouter une nouvelle Perspective

Une fois que l'écran est créé, vous devez créer une perspective sur laquelle cet écran va se trouver. Les perspectives peuvent être créées de la même façon que celle dont un écran est créé en cliquant sur le bouton**`Nouvelle...`** (nouvelle perspective) puis, en sélectionnant **`Nouvelle Perspective`**. Cela ouvrira un éditeur de plugin de perspective, un peu comme dans l'éditeur Screnn Plugin.

L'éditeur de perpective est comme un générateur de grille que l'on peut glisser / déplacer pour les écrans et composants HTML. Retirer les grilles existantes dans la partie droite, puis faites glisser une grille **`6 6`** sur le côté droit.

Ensuite, ouvrir la catégorie **`Composants`** et faites glisser un élément d'écran sur le côté droit (dans une grille). Ceci ouvrira la boîte de dialogue **`Modifier Composant`** où vous pouvez entrer votre écran créé à l'étape précédente (**`HelloWorldJS`**). Cliquer sur le bouton **`OK`**, puis sur le bouton  pour sauvegarder cette perspective. Saisir **`HelloWorldPerspective`**, puis **`Accueil`** dans le champ de nom de balise (et cliquer sur le bouton **`Ajouter Balise`**). Enfin, cliquer sur le bouton **`OK`** pour terminer l'enregistrement

Si vous avez besoin de charger cette perspective à nouveau, vous aurez besoin du nom que vous lui avez donnée car cette perspective n'apparaît pas dans la liste des perspectives. Pour charger, cliquer sur le bouton  et saisir le nom de la perspective.

Ajout d'un nouveau menu

La dernière étape dans la création de notre plugin est d'ajouter un menu dynamique d'où le nouvel écran/ou la nouvelle perspective peuvent être appelés. Pour ce faire, aller dans **Extensions** → **Gestion Plugin** puis cliquer sur le bouton **`Nouveau...`** pour sélectionner **`Nouveau Menu Dynamique`**. Nommer ce menu dynamique (HelloWorldMenu), puis cliquer sur le bouton **OK**. L'éditeur de menu dynamique s'ouvrira.

Saisir le nom de perspective (HelloWorldPerspective) comme **`Id d'activité`**, ainsi que le nom de la liste déroulante (HelloWorldMenuDropDown). Cliquer sur **OK**, puis cliquer sur le bouton **`Enregistrer`**.

Ce nouveau menu sera ajouté à votre plan de travail la prochaine fois que vous réactualiserez Business Central. Réactualiser maintenant pour voir **HelloWorldMenu** ajouté dans votre menu de niveau supérieur. Cliquer dessus pour que **HelloWorldMenuDropDown** apparaisse, et si vous cliquer dessus, votre écran ou perspective s'ouvrira accompagné de **`Hello World`**.

Vous venez de créer votre premier plugin !

Travailler avec des applications (optionnel)
Si vous créez plusieurs plugins, vous pourrez utiliser la fonctionnalité de répertoire d'applications pour organiser vos composants et plugins, au lieu de dépendre uniquement des entrées qui se trouvent dans le menu supérieur.

Quand vous sauvegardez une nouvelle perspective, vous pouvez ajouter des étiquettes (balises) et ces étiquettes (balises) sont utilisées pour associer une perspective à un répertoire d'applications. Vous pouvez ouvrir les répertoires d'applications en cliquant sur **Extensions** → **Apps**.

Le répertoire d'applications fournit un autre moyen d'ouvrir votre perspective. Lorsque vous avez créé votre **`HelloWorldPerspective`**, vous avez entré la balise **`Accueil`** de page d'accueil. L'annuaire d'applications par défaut contient un répertoire unique appelé **`Accueil`** auquel vous avez associé votre perspective. C'est là où vous le trouverez quand vous ouvrirez le répertoire d'applications. Vous pouvez cliquer dessus pour exécuter la perspective maintenant.

Vous pouvez créer plusieurs répertoires et associer des perspectives à ces répertoires en fonction des besoins fonctionnels et verticaux. Par exemple, vous pourriez créer un répertoire de ressources humaines et ensuite associer tous les perspectives HR à ce répertoire pour mieux gérer les applications.

Vous pouvez créer un nouveau répertoire en cliquant sur :

2.6.2. API JavaScript (JS) API pour Extensions

L'extensibilité de Business Central est réalisée grâce a un API de JavaScript (JS) sous-jacent qui sera automatiquement chargé s'il est présent dans le dossier **`plugins`** de l'application web de Business Central (normalement : {INSTALL_DIR}/business-central.war/plugins/) ou il peut être chargé par des appels réguliers de JavaScript.

Cet API est divisé en ensembles multiples suivant sa fonctionnalité.

- API Register Perspective : permet la création dynamique de perspectives. L'exemple ci-dessous crée un panneau par la méthode **`registerPerspective`** :

```
$registerPerspective({
    id: "Home",
    is_default: true,
    panel_type:
"org.uberfire.client.workbench.panels.impl.MultiListWorkbenchPanelPr
esenter",
    view: {
        parts: [
            {
```

```
                place: "welcome",
                min_height: 100,
                parameters: {}
            }
        ],
        panels: [
            {
                width: 250,
                min_width: 200,
                position: "west",
                panel_type:
"org.uberfire.client.workbench.panels.impl.MultiListWorkbenchPanelPr
esenter",
                parts: [
                    {
                        place: "YouTubeVideos",
                        parameters: {}
                    }
                ]
            },
            {
                position: "east",
                panel_type:
"org.uberfire.client.workbench.panels.impl.MultiListWorkbenchPanelPr
esenter",
                parts: [
                    {
                        place: "TodoListScreen",
                        parameters: {}
                    }
                ]
            },
            {
                height: 400,
                position: "south",
                panel_type:
"org.uberfire.client.workbench.panels.impl.MultiTabWorkbenchPanelPre
senter",
                parts: [
                    {
                        place: "YouTubeScreen",
                        parameters: {}
                    }
                ]
            }
        ]
    }
});
```

- API Editor : vous permet de créer des éditeurs de façon dynamique et de les associer à un type de fichier. L'exemple ci-dessous crée un exemple d'éditeur et l'associe au type de fichier **`filename`**.

```
$registerEditor({
    "id": "sample editor",
```

```
    "type": "editor",
    "templateUrl": "editor.html",
    "resourceType":
"org.uberfire.client.workbench.type.AnyResourceType",
    "on_concurrent_update":function(){
        alert('on_concurrent_update callback')

$vfs_readAllString(document.getElementById('filename').innerHTML,
function(a) {
            document.getElementById('editor').value= a;
        });
    },
    "on_startup": function (uri) {
        $vfs_readAllString(uri, function(a) {
            alert('sample on_startup callback')
        });
    },
    "on_open":function(uri){
        $vfs_readAllString(uri, function(a) {
            document.getElementById('editor').value=a;
        });
        document.getElementById('filename').innerHTML = uri;
    }
});
```

En plus des méthodes **on_startup** et **on_open** vues dans les exemples précédents, l'API expose les événements de callback (rappels) suivants pour le cycle de vie de l'éditeur :

- on_concurrent_update;
- on_concurrent_delete;
- on_concurrent_rename;
- on_concurrent_copy;
- on_rename;
- on_delete;
- on_copy;
- on_update;
- on_open;
- on_close;
- on_focus;
- on_lost_focus;
- on_may_close;
- on_startup;
- on_shutdown;

Vous pouvez afficher cet éditeur via un modèle html :

```
<div id="sampleEditor">
    <p>Sample JS editor (generated by editor-sample.js)</p>
    <textarea id="editor"></textarea>

    <p>Current file:</p><span id="filename"></span>
    <button id="save" type="button"
onclick="$vfs_write(document.getElementById('filename').innerHTML,
document.getElementById('editor').value,  function(a)
{});">Save</button>
    <br>

    <p>This button change the file content, and uberfire send a
callback to the editor:</p>
    <button id="reset" type="button"
onclick="$vfs_write(document.getElementById('filename').innerHTML,
'Something else',  function(a) {});">Reset File</button>
</div>
```

- API PlaceManager : les méthodes de cet API vous permettent de demander à Business Central d'afficher un composant particulier associ. à une cible :
  `$goToPlace("componentIdentifier");`

- API Register plugin : les méthodes de cet API vous permettent de créer des plugins dynamiques de façon dynamqiue (qui seront transformées dans les écrans de Business Central) via l'API JS.

```
$registerPlugin( {
    id: "my_angular_js",
    type: "angularjs",
    templateUrl: "angular.sample.html",
    title: function () {
        return "angular " + Math.floor(Math.random() * 10);
    },
    on_close: function () {
        alert("this is a pure JS alert!");
    }
});
```

Le plugin se réfère au modèle **`angular.sample.html`** :

```
<div ng-controller="TodoCtrl">
    <span>{{remaining()}} of {{todos.length}} remaining</span>
    [ <a href="" ng-click="archive()">archive</a> ]
    <ul class="unstyled">
        <li ng-repeat="todo in todos">
            <input type="checkbox" ng-model="todo.done">
            <span class="done-{{todo.done}}">{{todo.text}}</span>
        </li>
    </ul>
    <form ng-submit="addTodo()">
        <input type="text" ng-model="todoText" size="30"
placeholder="add new todo here">
```

```
        <input class="btn-primary" type="submit" value="add">
    </form>
    <form ng-submit="goto()">
        <input type="text" ng-model="placeText" size="30"
placeholder="place to go">
        <input class="btn-primary" type="submit" value="goTo">
    </form>
</div>
```

Un plugin peut être ajouté a un événement de Business Central par une série de rappels (callback) JavaScript :

- on_concurrent_update;
- on_concurrent_delete;
- on_concurrent_rename;
- on_concurrent_copy;
- on_rename;
- on_delete;
- on_copy;
- on_update;
- on_open;
- on_close;
- on_focus;
- on_lost_focus;
- on_may_close;
- on_startup;
- on_shutdown;

- API Register Splash Screen : utilise les méthodes de cette API pour créer des écrans de démarrage.

```
$registerSplashScreen({
    id: "home.splash",
    templateUrl: "home.splash.html",
    body_height: 325,
    title: function () {
        return "Cool Home Splash " + Math.floor(Math.random() * 10);
    },
    display_next_time: true,
    interception_points: ["Home"]
});
```

- API Virtual File System (VFS) API : avec cette API, vous pouvez lire ou écrire un fichier sauvegardé dans un système de fichiers par un appel asynchrone.

```
$vfs_readAllString(uri,  function(a) {
  //callback logic
});

$vfs_write(uri,content,  function(a) {
  //callback logic
})
```

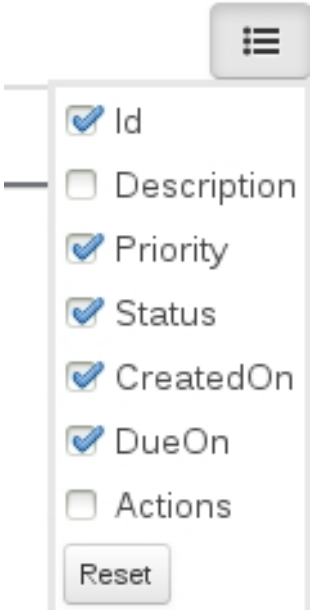
2.7. CONFIGURER DES COLONNES SUR UN TABLEAU

Business Central permet de configurer des vues qui contiennent des listes d'éléments sous forme de tableaux. Vous pouvez redimensionner les colonnes, déplacer des colonnes, ajouter ou supprimer la liste par défaut de colonnes et les trier. Cette fonctionnalité est fournie pour toutes les vues qui contiennent des tableaux.

Une fois que vous aurez fait des changements aux colonnes d'un affichage de colonnes, ces changements seront persistés pour l'utilisateur connecté.

Ajout et suppression de colonnes

Les tableaux qui permettent que les colonnes soient configurées ont un bouton  dans le coin à droite. Si vous cliquer sur ce bouton, vous ouvrirez une liste de colonnes qui peuvent être ajoutées ou supprimées dans le tableau par une case se situant à côté de chaque colonne :

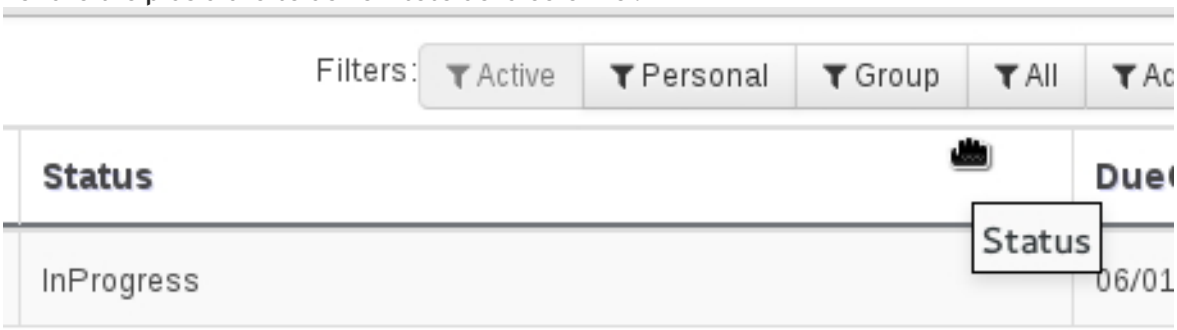


Redimensionner les colonnes

Pour redimensionner les colonnes, placer votre curseur entre les bordures de l'en-tête de la colonne et déplacer-la dans la direction souhaitée :

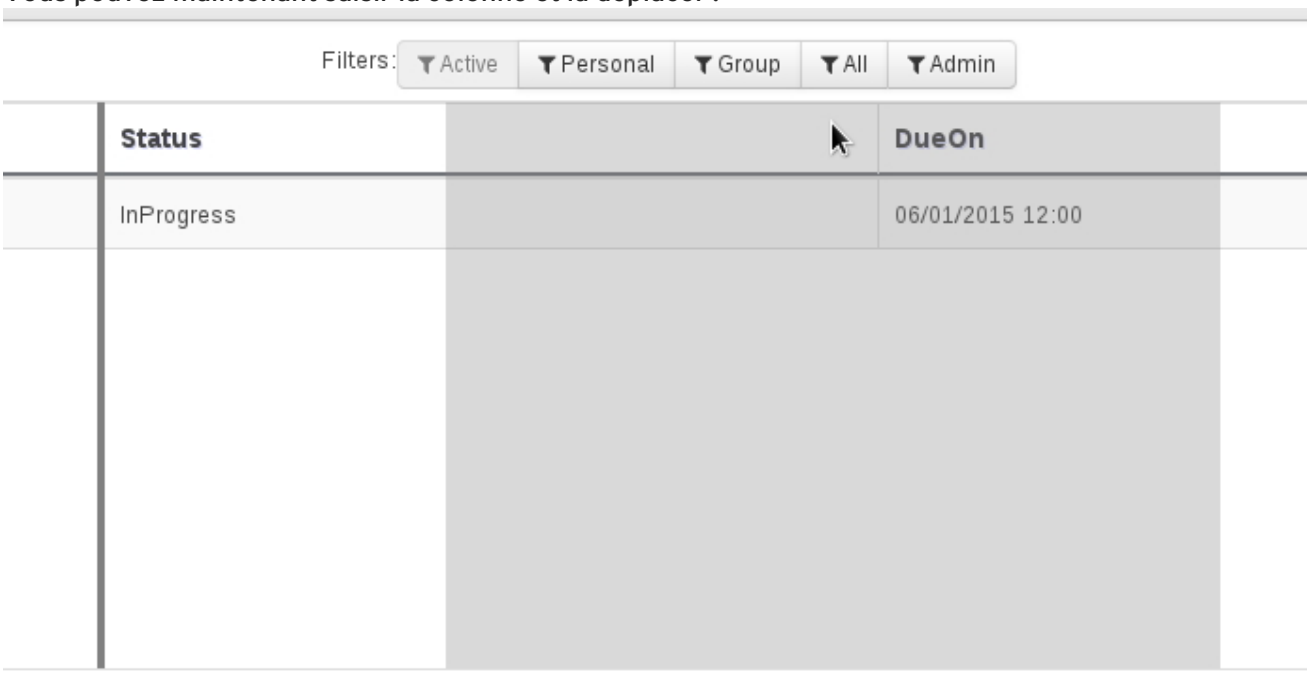
Déplacer des colonnes

Pour repositionner, glissez-déplacez une colonne dans une position différente, pointer votre souris sur l'endroit le plus à droite de l'en-tête de la colonne :



.

Vous pouvez maintenant saisir la colonne et la déplacer :



.

Déposer-le sur l'en-tête de colonne que vous souhaitez déplacer.

Ordonnancer les colonnes

Pour ordonnancer les colonnes, cliquer sur l'en-tête de colonne que vous souhaitez. Pour revenir en arrière sur un ordonnancement, cliquer à nouveau.

CHAPITRE 3. CONFIGURATION EN LIGNE DE COMMANDE

L'outil **kie-config-cli** est un outil de ligne de commande qui fournit des fonctions de gestion de répertoires de systèmes par ligne de commande et qui peut être utilise en ligne ou hors ligne.

1. **Online mode** (par défaut et conseillé) - au démarrage, l'outil se connecte a un répertoire Git par le serveur Git fourni par **kie-wb**. Toutes les modifications ont lieu localement et sont publiées en amont une fois que la commande push-changes a été lancée. Utilisez la commande exit pour publier les modifications locales. Pour ignorer les modifications locales à la sortie, utiliser la commande discard.

2. **Offline mode** (un style d'installateur) - crée et manipule le référentiel du système directement sur le serveur - il n'y a aucune option de rejet (discard).

L'outil se trouve dans le Red Hat Customer Portal . Pour télécharger kie-config-cli, procéder ainsi :

1. Rendez-vous dans Red Hat Customer Portal et connectez-vous.

2. Cliquer sur **Téléchargements** → **Téléchargements de produits**.

3. À partir de la page **Téléchargements de produits** qui s'ouvre, cliquer sur **Red Hat JBoss BPM Suite**.

4. À partir du menu déroulant **Version**, sélectionner 6.1.

5. Dans le tableau qui s'affiche, naviguer sur la ligne **Outils supplémentaires** et cliquer sur **Télécharger**.

Extraire le package zip pour obtenir des outils supplémentaires que vous pouvez télécharger de Red Hat Customer Portal. Il contient le répertoire **kie-config-cli-6.*MINOR_VERSION*-redhat-*x*-dist** avec fichier **kie-config-cli.sh**.

3.1. DÉMARRER L'OUTIL KIE-CONFIG-CLI TOOL EN MODE EN LIGNE

1. Pour démarrer l'outil kie-config-clie en ligne, naviguer dans le répertoire **kie-config-cli-6.*MINOR_VERSION*-redhat-*x*-dist** où vous avez installé l'outil et exécutez ensuite la commande suivante.

2. Dans un environnement Unix, exécuter :

```
./kie-config-cli.sh
```

Dans un environnement Windows, exécuter :

```
./kie-config-cli.bat
```

Par défaut, l'outil démarre en mode en ligne et demande les informations d'identification de l'utilisateur et une URL de Git pour se connecter (la valeur par défaut est git://localhost/system). Pour vous connecter à un serveur distant, remplacer l'hôte et le port par les valeurs appropriées. Exemple : **git://kie-wb-host:9148/system**

3.2. DÉMARRER L'OUTIL KIE-CONFIG-CLI TOOL EN MODE HORS LIGNE

Pour opérer en mode hors ligne, ajouter le paramètre hors ligne à la commande ci-dessous.

1. Naviguer dans le répertoire **kie-config-cli-6.*MINOR_VERSION*-redhat-*x*-dist** où vous avez installé l'outil.
2. Dans un environnement Unix, exécuter :

```
./kie-config-cli.sh offline
```

Dans un environnement Windows, exécuter :

```
./kie-config-cli.bat offline
```

L'exécution de cette commande change le comportement de l'outil et affiche une requête pour spécifier le dossier où se trouve le référentiel du système (**.niogit**). Si .niogit n'existe pas encore, la valeur du dossier peut être laissée vide et une toute nouvelle configuration sera créée.

3.3. COMMANDES DISPONIBLES POUR L'OUTIL KIE-CONFIG-CLI

Les commandes suivantes sont disponibles pour gérer le répertoire GIT par l'outil kie-config-cli :

- **add-deployment** - ajoute une nouvelle unité de déploiement
- **add-repo-org-unit** - ajoute un référentiel à l'unité organisationnelle
- **add-role-org-unit** - ajoute un ou des rôle(s) à l'unité organisationnelle
- **add-role-project** - ajoute un ou des rôle(s) a un projet
- **add-role-repo** - ajoute un ou des rôle(s) a un référentiel
- **create-org-unit** - crée une nouvelle unité organisationnelle
- **create-repo** - crée un nouveau référentiel git
- **discard** - ne publie aucun changement local, nettoie les répertoires temporaires et ferme l'outil
- **exit** - publie le travail, nettoie les répertoires temporaires et ferme l'outil
- **fetch-changes** - récupère les changements faits en amont dans le référentiel
- **help** - imprime les commandes disponibles avec leurs descriptions
- **list-deployment** - fait la liste des déploiements disponibles
- **list-org-units** - fait le liste des unités organisationnelles
- **list-repo** - fait les liste des référentiels disponibles
- **push-changes** - pousse les modifications dans le référentiel en amont (en mode en ligne uniquement)
- **remove-deployment** - supprime le déploiement existant

- **`remove-org-unit`** -supprime l'unité organisationnelle existante
- **`remove-repo`** - supprime un référentiel existant de la config uniquement
- **`remove-repo-org-unit`** - supprime un référentiel de l'unité organisationnelle
- **`remove--role-org-unit`** -supprime le ou les rôle(s) d'une unité organisationnelle
- **`add-role-project`** - supprime un ou des rôle(s) d'un projet
- **`remove-role-repo`** - supprime un ou des rôle(s) d'un référentiel

CHAPITRE 4. MIGRATION

La migration de vos projets de Red Hat JBoss BPM Suite 5 à Red Hat JBoss BPM Suite 6 exige une planification minutieuse et une évaluation étape par étape des différentes questions. Vous pouvez planifier des migrations, soit manuellement, soit en utilisant des processus automatiques. La plupart des migrations nécessiteront une combinaison des deux.

Comme JBoss BPM Suite 6 utilise GIT pour le stockage des ressources, des artefacts et des référentiels de code comprenant des procédures et des règles, vous devez commencer par créer un projet vide dans JBoss BPM Suite 6 comme base pour votre migration avec des fichiers factices comme espaces réservés pour vos différentes ressources et artefacts. En exécutant un clone GIT de ce projet vide dans votre IDE préféré, vous initierez le procesus de migration.

Sur la base des fichiers d'espace réservé dans votre projet cloné, vous pouvez commencer à ajouter des ressources dans les emplacements corrects. Le système de JBoss BPM Suite 6 est assez intelligent pour prendre ces changements en compte et les appliquer correctement. Assurez-vous bien que, lorsque vous importez des vieux fichiers de règles, qu'ils soient importés avec la bonne structure de nommage de package.

Comme Maven est utilisé pour créer des projets, les ressources de projets comme les règles, processus et modèles sont accessibles sous forme de simple fichier jar.

Cette section fait la liste des manières de migrer votre projet étape par étape. Cependant, il ne s'agit là que de lignes directrices, et la migration peut varier sérieusement de ce modèle.

En général, vous devez...

1. Commencer par migrer les données : ce sont vos ressources d'entreprise.
2. Puis, migrer vos processus de runtime.
3. Enfin, convertissez vos anciens appels d'API en nouveaux, l'un après l'autre.

Voyons ces étapes en détail dans les prochaines sections.

4.1. MIGRATION DES DONNÉES

Pour migrer des données de Red Hat JBoss BPM Suite, procédez ainsi :

1. Télécharger l'outil de migration en vous connectant sur le Portail client de Red Hat , puis naviguer vers la section de téléchargements des logiciels Red Hat JBoss BPM Suite. Cliquer sur **`Red Hat JBoss BPM Suite Migration Tool`** pour télécharger l'archive zip.
2. Décompresser l'archive zip téléchargée dans un répertoire de votre choix, et naviguer vers ce répertoire par l'intermédiaire d'une invite de commande. Ce répertoire contient quatre dossiers :
    - **`bin`** - contient les scripts de lancement.
    - **`jcr-exporter-libs`** - contient les fichiers libs spécifiques à la partie **`export-from-JCR`** de la migration.
    - **`jcr-importer-libs`** - contient les fichiers libs spécifiques à la partie **`import-from-Git`** de la migration.
    - **`conf`** - contient la configuration de l'outil de migration global.

3. Pour les bases de données de production, copier le pilote JDBC pour la base de données utilisée par le référentiel JCR du répertoire **`jcr-exporter-libs`** de l'outil de migration.

4. Exécuter la commande suivante :

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

Où :

- **`<source-path>`** est un chemin d'accès vers le référentiel JCR source.
- **`<desintation-path>`** est un chemin d'accès vers une destination GIT VFS. Ce dossier ne doit pas être déjà existant.
- **`<repository-name>`** un nom arbitraire de nouveau référentiel.

Le référentiel est ensuite migré vers l'emplacement spécifié.

En plus de la commande **`-i`**, vous pouvez également utiliser **`-h`** pour imprimer un message d'aide et **`-f`** qui oblige un écrasement du répertoire de sortie, éliminant ainsi le besoin de suppression manuelle de ce répertoire.

Comment importer un référentiel de Business Central

Le référentiel peut être importé dans Business Central en le clonant. Dans la perspective Administration, cliquer sur le menu **Référentiels**, puis cliquer sur le menu **Cloner Référentiel** pour démarrer le processus.

NOTE

Les ressources peuvent être migrées manuellement. Après tout, ce ne sont que des fichiers de texte. La spécification BPMN2 et la syntaxe DRL n'ont pas changé entre les versions.

Importer le référentiel dans JBDS

Pour importer le référentiel dans JBoss Developer Studio, proécéder ainsi

1. Démarrer JBoss Developer Studio
2. Démarrer le serveur Red Hat JBoss BPM Suite (s'il n'est pas déjà en cours d'exécution) en sélectionnant le serveur à partir de l'onglet serveur et en cliquant sur l'icône de démarrage.
3. Sélectionner **Fichier** → **Importer...** et naviguer dans le fichier Git. Ouvrir le dossier Git et sélectionner **Projets de Git** puis cliquer Suite.
4. Sélectionner la source du référentiel en tant que **Sortir du référentiel local** et cliquer sur Suite.
5. Sélectionner le référentiel qui doit être configuré à partir de la liste de référentiels disponibles.
6. Importer le projet en tant que projet général dans la prochaine fenêtre et cliquer sur Suite. Nommer ce projet et cliquer sur Terminer.

4.2. MIGRATION DE RUNTIME

Pour exécuter les processus de Red Hat JBoss BPM Suite 5 dans Red Hat JBoss BPM Suite 6, procéder ainsi :

1. Démarrer la propriété système **`jbpm.v5.id.strategy`** à true dans le fichier JBoss BPM Suite **`standalone.xml`** :

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

2. Charger la KieSession comme suit :

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

3. Continuer l'exécution normale du processus en utilisant les méthodes :

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

4.3. API ET COMPATIBILITÉ RÉTROACTIVE

Migration vers Version 6.1

Dans la version 6.1, les API 5.x ne sont plus pris en charge.

Red Hat JBoss BPM Suite n'assure plus la compatibilité rétroactive avec les règles, événements et interfaces de programmation d'applications de processus (API) à partir de JBoss BRMS 5. Le contenu du fichier **`knowledge-api JAR`** n'est plus supporté dans la version 6.1 et est remplacé par des API contenues dans le fichier **`kie-api JAR`** qui ont été introduites dans JBoss BPM Suite 6.0.

Si vous avez utilisé une ancienne API 5.x (se trouvant dans **`knowledge-api.jar`**), veuillez migrer (réécrire) les appels d'API dans la nouvelle API de KIE. Notez bien que plusieurs autres API ont changé entre JBoss BRMS 5.x et JBoss BPM Suite 6.x, à savoir l'API de service de tâches et l'API REST.

Migration vers Version 6.0

Le système de 6 JBoss BRMS assure la compatibilité rétroactive avec les interactions de processus, règles, et événements de JBoss BRMS 5. Vous devez éventuellement migrer (réécrire) ces interactions sur le tout nouvel API de base nouvellement conçu car cette compatibilité rétroactive est susceptible d'être obsolète.

Si vous ne pouvez pas migrer votre code pour utiliser la nouvelle API, vous pouvez utiliser l'API fournie par le jar **`connaissances-api`** spécialement conçu pour le code rétro-compatible. Cette API est l'interface publique qui sert à travailler dans JBoss BPM Suite et JBoss BRMS et elle est rétro-compatible.

Si vous utilisez à la place l'API REST dans JBoss BPM Suite 5, notez que cela a changé aussi, et qu'il n'existe aucun mécanisme de rétro-compatibilité dedans.

4.4. MIGRATION DU SERVICE DE TÂCHES

JBoss BPM Suite 6 prend en charge un serveur de tâches en cours d'exécution locale uniquement. Cela signifie que vous n'avez pas besoin de configurer n'importe quel service de messagerie dans votre projet. Cela diffère de JBoss BPM Suite 5 parce qu'il fournissait un serveur de tâches relié au moteur de base, en utilisant, le plus souvent, du système de messagerie fourni par HornetQ

Pour vous aider à combler l'écart jusqu'à ce que vous migriez ceci dans votre architecture actuelle, il y

a une méthode d'utilitaire ou assistant, **`LocalHTWorkItemHandler`**.

Comme l'API TaskService fait partie de l'API publique, vous allez maintenant devoir refactoriser vos importations à cause des changements de packages ou bien, vous allez devoir refactoriser vos méthodes à cause des changements au niveau des API elles-mêmes.

CHAPITRE 5. GESTION DES DONNÉES

5.1. SAUVEGARDE DES DONNÉES

Quand vous appliquez un mécanisme de sauvegarde à Red Hat JBoss BPM Suite , assurez-vous bien de sauvegarder les ressources suivantes :

- tout descripteur de déploiement personnalisé (tel que, **web.xml**, **jboss-web.xml**, **jboss.xml**)
- tous les fichiers de propriétés personnalisés

NOTE

Considérez la sauvegarde du fichier **business-central.war** total et des fichiers **dashbuilder.war**.

5.2. INSTALLATION DES INDEXES

Installation des indexes de clés étrangères

Certaines bases de données, comme Oracle et Postgres, ne créent pas automatiquement un index pour chaque clé étrangère. Cela peut entraîner les blocages. Pour éviter cette situation, il faut créer un index sur toutes les clés étrangères, en particulier dans la base de données Oracle.

Installation d'indexes pour le tableau de tâches et de processus

Le tableau de bord de processus et de tâches de 6.1 a été refactorisé pour faire face à un volume élevé d'instances de processus et de tâches. Afin d'obtenir des temps de réponse satisfaisants lors de l'interrogation de la base de données, les suites BPM de JBoss doivent être indexées : **processinstancelog** et **bamtasksummary**.

Notez que *TOUTES* les colonnes de ces deux tableaux ont besoin d'être indexées, pas seulement les clés primaires ou étrangères.

5.3. INSTALLER LA BASE DE DONNÉES

L'application dashbuilder requiert une base de données existante, qui aura été créée avant d'exécuter l'application. Pour créer une base de données, vous pouvez utiliser n'importe quel outil client de base de données et exécuter les commandes suivantes :

Postgres

L'énoncé sql suivant est utilisé pour créer une base de données Postgres :

```
CREATE DATABASE dashbuilder
  WITH ENCODING='UTF8'
         OWNER=dashbuilder
         CONNECTION LIMIT=-1
```

NOTE

La codification de la base de données doit être UTF8

DB2

La base de données DB2 peut être créée en utilisant l'énoncé sql suivant :

```
CREATE DATABASE dashb PAGESIZE 16384
```

NOTE

La taille de page des systèmes DB2 par défaut est de 4k, ce qui n'est pas suffisant pour la taille des colonnes du tableau dashbuilder. La taille d'une page doit être forcée à 16384 comme le montre l'énoncé suivant.

Une fois que la base de données est créée, la source de données du serveur d'applications doit être configurée. Vous devez modifier le fichier de configuration de JBoss EAP et configurer le sous-système de la source de données comme dans l'un des exemples suivants :

```
<datasource jndi-name="java:jboss/datasources/jbpmDS" enabled="true" use-
java-context="true" pool-name="postgresqlDS">
    <connection-url>jdbc:postgresql://localhost/test</connection-url>
    <driver>postgresql</driver>
    <pool></pool>
    <security>
       <user-name>sa</user-name>
       <password>sa</password>
    </security>
</datasource>
<drivers>
    <driver name="postgresql" module="org.postgresql">
        <xa-datasource-class>org.postgresql.xa.PGXADataSource</xa-
datasource-class>
    </driver>
</drivers>
```

5.4. MODIFIER LA BASE DE DONNÉES

Dashbuilder exige que JBoss BPM Suite ait des tableaux de bases de données de journalisation de l'historique. Il est impératif de déployer la console de tâches humaines (ou un sur-ensemble, c'est à dire : kie-wb) pour commencer. Sinon, le tableau de bord n'est pas initialisé correctement et il ne sera pas possible d'afficher ses indicateurs de performance clés.

Par défaut, l'application est configurée de façon à utiliser une source de données ayant le nom JNDI suivant :

```
java:jboss/datasources/ExampleDS
```

C'est spécifié dans le fichier de configuration de JBoss EAP; par exemple, `standalone.xml`.

NOTE

Cette source de données est à but de développement/démo ; elle est présente par défaut dans les installations JBoss.

Si vous souhaitez déployer dans une base de données différente d'H2 comme Oracle, MySQL, Postgres ou MS SQL Server, veuillez procéder aux étapes suivantes :

Procédure 5.1. Changer de base de données

1. Installer le pilote de base de données sur JBoss (voir la documentation du pilote de JBoss).
2. Créer une base de données vide et une source de données de JBoss qui se connecte au pilote de la base de données.
3. Modifier le fichier **`dashbuilder.war/WEB-INF/jboss-web.xml`** :

```
<jboss-web>
   <context-root>/dashbuilder</context-root>
   <resource-ref>
       <res-ref-name>jdbc/dashbuilder</res-ref-name>
       <res-type>javax.sql.DataSource</res-type>
       <jndi-name>java:jboss/datasources/myDataSource</jndi-name>
   </resource-ref>
   ...
```

4. Remplacer la valeur du paramètre jndi-name par le nom du chemin JNDI de la source de données JBoss que vous venez de créer.
5. Modifier le fichier **`dashbuilder.war/WEB-INF/jboss-deployment-structure.xml`**
6. Ajouter l'extrait de configuration suivant à l'intérieur de la balise **`deployment`**, ou **`jdbcDriverModuleName`** correspond au nom du module de pilote de JBoss JDBC :

```
   <dependencies>
       <module name="jdbcDriverModuleName" />
   </dependencies>
```

5.5. SCRIPTS DDL

Les scripts DDL de tableaux de bases de données pour JBoss BRMS et BPM Suite sont disponibles par téléchargement via le portail client. Ces scripts vous permettent d'étudier les tableaux et de les utiliser pour créer les tableaux et les index manuellement ou dans les bases de données qui ne sont pas directement prises en charge.

Pour télécharger ces scripts, connectez-vous à Customer Portal et cliquer sur Red Hat JBoss BPM Suite. Sélectionner la version du produit que vous souhaitez, puis cliquer sur **`Téléchargement`** sur la ligne **`Red Hat JBoss BPM Suite 6.x.x Supplementary Tools`** pour télécharger les outils supplémentaires.

Décompresser le fichier dans votre machine. Les scripts DDL se situent dans le dossier **`ddl-scripts`**. Les scripts de bases de données sont fournis pour DB2, H2, MySQL5, Oracle, PostgreSQL et SQLServer.

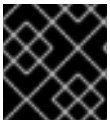
Le diagramme Entity Relationship complet se trouve à l'adresse suivante Red Hat Solution.

CHAPITRE 6. RÉFÉRENTIEL DE RESSOURCES

Les règles métier, les fichiers de définitions de processus et les autres ressources créées dans Business Central sont stockées dans le référentiel de ressources (Asset), connu également sous le nom de Knowledge Store.

Le Knowledge Store est un référentiel centralisé de connaissances commerciales. Il est connecté au référentiel GIT, ce qui vous permet de stocker différentes ressources de connaissances et artefacts en un seul et même endroit. Business Central vous offre une base web frontale qui permet aux utilisateurs de visualiser et de mettre à jour le contenu stocké. Vous pouvez y accéder via **`Project Explorer`** à partir de l'environnement unifié de Red Hat JBoss BPM Suite.

6.1. CRÉATION D'UN RÉFÉRENTIEL



IMPORTANT

Notez que seul un utilisateur ayant le rôle **`ADMIN`** peut créer un référentiel.

Procédure 6.1. Création d'un nouveau référentiel

1. Ouvrez la perspective **`Administration`** : à partir du menu principal, cliquer sur **Création** → **Administration**.
2. Dans le menu perspective, cliquer sur **Référentiels** → **Nouveau référentiel**.
3. La fenêtre **`Création de référentiel`** s'ouvre.

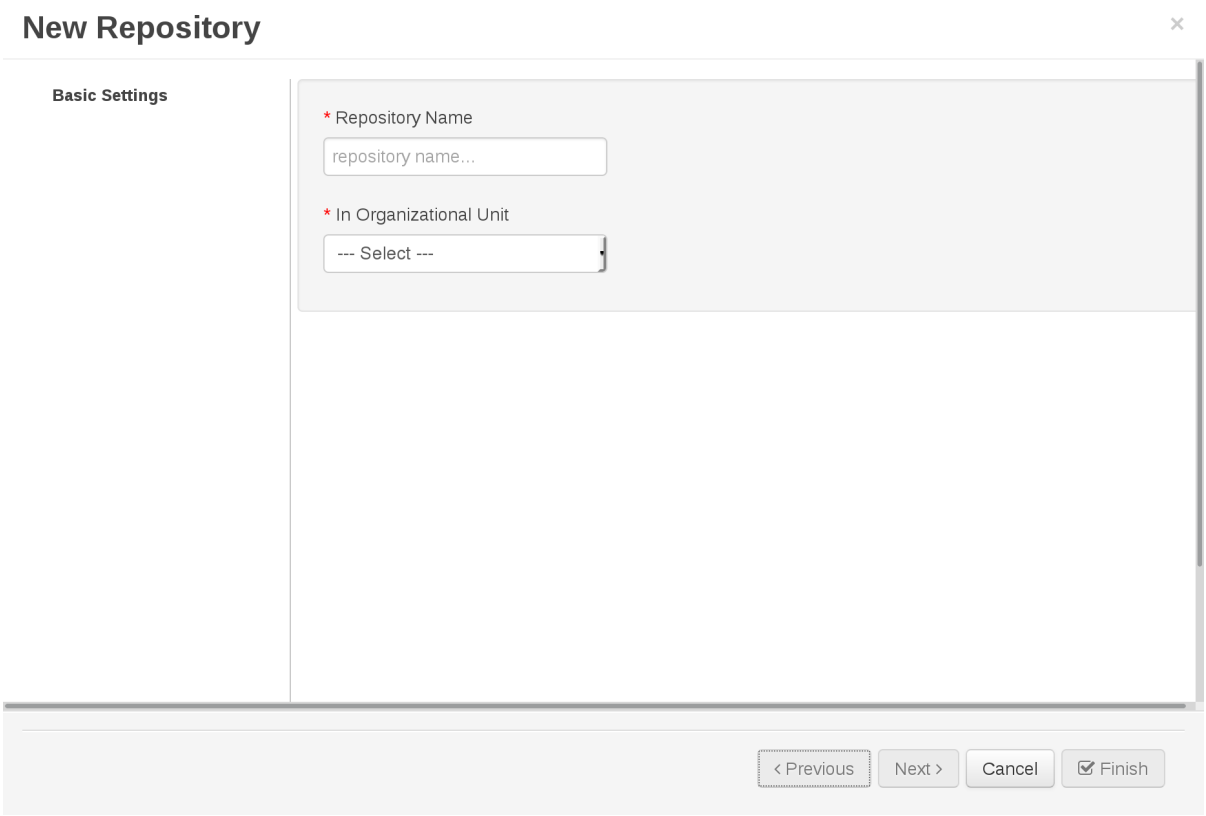
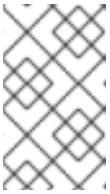


Figure 6.1. Fenêtre de création de référentiel

4. Saisir les informations obligatoires :

- Nom de référentiel.



NOTE

Notez que le nom du référentiel doit avoir un nom de fichier valide. Évitez d'utiliser un espace ou un caractère spécial qui puisse mener à un nom de dossier non valide.

- Sélectionner une unité organisationnelle dans laquelle le référentiel doit être créé à partir du menu déroulant **`Unité organisationnelle`**.

5. Cliquer sur **`Terminer`**

Le nouveau référentiel peut être visualisé avec **`File Explorer`** ou **`Project Explorer`**.

6.2. CLONAGE D'UN RÉFÉRENTIEL

IMPORTANT

Notez que seul un utilisateur ayant le rôle **`ADMIN`** peut cloner un référentiel.

Procédure 6.2. Clonage d'un référentiel

1. Ouvrez la perspective **`Administration`**.
2. Dans le menu **`Référentiels`**, sélectionner **`Clonage d'un référentiel`**.
3. La fenêtre **`Clonage d'un référentiel`** s'ouvre.

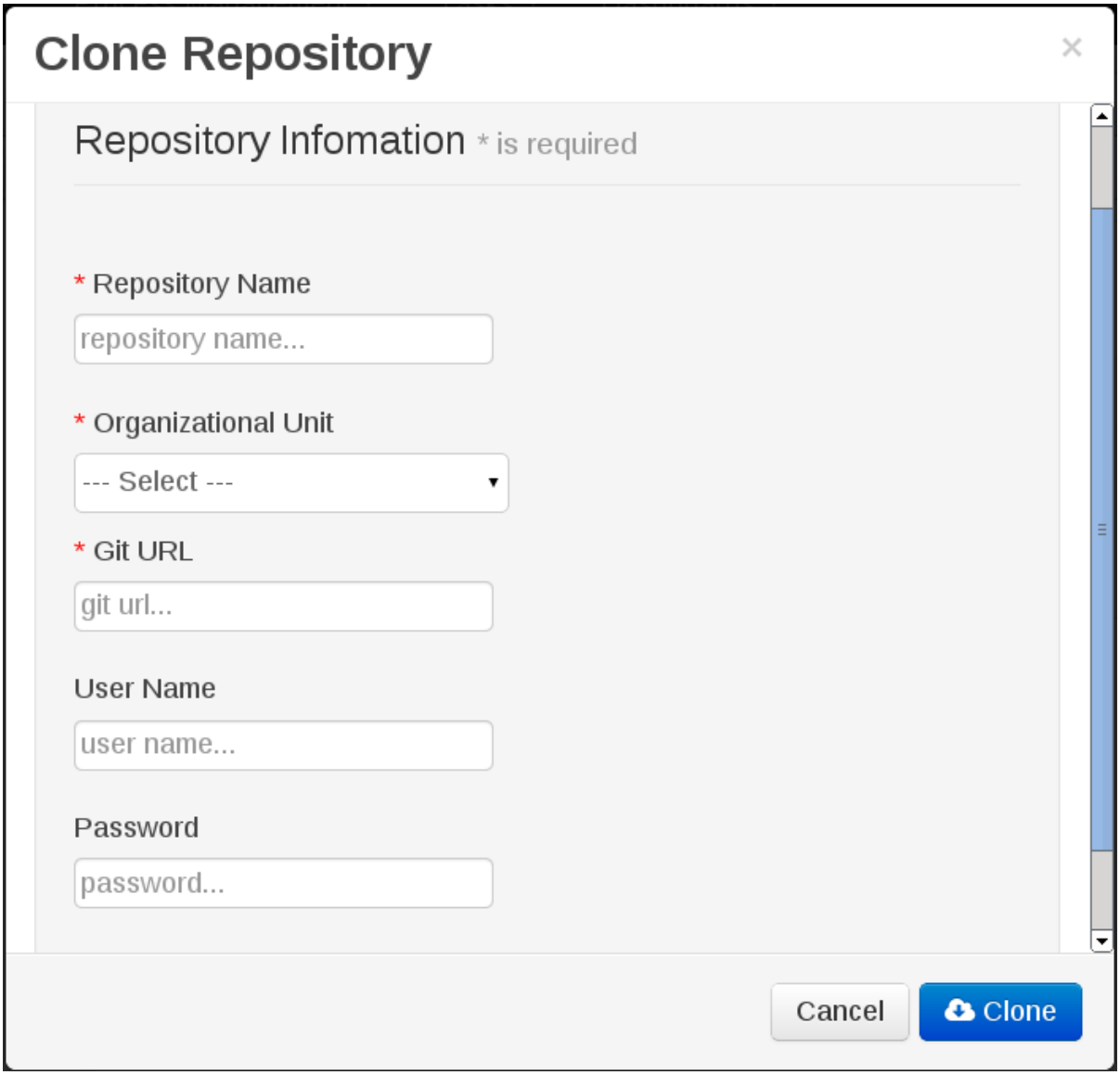


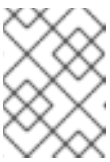
Figure 6.2. Fenêtre de clonage d'un référentiel

4. Dans la boîte de dialogue **`Clonage d'un référentiel`**, saisir les informations sur le référentiel :

   a. Saisir le **`Nom de référentiel`** à utiliser comme identifiant pour le référentiel de ressources Asset et sélectionner l'**`Unité organisationnelle`** dans laquelle il devra être ajouté.

   b. Saisir l'URL du référentiel GIT :

      - Pour un référentiel local : **`file:///path-to-repository/reponame`**
      - Pour un référentiel déjà existant ou distant : **`git://hostname/reponame`**



NOTE

Le protocole de fichier n'est pris en charge que pour les opérations 'READ'. Les opérations 'WRITE' ne sont *pas* prises en charge.

   c. Si nécessaire, saisir les **`Nom d'utilisateur`** et **`Mot de passe`** à utiliser pour l'authentification quand on clone le référentiel.

5. Cliquer sur **Cloner**.

6. Vous apercevrez une invite avec un bouton **OK** qui notifie l'utilisateur que le référentiel a été créé. Cliquer sur **OK**. Le référentiel sera indexé. Certaines fonctionnalités de workbench risquent de pas être disponibles tant que l'indexation est en cours.

Le référentiel cloné peut être vérifié grâce à **File Explorer** ou **Project Explorer**.

6.3. SUPPRIMER UN RÉFÉRENTIEL

On peut supprimer les référentiels par l'une des procédures suivantes :

Comment supprimer un référentiel de Business Central

La manière la plus simple de supprimer un référentiel est d'utiliser le **RepositoryEditor** de Business Central.

Procédure 6.3. Utiliser Business Central pour supprimer un référentiel

1. Accéder au **RepositoryEditor** dans Business Central **Création** → **Administration**.

2. Sélectionner les **Référentiels** du menu arborescent sur la gauche.

3. Sur la droite du **RepositoryEditor**, chercher le référentiel à supprimer de la liste des référentiels disponibles.

4. Sélectionner **master** dans le menu descendant, et cliquer sur le bouton **Suppression**.

5. Le message suivant apparaîtra :

   ```
   Êtes-vous certain de vouloir supprimer le référentiel "
   <$RepositoryName>" ? Certains éditeurs de texte deviennent soudain
   inopérables si on ne peut plus accéder à leur contenu.
   ```

   Cliquer sur **Ok** pour supprimer.

Supprimer un référentiel par l'outil kie-config-cli

Les référentiels peuvent être supprimés grâce à l'outil **kie-config-cli** par la commande **remove-repo**.

Pour plus d'informations concernant l'outil **kie-config-cli**, voir Chapitre 3, *Configuration en ligne de commande*.

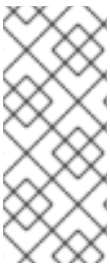
Supprimer un référentiel par l'API REST

Pour supprimer un référentiel du Knowledge Store, lancer l'appel d'API REST **DELETE**. Cet appel assume que l'utlisateur a créé une session HTTP authentifiée avant de lancer cette commande.

Exemple 6.1. Supprimer un référentiel avec curl

```
curl -H 'Accept: application/json' -H 'Content-Type: application/json' -
X DELETE 'localhost:8080/business-
central/rest/repositories/REPOSITORY_NAME'
```

6.4. GESTION DES RESSOURCES



NOTE

Le contenu de cette section est classifiée aperçu technologique de la version Red Hat JBoss BPM Suite. Il est fourni tel quel et aucun support n'est offert.

Pour activer et utiliser les fonctionnalités décrites ici, vous devrez vous connecter à Business Central par un utilisateur à qui on a donné le rôle spécial de **`kiemgmt`**.

Pour faciliter la gestion des projets, Red Hat JBoss BPM Suite offre maintenant une façon de gérer plusieurs projets sur la base de standards établis. Cela permet de créer des structures de référentiels conformes aux meilleures pratiques de l'industrie pour les questions de maintenance, de contrôle de version et de distribution de vos projets.

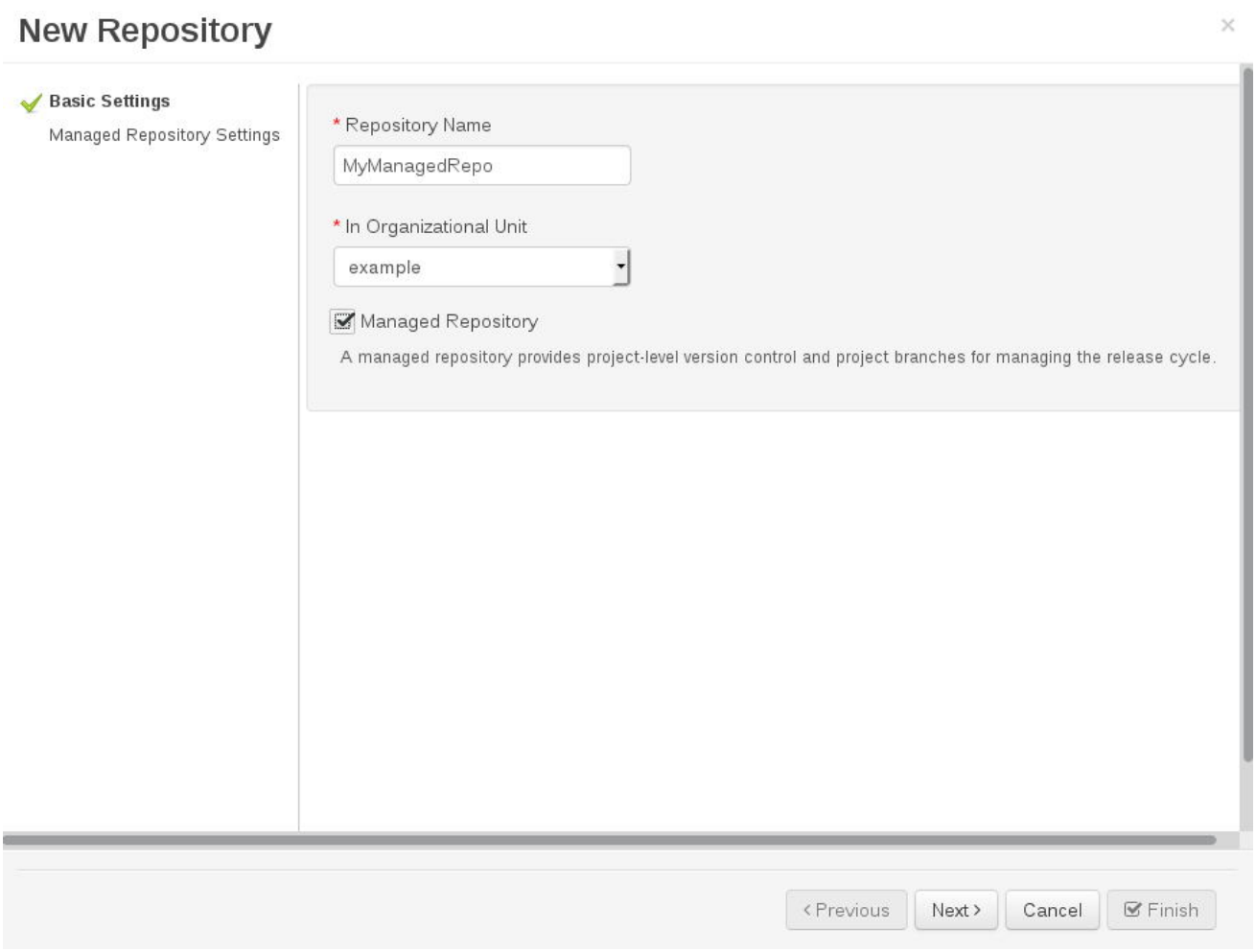
Pour commencer, les référentiels peuvent maintenant être gérés ou non.

Référentiels gérés ou non-gérés

Les référentiels non gérés correspondent aux structures de référentiels auquelles vous êtes habitués. Ils peuvent contenir plusieurs projets sans aucun lien entre eux.

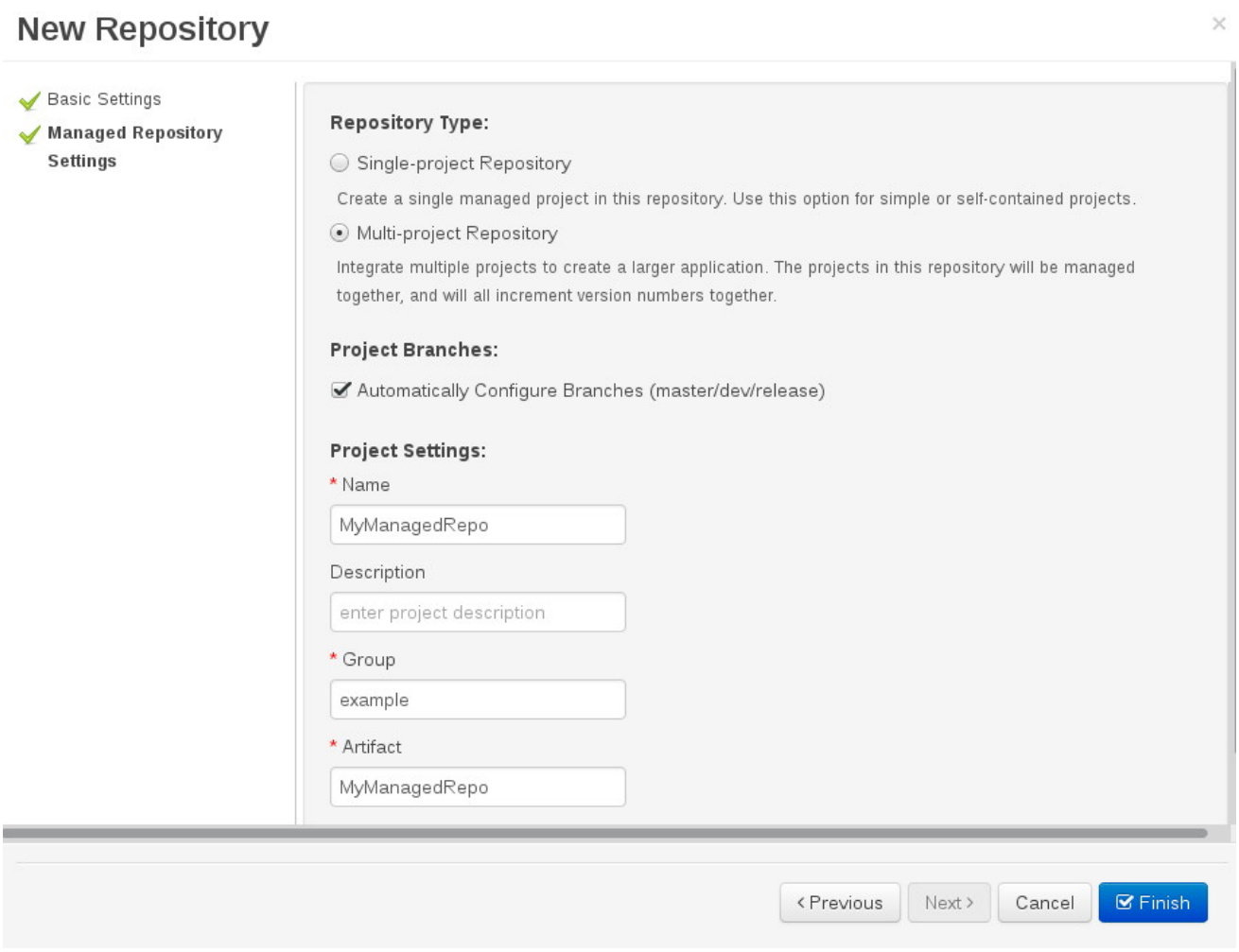
Les référentiels gérés, d'autre part, offrent un contrôle de version au niveau du projet et des branches de projets pour gérer le cycle de sortie de projet. De plus, les référentiels de projets peuvent se limiter à un seul projet ou réunir plusieurs projets. Quand un référentiel géré est créé, le processus de configuration de la gestion des ressources est lancé automatiquement afin de créer des branches de référentiel, et la structure de projet correspondante est également créée.

Pour créer un référentiel Géré ou Non-géré, ouvrir une page de création de nouveau référentiel. Allez à **Création** → **Administration**, puis cliquer sur **Référentiels** → **Nouveau référentiel**. Cela vous mènera à la page **`Nouveau référentiel`**.



La création d'un référentiel non-géré est la même qu'auparavant. Saisir le nom du référentiel et sélectionner l'unité organisationnelle à laquelle il appartient, puis cliquer sur le bouton **`Terminer`**.

Pour créer un référentiel géré, sélectionner la case **`Référentiel géré`**, après avoir donné un nom au référentiel et après avoir sélectionné l'unité organisationnelle à laquelle il appartient. Cliquer le bouton **`Suite`** pour saisir des informations sur le référentiel géré.

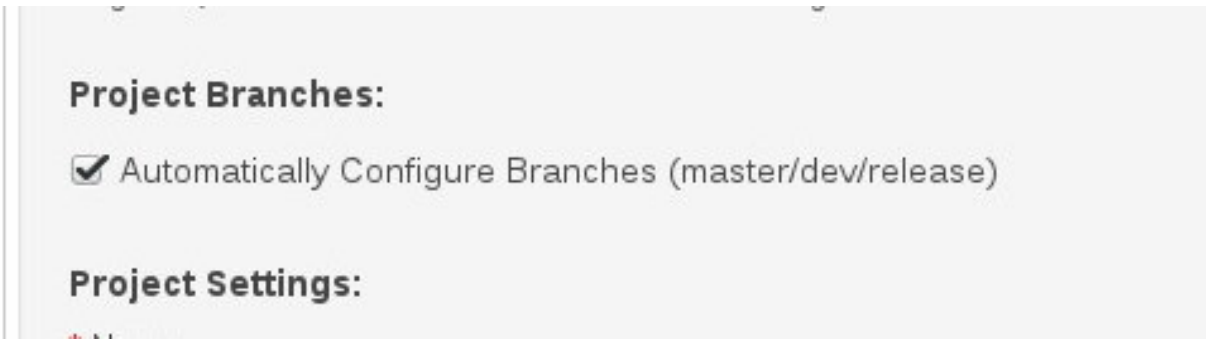


Sélectionner l'étiquette **`Projet simple`** si le projet que vous créez est un simple projet autonome. Saisir les informations sur le projet géré, ainsi que les détails GAV. Vous ne pourrez pas ajouter plus de projets à ce référentiel par la suite.

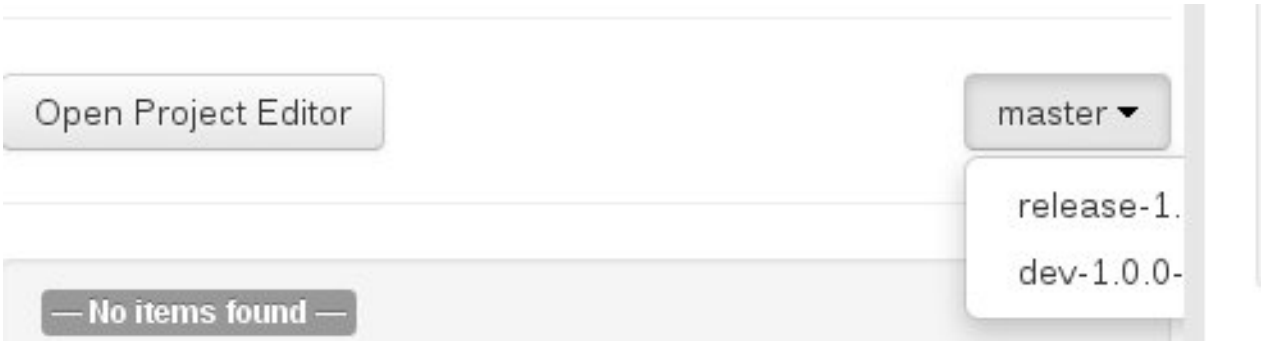
Pour des projets plus complexes, pour lesquels il s'agit sans doute d'un projet parent comprenant plusieurs petits projets, sélectionner le référentiel **`Projet multiple`**. Tous les projets créés dans un référentiel multi-projets seront gérés ensemble, avec leurs numéros de projets incrémentés également. Aussi, saisir les détails du projet parent et le GAV, qui sera hérité par tous les projets futurs que vous allez créer dans le référentiel géré.

Branches gérées

Les référentiels gérés ont l'avantage de venir avec des branches gérées. Comme dans GIT, vous pouvez choisir de travailler sur des branches différentes de votre projet (par exemple : master, dev ou sorties de version). Ce processus de branching peut être fait pour vous automatiquement, si vous sélectionnez la case quand vous créez un nouveau référentiel géré (à la fois pour les projets simple ou multiples).

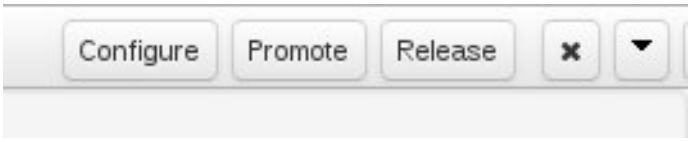


Vous pouvez passer d'une branche à l'autre en sélectionnant la branche voulue quand vous travaillez dans Project Explorer.

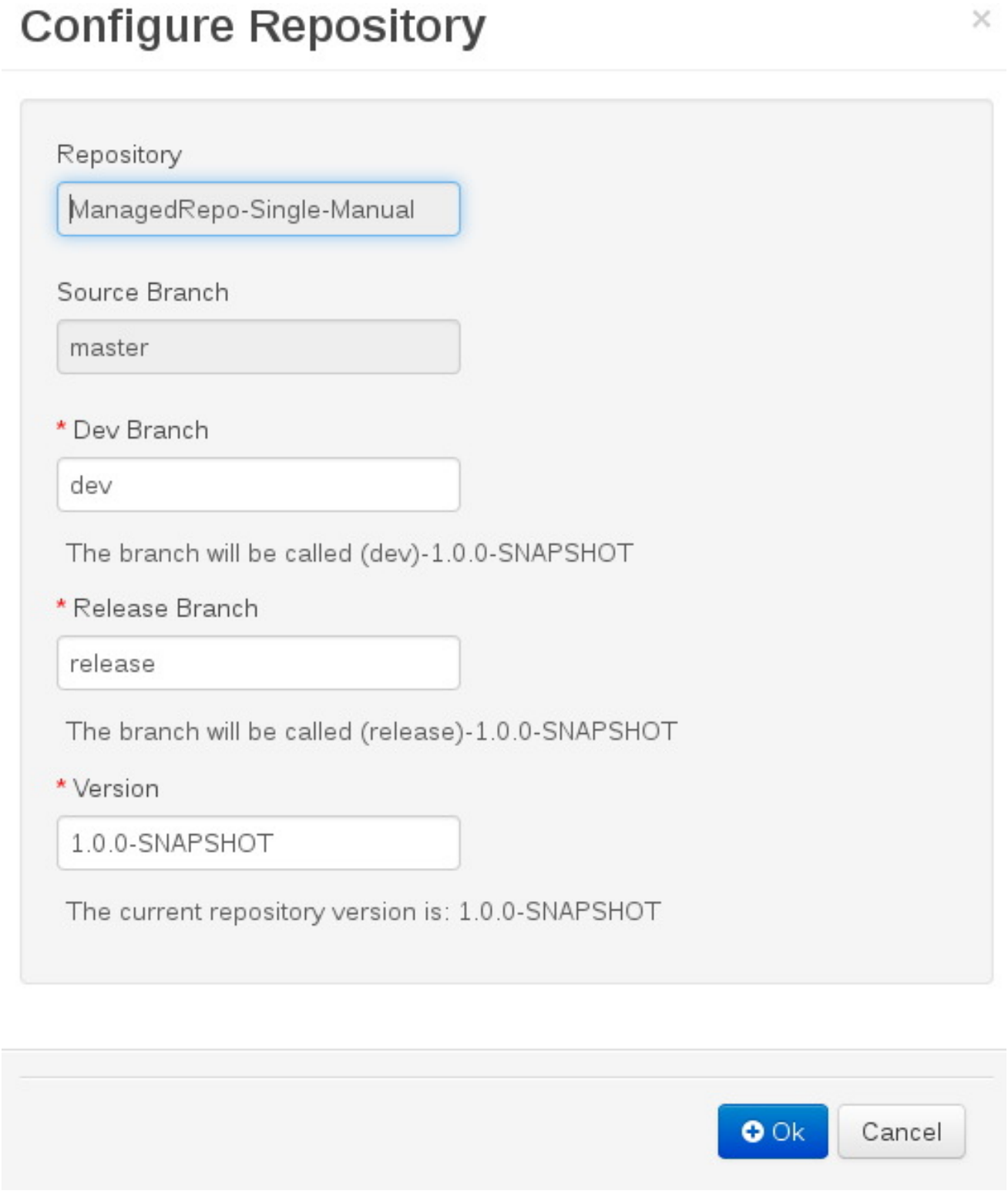


Structure du référentiel

Si vous ne sélectionnez pas le branching automatique lors de la création d'un référentiel, vous pourrez créer des branches manuellement par la suite. Pour les référentiels gérés, vous pouvez le faire en utilisant le bouton **`Configurer`**. Ce bouton, ainsi que des boutons **`Release`** et **`Diffuser`**, est donné sur la page **`Structure du référentiel`**. Vous pouvez accéder à cette page, en cliquant sur **Référentiel** → **Structure du référentiel** dans le menu de perspective



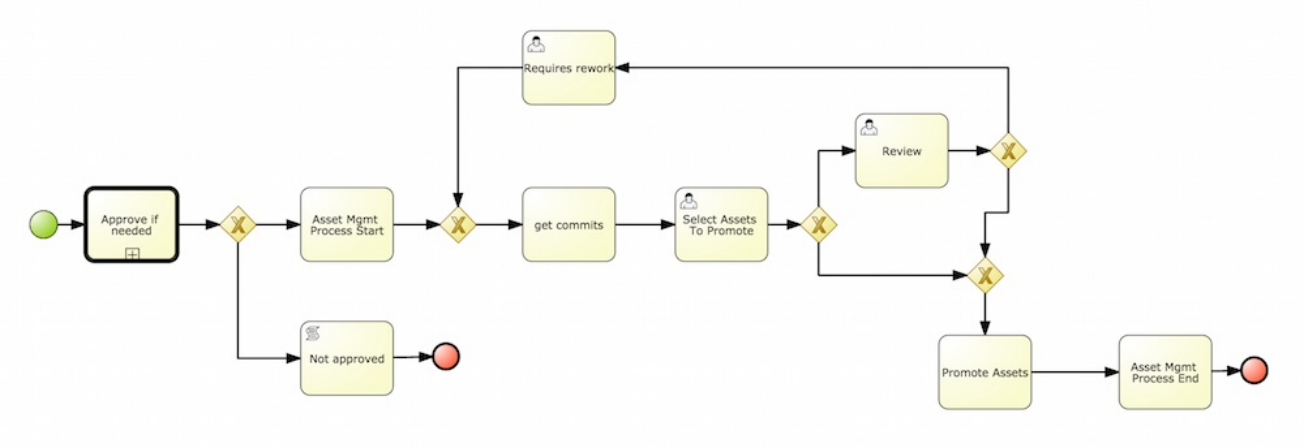
. En cliquant sur le bouton **`Configurer`**, vous pourrez créer des branches ou modifier celles qui sont créées automatiquement.



Vous pouvez promouvoir des actifs de la branche master à d'autres branches en utilisant le bouton **`Promouvoir`**. De même, vous pouvez libérer les branches et les déployer sur le serveur en utilisant le bouton **`Release`**.

Ces deux fonctions sont contrôlées en interne par l'utilisation de processus prédéfinis qui sont déployées sur votre instance. Par exemple, lorsque vous cliquer sur le bouton **`Promouvoir`** après avoir effectué des travaux sur votre branche de développement, un processus de promotion des changements démarre en arrière-plan. Un utilisateur, avec le rôle **`kiemgmt`** verra une tâche utilisateur apparaître dans cette liste de tâches, lui indiquant de revoir les ressources à promouvoir. Cet utilisateur peut alors demander cette tâche, et décider de promouvoir tous, certains ou aucun des éléments de ressources. Le processus sous-jacent va choisir les commits sélectionnées par l'utilisateur dans une branche de sortie de version. Cet utilisateur peut également demander un nouvel

examen de ces ressources, et ce processus peut être répété plusieurs fois jusqu'à ce que toutes les ressources soient prêtes à sortir dans une nouvelle version. Le flux de ce processus est illustré ci-dessous :

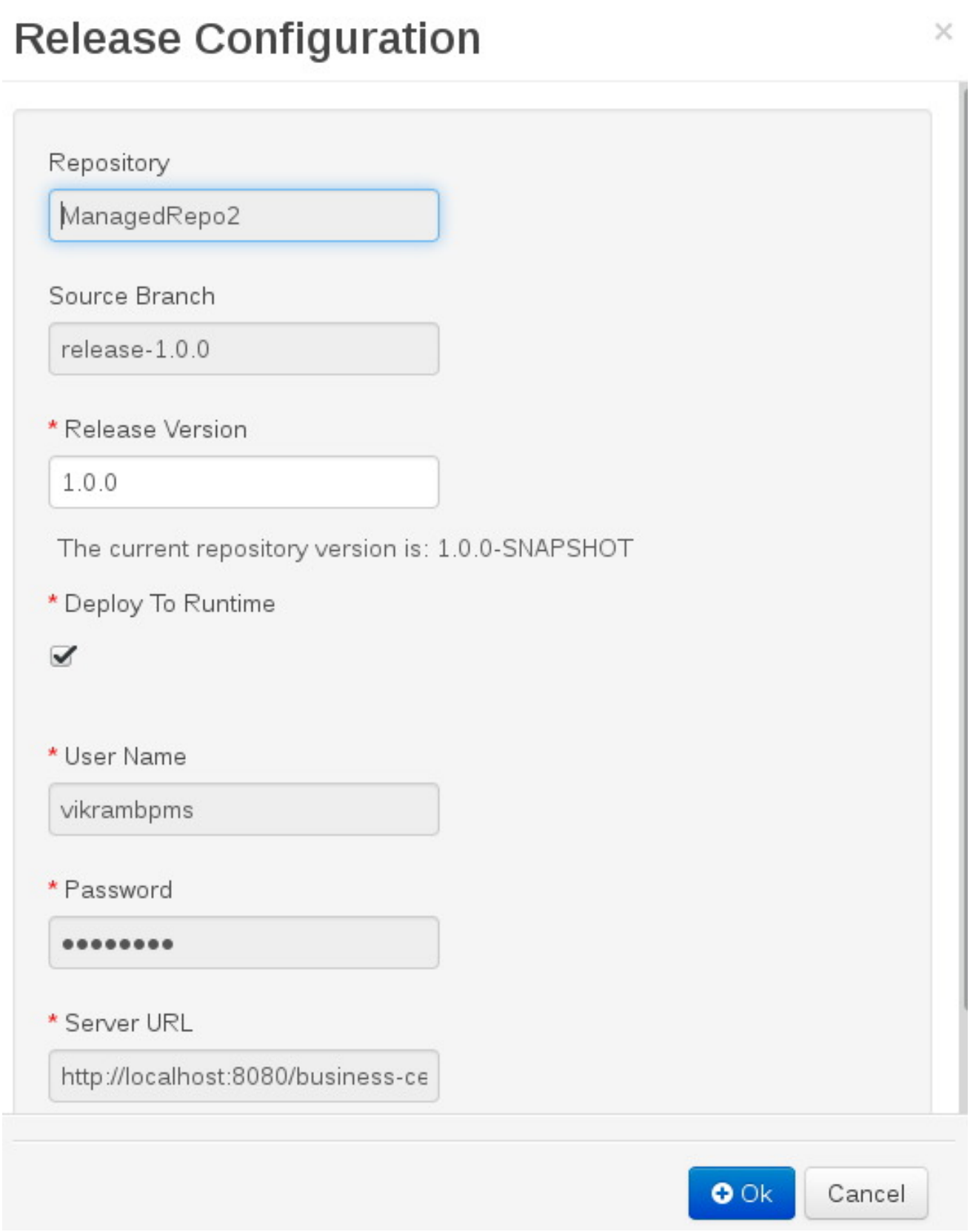


De même, lorsque vous cliquer sur le bouton **`Release`**, un flux de processus de sortie de version est initié. Ce flux de processus génère le projet, met à jour tous les artefacts Maven pour la prochaine version et déploie le projet à l'exécution, si les informations de déploiement sont fournies.

AVERTISSEMENT

Les branches de projets à faire sortir dans les nouvelles versions doivent commencer par le mot clé `release`

.



6.5. RÉFÉRENTIEL MAVEN

Maven est un outil de gestion de projets de logiciels qui utilise un fichier POM (Project Object Model) pour gérer :

- Les builds
- La documentation

- Les rapports
- Les dépendances
- Les versions
- Les SCM
- La distribution

Un référentiel Maven est utilisé pour contenir et stocker des artefacts de builds et des dépendances de projets. Il est en général de deux types :

- Local : se réfère à un référentiel local où toutes les dépendances de projet sont stockées et se trouvent dans l'installation en cours comme « m2 » dans le dossier par défaut. C'est un cache des téléchargements à distance qui contient également les artefacts de builds temporaires non encore sortis.
- Remote (distant) : se réfère à n'importe quel autre type de référentiel qui puisse être accessible par une variété de protocoles tels que file:// ou http://. Ces référentiels peuvent être dans un lieu éloigné, mis en place par un tiers pour le téléchargement des artefacts ou un référentiel interne mis en place sur un fichier ou un serveur HTTP, utilisé pour partager des objets privés entre les équipes de développement pour la gestion des versions internes.

6.6. CONFIGURER UN DÉPLOIEMENT POUR UN RÉFÉRENTIEL NEXUS DISTANT

Nexus est un gestionnaire de référentiels souvent utilisé dans les organisations pour centraliser le stockage et la gestion des artefacts de développement de logiciels. Il est possible de configurer votre projet pour que les artefacts produits par chaque build soient automatiquement déployés dans un référentiel sur un serveur Nexus distant.

Pour configurer votre projet afin qu'il déploie des artefacts dans un référentiel Nexus distant, ajouter un élément **`distributionManagement`** à votre fichier **`pom.xml`** de projet, comme l'illustre l'exemple ci-dessous.

```
<distributionManagement>
  <repository>
    <id>deployment</id>
    <name>Internal Releases</name>

<url>http://your_nexus_host:8081/nexus/content/repositories/releases</url>
  </repository>
  <snapshotRepository>
    <id>deployment</id>
    <name>Internal Releases</name>

<url>http://your_nexus_host:8081/nexus/content/repositories/snapshots/</ur
l>
  </snapshotRepository>
</distributionManagement>
```

Remplacer les URL de l'exemple par les URL réels de vos référentiels Nexus. Le référentiel spécifié dans l'élément **`snapshotRepository`** est utilisé quand le qualificateur **`-SNAPSHOT`** est ajouté au numéro de version actuel du projet. Dans les autres cas, le référentiel spécifié dans l'élément du

repository sera utilisé.

Si votre serveur Nexus requiert une authentification, vous devrez également modifier vos paramètres de projet Maven pour ajouter vos informations d'identification dans le fichier **paramètres-security.xml** à l'aide d'un mot de passe maître. Par défaut, ce fichier est dans le dossier **~/.m2**, sauf si vous avez changé son emplacement en modifiant la propriété système kie.maven.settings.custom.

```
<servers>
  <server>
    <id>deployment</id>
    <username>admin</username>
    <password>admin.123</password>
  </server>
</servers>
```

Avec cette configuration en place, en cliquant sur le bouton **Construire et Déployer** de Business Central, vous exécutez une build Maven et déployez les artefacts du build dans le référentiel local et dans l'un des référentiels Nexus indiqués dans le fichier **pom.xml**.

6.7. CONFIGURATION SYSTÈME

Dans JBoss EAP, pour changer une propriété de Business Central, comme la configuration de SSH, procédez ainsi :

Procédure 6.4. Modifier les propriétés système

1. Modifier le fichier **$JBOSS_HOME/domain/configuration/host.xml**
2. Cherchez les éléments XML qui appartiennent au main-server-group et ajouter la propriété système. Par exemple :

```
<system-properties>
 <property name="org.uberfire.nio.git.dir" value="..." boot-
time="false"/>
 ...
</system-properties>
```

Voici une liste de toutes les propriétés système disponibles :

- **org.uberfire.nio.git.dir** : emplacement du répertoire .niogit. Par défaut : le répertoire de travail
- **org.uberfire.nio.git.daemon.enabled** : active/désactive le démon GIT. Par défaut: true
- **org.uberfire.nio.git.daemon.host** : si le démon GIT est activé, utiliser cette propriété comme identifiant d'hôte local. Valeur par défaut : localhost
- **org.uberfire.nio.git.daemon.port** : si le démon GIT est activé, utiliser cette propriété comme numéro de port. Valeur par défaut : 9418
- **org.uberfire.nio.git.ssh.enabled** : active/désactive le démon SSH. Par défaut: true

- **`org.uberfire.nio.git.ssh.host`** : si le démon SSH est activé, utiliser cette propriété comme identifiant d'hôte local. Valeur par défaut : localhost
- **`org.uberfire.nio.git.ssh.port`** : si le démon SSH est activé, utiliser cette propriété comme numéro de port. Valeur par défaut : 8001
- **`org.uberfire.nio.git.ssh.cert.dir`** : emplacement du répertoire **`.security`** où les certificats locaux seront stockés. Valeur par défaut : répertoire de travail.
- **`org.uberfire.metadata.index.dir`** : emplacement du dossier **`.index`** de Lucene. Par défaut : le répertoire de travail
- **`org.uberfire.cluster.id`** : nom du cluster Helix, par example : kie-cluster
- **`org.uberfire.cluster.zk`** : string de connexion à Zookeeper. Se présente sous la forme : **`host1:port1,host2:port2,host3:port3`**. Par exemple : **`localhost:2188`**.
- **`org.uberfire.cluster.local.id`** : id unique du noeud de cluster Helix. Notez que ':' est remplacé par '_'. Par exemple : node1_12345.
- **`org.uberfire.cluster.vfs.lock`** : nom de la ressource définie sur le cluster Helix, par exemple : kie-vfs
- **`org.uberfire.cluster.autostart`** : cette valeur retarde la mise en cluster VFS jusqu'à ce que l'application soit entièrement initialisée pour éviter les conflits lorsque tous les membres de cluster créent des clones locaux.
- **`org.uberfire.sys.repo.monitor.disabled`** : désactive le moniteur de configuration (ne pas désactiver à moins de savoir ce que vous faîtes). Valeur par défaut : false
- **`org.uberfire.secure.key`** : mot de passe utilisé par le cryptage de mots de passe. Valeur par défaut : org.uberfire.admin
- **`org.uberfire.secure.alg`** : algorythme Crypto utilisé par le crytage de mots de passe. Valeur par défaut : PBEWithMD5AndDES
- **`org.guvnor.m2repo.dir`** : endroit où le dossier de référentiel Maven est stocké. Valeur par défaut : working-directory/repositories/kie
- **`org.kie.example.repositories`** : dossier à partir desquels les référentiels de démo seront clonés. Les référentiels de démo doivent avoir été obtenus et doivent être placés dans ce dossier. Cette propriété système prévaut sur les propriétés org.kie.demo et org.kie.example. Par défaut : Not used (non utilisé).
- **`org.kie.demo`** : active le clone externe d'une application démo de GitHub. Ce système a priorité sur org.kie.example. Valeur par défaut : true.
- **`org.kie.example`**: active la structure de l'exemple composé par un référentiel, une unité organisationnelle ou un projet. Valeur par défaut : false.

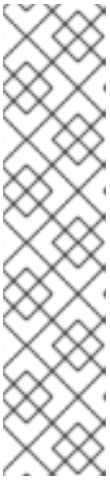
CHAPITRE 7. EXPORTATION ET IMPORTATION D'UN PROCESSUS

7.1. CRÉER UNE DÉFINITION DE PROCESSUS

Assurez-vous de vous être bien connecté à JBoss BPM Suite ou si vous êtes dans JBoss Developer Studio au référentiel connecté.

Pour créer un processus, procéder ainsi :

1. Ouvrir la perspective Création de projet (**Création** → **Création Projet**).
2. Dans **`Project Explorer`** (**Création Projet** → **Project Explorer**), naviguer jusqu'au projet où vous souhaitez créer la définition de processus (dans la vue **`Projet`**, sélectionner le projet et le référentiel qui conviennent dans les listes déroulantes ; et dans la vue **`Référentiel`**, naviguer jusqu'au répertoire ***`REPOSITORY/PROJECT`*`/src/main/resources/`**).



NOTE

Il est conseillé de créer vos ressources, y compris les définitions de processus, dans un package de projet pour permettre l'importation de ressources et leur référencement. Pour créer un package, procéder ainsi :

- Dans la vue **`Référentiel`** de Project Explorer, naviguer dans le répertoire ***`REPOSITORY/PROJECT`*`/src/main/resources/`**.
- Aller dans **Nouvel élément** → **Package**.
- Dans la boîte de dialogue **`Nouvelle ressource`**, définir le nom du package et vérifier l'emplacement du package dans le référentiel.

3. Dans le menu perspective, cliquer sur **Nouvel élément** → **Business Process**.
4. Dans la boîte de dialogues **`Nouveaux Processus`**, saisir le nom du processus et cliquer sur le bouton **OK**. Attendez que l'éditeur de processus apparaisse avec le diagramme de processus.

7.2. IMPORTATION D'UNE DÉFINITION DE PROCESSUS

Pour importer une définition BPMN2 ou JSON existante, procéder ainsi :

1. Dans **`Project Explorer`**, sélectionner un projet et un package respectif dans lequel vous souhaitez importer la définition du processus.
2. Créer un nouveau processus métier en allant dans **Nouvel Élément** → **Business Process**.
3. Dans la barre d'outils de Process Designer, cliquer sur l'icône **`importation`**  dans la barre d'outils Éde l'éditeur et choisissez le format de la définition de processus importé. Notez que vous devez choisir de remplacer la définition de processus existante pour importer.
4. À partir de la fenêtre **`Importer`**, cherchez le fichier de processus (Process) et cliquer sur **`Importer`**.

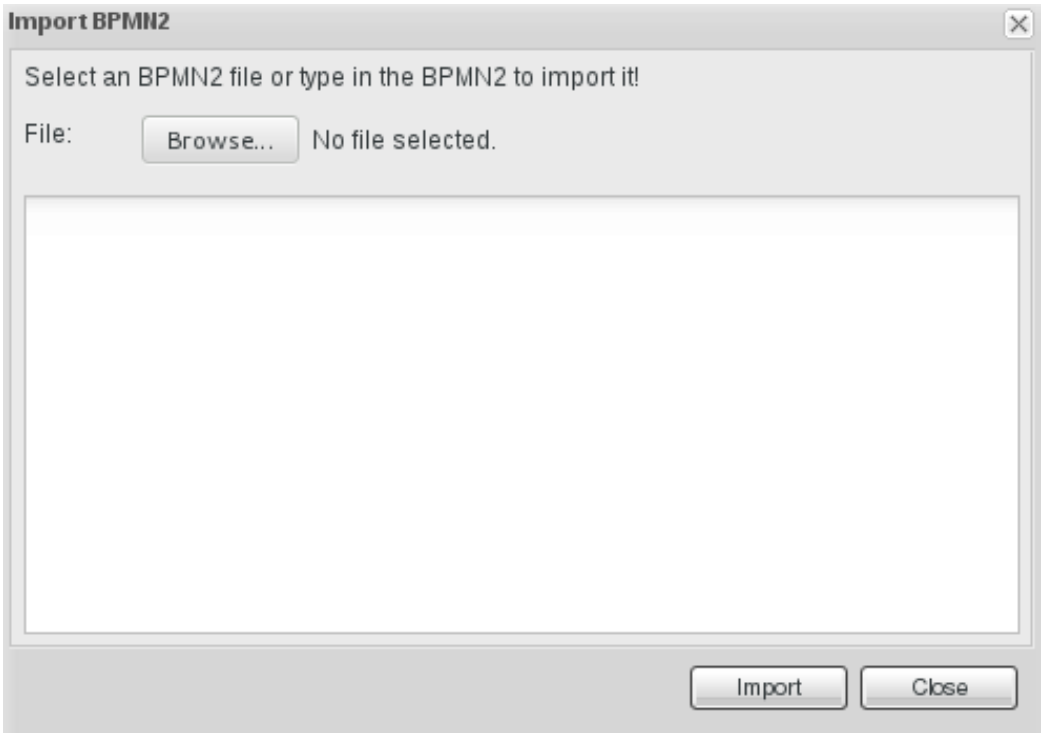


Figure 7.1. Fénêtre d'importation

Quand une définition de processus est importée, la définition importée existante est remplacée. Veillez à ne pas remplacer une définition de processus que vous avez modifiée de façon à ne perdre aucun changement.

Un processus peut aussi être importé dans le référentiel git du système de fichiers en clonant le référentiel, en ajoutant les fichiers de processus et en poussant les changements dans le git. De plus, pour les autres méthodes d'importation, vous pouvez copier et coller un processus ou simplement ouvrir un fichier dans une boîte de dialogue d'importation.

Quand on importe des processus, le Process Designer fournit un support visuel aux éléments de processus et a donc besoin d'informations sur les positions de l'élément sur le canvas. Si l'information n'est pas fournie dans le processus d'importation, vous devrez l'ajouter manuellement.

7.3. IMPORTER JPDL 3.2 DANS BPMN2

Pour migrer et importer une définition jPDL dans BPMN2, dans le Process Designer, cliquer sur le bouton d'importation, puis naviguer vers le bas et sélectionner **`Migrer jPDL 3.2 vers BPMN2`**.

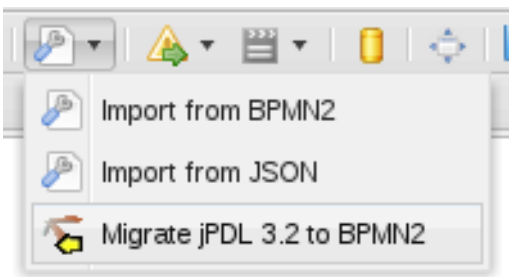


Figure 7.2. Migrer jPDL 3.2 dans BPMN2

Dans la boîte de dialogue **`Migrer vers BPMN2`**, sélectionner le fichier de définition de processus et le nom du fichier **gpd**. Confirmer en cliquant sur le bouton **`Migrer`**.

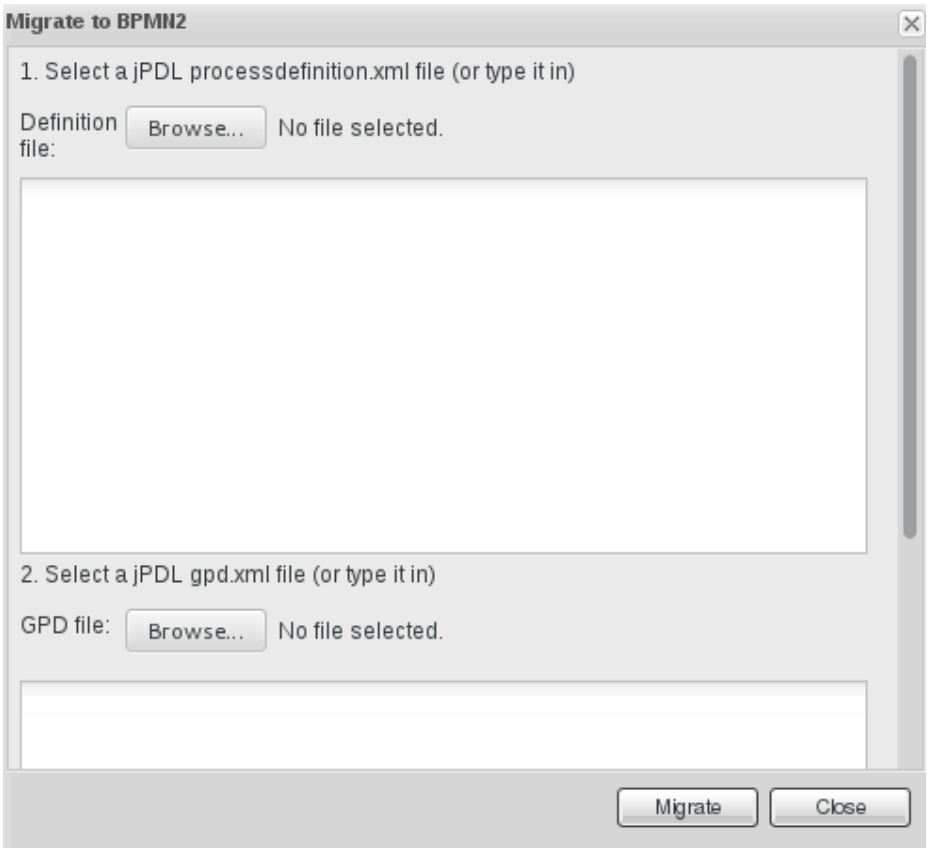
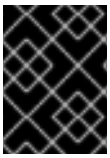


Figure 7.3. Migrer dans la boîte de dialogue BPMN2



IMPORTANT

L'outil de migration de jPDL 3.2 à BPMN2 est une fonctionnalité en aperçu technique, et n'est donc pas prise en charge par le Red Hat JBoss BPM Suite.

7.4. EXPORTATION D'UN PROCESSUS

Procédure 7.1. Exportation d'un processus métier

Pour exporter un processus métier, procéder ainsi :

1. Ouvrir la perspective **`Création Projet`** : à partir du menu principal, cliquer sur **Création** → **Création Projet**.
2. Sélectionner le processus métier à exporter pour le voir dans le Process Designer.
3. Cliquer sur le bouton () de la barre de conception de processus et sélectionner **`Afficher les sources des processus`** parmi les options de menu déroulant.
4. La fenêtre **`Sources des processus`** apparaît
5. Cliquer sur le bouton **`Télécharger BPMN2`** et sauvegarder le processus métier à l'emplacement désiré.

PARTIE III. INTÉGRATION

CHAPITRE 8. LE DÉPLOIEMENT D'ARTEFACTS DE RED HAT JBOSS BPM SUITE DANS S-RAMP (DE L'ANGLAIS REPOSITORY ARTIFACT MODEL AND PROTOCOL)

Alors que Red Hat JBoss BPM Suite et S-RAMP sont deux produits indépendants, il est possible de déplacer des artefacts entre les deux. Vous pouvez déplacer des artefacts de JBoss BPM Suite à S-RAMP en utilisant Maven ou via une interface utilisateur.

Cette section donne des informations sur ces deux processus.

8.1. LE DÉPLOIEMENT D'ARTEFACTS DE RED HAT JBOSS BPM SUITE DANS SOA REPOSITORY ARTIFACT MODEL AND PROTOCOL (S-RAMP) EN UTILISANT MAVEN

Avant que vous puissiez déployer des artefacts de Red Hat JBoss BPM Suite dans S-RAMP, vous devrez activer S-RAMP Maven Wagon. Maven Wagon est une fonctionnalité clé qui prend en charge le protocole API REST basé S-RAMP Atom. En activant S-RAMP Maven Wagon, les utilisateurs seront en mesure d'accéder aux artefacts du référentiel S-RAMP de dépendances dans un projet Maven.

Activer S-RAMP Maven Wagon en faisant une modification dans le fichier **`pom.xml`** comme dans l'exemple ci-dessous :

```
<build>
  <extensions>
    <extension>
      <groupId>org.overlord.sramp</groupId>
      <artifactId>s-ramp-wagon</artifactId>
      <version>${s-ramp-wagon.version}</version>
    </extension>
  </extensions>
</build>
```

Une fois que S-RAMP Maven Wagon est activé, vous pouvez déployer les artefacts de JBoss BPM Suite dans le référentiel S-RAMP. Pour ce faire, procédez aux étapes suivantes :

1. Cloner le référentiel git où vous avez sauvegardé le projet BPM Suite en exécutant cette commande :

   ```
   git clone http://localhost:8001/REPOSITORY_NAME
   ```

2. Par la ligne de commande, déplacez-vous dans le dossier qui contient le projet.

3. Suivez les instructions qui se trouvent dans *Red Hat JBoss Fuse Service Works 6 Development Guide, Volume 3: Governance*, section Deploiement dans S-RAMP. Utiliser l'URL de l'exemple ci-dessous :

   ```
   <distributionManagement>
     <repository>
       <id>local-sramp-repo</id>
       <name>S-RAMP Releases Repository</name>
       <url>sramp://S-RAMP_SERVER_URL/s-ramp-server/</url>
     </repository>
     <snapshotRepository>
   ```

```
    <id>local-sramp-repo-snapshots</id>
    <name>S-RAMP Snapshots Repository</name>
    <url>sramp://S-RAMP_SERVER_URL/s-ramp-server/</url>
  </snapshotRepository>
</distributionManagement>
```

Avec ces paramètres de configuration, les déploiements de Maven sont envoyés directement dans le référentiel S-RAMP en utilisant l'API S-RAMP. Notez que les artefacts sont ajoutés au référentiel S-RAMP avec un type d'artefact basé sur le type Maven du projet. Vous pouvez substituer ce comportement en ajoutant un paramètre de recherche dans l'URL du référentiel dans le fichier **pom.xml** file. Par exemple :

```
<distributionManagement>
  <repository>
    <id>local-sramp-repo</id>
    <name>S-RAMP Releases Repository</name>
    <url>sramp://S-RAMP_SERVER_URL/s-ramp-server/?
artifactType=KieJarArchive</url>
  </repository>
</distributionManagement>
```

L'exemple ci-dessus entraîne l'artefact Maven à être téléchargé avec un type d'artefact S-RAMP de KieJarArchive.

4. Modifier le plug-in de Maven dans le fichier **pom.xml** et ajouter y une dépendance comme suit au cas où le projet ne contienne pas de tables de décisions :

```
<plugins>
    <plugin>
      <groupId>org.kie</groupId>
      <artifactId>kie-maven-plugin</artifactId>
      <version>6.0.2-redhat-6</version>
      <extensions>true</extensions>
      <dependencies>
          <dependency>
              <groupId>org.jbpm</groupId>
              <artifactId>jbpm-bpmn2</artifactId>
              <version>6.0.2-redhat-6</version>
          </dependency>
      </dependencies>
     </plugin>
    </plugins>
```

Si le projet contient des tables de décisions, utiliser cette dépendance pour kie-maven-plugin à la place :

```
<plugins>
    <plugin>
      <groupId>org.kie</groupId>
      <artifactId>kie-maven-plugin</artifactId>
      <version>6.0.2-redhat-6</version>
      <extensions>true</extensions>
      <dependencies>
          <dependency>
               <groupId>org.drools</groupId>
```

```
            <artifactId>drools-decisiontables</artifactId>
            <version>6.0.2-redhat-6</version>
        </dependency>
    </dependencies>
  </plugin>
 </plugins>
```

5. Exécuter un déploiement propre par la commande suivante :

```
mvn -s sramp-settings.xml deploy
```

.

NOTE

Pour le déploiement Maven dans le référentiel de S-RAMP, il vous faut disposer d'informations d'identification définies dans le fichier **`settings.xml`**. Pour plus de détails sur les informations d'identification, consulter la documentation *Red Hat JBoss Fuse Service Works* (FSW) sur l'authentification.

8.2. LE DÉPLOIEMENT D'ARTEFACTS DE RED HAT JBOSS BPM SUITE DANS SOA REPOSITORY ARTIFACT MODEL AND PROTOCOL (S-RAMP) EN UTILISANT L'INTERFACE GRAPHIQUE (GUI).

Pour déployer les artefacts de Red Hat JBoss BPM Suite dans un référentiel S-RAMP utilisant l'interface utilisateur, procéder ainsi :

1. Naviguer dans http://localhost:8080/s-ramp-ui/ via navigateur web. Si l'interface utilisateur est configurée pour exécuter à partir d'un nom de domaine, substituer **`localhost`** par le nom de domaine. Par exemple http://www.example.com:8080/s-ramp-ui/.

2. Cliquer sur **`Artefacts`**.

3. Dans la section **`Gestion Artefacts`**, sélectionner le bouton **`Importer`**.

4. Situer l'archive kie que vous souhaitez déployer. Dans la boite de dialogue qui s'ouvre, remplir **`KieJarArchive`** comme type, et sélectionner **`Importer`**.

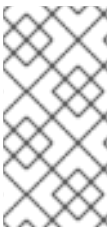
5. Le déploiement crée alors ces entrées dans un référentiel S-RAMP :

   `KieJarArchive`, à partir duquel on a :

   - **`KieXmlDocument`** (si l'archive contient **`kmodule.xml`**)
   - **`BpmnDocument`** (si l'archive contient les définitions **`bpmn`**)
   - **`DroolsDocument`** (si l'archive contient les définitions **`drl`**)

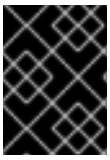
CHAPITRE 9. INTÉGRER RED HAT JBOSS BPM SUITE DANS RED HAT JBOSS FUSE

L'intégration de Red Hat JBoss Fuse permet aux utilisateurs de JBoss Fuse de complémenter leur solution d'intégration avec les fonctionnalités supplémentaires fournies par JBoss BPM et JBoss BRMS. L'intégration de Red Hat JBoss BPM Suite est assurée par deux fichiers `features.xml` : un fichier fournissant des fonctionnalités de base de JBoss BPM Suite et de JBoss BRMS, qui définissent les fonctionnalités d'OSGi pouvant être déployées dans JBoss Fuse et un autre fichier fournissant un support supplémentaires à l'intégration à SwitchYard et Camel.



NOTE

Pour les utilisateurs de JBoss FUse 6.1, les fonctionnalités de base uniquement de JBoss BPM et de JBoss BRMS fournies dans le fichier de fonctionnalités suivant sont prises en charge. Les clients qui utilisent une ancienne version de ce fichier doivent le mettre à jour.



IMPORTANT

L'intégration SwitchYard est un aperçu technologique de JBoss Fuse 6.2, et n'est donc pas actuellement prise en charge.

Les fonctionnalités de base de JBoss BPM Suite et de JBoss BRMS sont fournies par **`drools-karaf-features-6.2.0.Final-redhat-6-BZ1232486-features.xml`**:

- drools-common
- drools-module
- drools-templates
- drools-decisiontable
- drools-jpa
- kie
- kie-ci
- kie-spring
- kie-aries-blueprint
- jbpm-commons
- jbpm-human-task
- jbpm
- droolsjbpm-hibernate
- h2

Ce fichier de fonctionnalités (et ses référentiels dépendants) est actuellement fourni sous forme de correctif dans Red Hat Customer Portal .

Le tableau suivant donne un exemple de cas d'utilisation de certaines fonctionnalités listées ci-dessus.

Tableau 9.1. Exemples de fonctionnalités et de cas d'utilisation

Fonctionnalité	Cas d'utilisation
`drools-module`	Utiliser le moteur JBoss BRMS pour l'évaluation de règles, sans avoir besoin de tables de décisions, processus ou persistances.
`drools-jpa`	Utiliser le moteur de JBoss BRMS pour l'évaluation de règles de persistance ou de transactions, sans exiger de tables de décision ou de processus. La fonctionnalité **`drools-jpa`** contient déjà **`drools-module`**, mais vous devrez peut-être également installer la fonctionnalité **`droolsjbpm-hibernate`**, ou vous assurer qu'il y ait un package Hibernate compatible installé.
`drools-decisiontable`	Utiliser le moteur JBoss BRMS avec des tables de décision.
`jbpm`	Utiliser JBoss BPM Suite (ou le moteur JBoss BRMS avec les processus). La fonctionnalité **`jbpm`** contient déjà **`drools-module`**, et **`drools-jpa`**. Vous devrez peut-être également installer la fonctionnalité **`droolsjbpm-hibernate`**, ou vous assurer qu'il y ait un package Hibernate compatible installé.
`jbpm` et **`jbpm-human-task`**	Utiliser JBoss BPM Suite (ou le moteur JBoss BRMS avec des processus) avec Human Task.
Jars de moteurs de base et **`kie-ci`**.	Utiliser JBoss BRMS ou JBoss BPM Suite avec KieScanner (KIE-CI) pour télécharger les kJARs d'un référentiel Maven.
`kie-spring`	Utiliser l'intégration KIE-Spring.
`kie-spring` et **`kie-aries-blueprint`**.	Utiliser l'intégration KIE-Aries-Blueprint.

Les fonctionnalités supplémentaires suivantes pour l'intégration de SwitchYard et Camel dans JBoss Fuse sont fournies par **`org/jboss/integration/fuse/karaf-features/1.0.0.redhat-620137/karaf-features-1.0.0.redhat-620137-features.xml`**:

- fuse-bxms-switchyard-common-knowledge
- fuse-bxms-switchyard-rules
- fuse-bxms-switchyard-bpm
- kie-camel

- jbpm-workitems-camel

Ce fichier (et les référentiels dépendants) se situe dans http://repository.jboss.org/nexus/content/repositories/public, qui est déjà configuré pour être utilisé dans JBoss Fuse 6.2 out of the box dans ***`installDir`*`/etc/org.ops4j.pax.url.mvn.cfg`**.

Le fichier peut également être téléchargé de la page produits de JBoss Fuse 6.2 ou JBoss BPM Suite à partir du portail clients de Red Hat.

9.1. INSTALLER / METTRE À JOUR LES FONCTIONNALITÉS D'INTÉGRATION PRINCIPALES

Si vous avez déjà installé une ancienne version des fonctionnalités de base de JBoss BPM Suite et de JBoss BRMS (par exemple, **`drools-karaf-features-6.2.0.Final-redhat-6-features.xml`**), vous devrez les supprimer, ainsi que tous les fichiers associés avant d'installer le fichier **`features.xml`** plus récent.

Procédure 9.1. Supprimer une installation `drools-karaf-features` existante

1. Démarrer la console Fuse par :

```
$ ./installDir/bin/fuse
```

2. Retirer l'installation d'anciennes fonctionnalités/apps qui utilisaient l'ancien fichier **`features.xml`**. Par exemple :

```
JBossFuse:karaf@root> features:uninstall drools-module
JBossFuse:karaf@root> features:uninstall jbpm
JBossFuse:karaf@root> features:uninstall kie-ci
```

3. Chercher des références de lots en utilisant drools/kie/jbpm et les supprimer :

```
list -t 0 -s | grep drools
list -t 0 -s | grep kie
list -t 0 -s | grep jbpm
```

Pour supprimer les lots :

```
karaf@root> osgi:uninstall <BUNDLE_ID>
```

4. Supprimer l'ancien url drools-karaf-features :

```
karaf@root> features:removeurl mvn:org.drools/drools-karaf-
features/6.2.0.Final-redhat-<VERSION>/xml/features
```

5. Redémarrer Fuse

6. Ajouter le fichier de nouvelles fonctionnalités :

```
karaf@root> features:addurl mvn:org.drools/drools-karaf-
features/6.2.0.Final-redhat-6-BZ1232486/xml/features
```

7. Installer les fonctionnalités :

```
karaf@root> features:install ...
```

Pour installer **drools-karaf-features** :

Procédure 9.2. Installer les fonctionnalités de base de JBoss BPM Suite et de JBoss BRMS

1. Télécharger et Installer le correctif.
   a. Télécharger **jboss-brms-6.1.1-BZ-1232486.zip**.
   b. Le décompresser.
   c. Naviguer dans le répertoire **BZ-1232486**.
   d. Exécuter la commande de console suivante :

```
$ mvn org.apache.maven.plugins:maven-install-
plugin:2.5.2:install-file -Dfile=drools-karaf-features-
6.2.0.Final-redhat-6-BZ1232486-features.xml -DgroupId=org.drools
-DartifactId=drools-karaf-features -Dversion=6.2.0.Final-redhat-
6-BZ1232486 -Dpackaging=xml -Dclassifier=features
```

2. Configurer les référentiels demandés
   a. Modifier le fichier ***installDir*/etc/org.ops4j.pax.url.mvn.cfg** dans votre installation JBoss Fuse et ajouter la variable **org.ops4j.pax.url.mvn.repositories**, et notez que les entrées sont séparées par ', \' :
      - http://maven.repository.redhat.com/techpreview/all/@id=bxms-product-repo
3. Démarrer JBoss Fuse :

```
$ ./installDir/bin/fuse
```

4. Ajouter un référence au fichier de fonctionnalités de base en exécutant la commande de console suivante :

```
JBossFuse:karaf@root> features:addurl mvn:org.drools/drools-karaf-
features/6.2.0.Final-redhat-6-BZ1232486/xml/features
```

5. Vous pouvez installer les fonctionnalités fournies dans ce fichier en exécutant, par exemple, la commande de console suivante :

```
JBossFuse:karaf@root> features:install drools-module
```

9.2. INSTALLER DES FONCTIONNALITÉS D'INTÉGRATION SUPPLÉMENTAIRES

Utiliser la procédure suivante pour une intégration supplémentaire avec SwitchYard et Camel.

IMPORTANT

L'intégration SwitchYard est un aperçu technologique de JBoss Fuse 6.2, et n'est donc pas actuellement prise en charge.

Procédure 9.3. Intégration SwitchYard et Camel

1. Ajouter une référence au fichier de fonctionnalités en vue d'une intégration supplémentaire avec SwitchYard et Camel en exécutant la commande de console suivante :

```
JBossFuse:karaf@root> features:addurl
mvn:org.jboss.integration.fuse/karaf-features/1.0.0.redhat-
620137/xml/features
```

2. Vous pouvez installer les fonctionnalités fournies pour l'intégration SwitchYard and Camel en exécutant, par exemple, la commande de console suivante :

```
JBossFuse:karaf@root> features:install fuse-bxms-switchyard-rules
```

9.3. INSTALLER LES APPLICATIONS QUICKSTART DEJBOSS FUSE INTEGRATION

Les fonctionnalités suivantes des applications quickstart de JBoss Fuse Integration sont fournies par **`org/jboss/integration/fuse/quickstarts/karaf-features/1.0.0.redhat-620137/karaf-features-1.0.0.redhat-620137-features.xml`**:

- fuse-bxms-switchyard-quickstart-bpm-service
- fuse-bxms-switchyard-quickstart-remote-invoker
- fuse-bxms-switchyard-quickstart-rules-camel-cbr
- fuse-bxms-switchyard-quickstart-rules-interview
- fuse-bxms-switchyard-quickstart-rules-interview-container
- fuse-bxms-switchyard-quickstart-rules-interview-dtable
- fuse-bxms-switchyard-demo-library
- fuse-bxms-switchyard-demo-helpdesk

Ce fichier et les référentiels dépendants se situent dans http://repository.jboss.org/nexus/content/repositories/public, qui est déjà configuré pour être utilisé dans JBoss Fuse 6.2 out of the box dans **`installDir/etc/org.ops4j.pax.url.mvn.cfg`**.

Procédure 9.4. Installer l'application Quickstart

1. Ajouter une référence au fichier de fonctionnalités en exécutant la commande de console suivante :

```
JBossFuse:karaf@root> features:addurl
mvn:org.jboss.integration.fuse.quickstarts/karaf-
features/1.0.0.redhat-620137/xml/features
```

2. Vous pouvez maintenant installer les applications quickstart fournies par de fichier de fonctionnalités en exécutant, par exemple la commande de console suivante :

```
JBossFuse:karaf@root> features:install fuse-bxms-switchyard-
quickstart-bpm-service
```

Vous pouvez aussi télécharger un fichier ZIP à partir de la page de connaissance produits : https://repository.jboss.org/nexus/content/repositories/public/org/jboss/integration/fuse/fuse-integration-karaf-distro/1.0.0.redhat-620137/ . Le fichier fournit le code source de chaque application quickstart, ainsi que le code pour tester.

Procédure 9.5. Télécharger et installer les fichiers ZIP de Quickstart

1. Télécharger le fichier ZIP d'installation Quickstart.
2. Extraire les éléments de contenu du répertoire quickstart dans votre répertoire ***installDir*/quickstart** existant.
3. Extraire les éléments de contenu du répertoire de système dans votre répertoire ***installDir*/system** existant.

9.3.1. Tester votre première application Quickstart

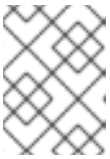
Procédure 9.6. Tester l'application Quickstart

1. Démarrer JBoss Fuse :

```
$ ./installDir/bin/fuse
```

2. Installer et démarrer **switchyard-bpm-service** en exécutant la commande de console suivante :

```
JBossFuse:karaf@root> features:install fuse-bxms-switchyard-
quickstart-bpm-service
```

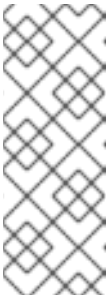


NOTE

Toutes les fonctionnalités dépendantes spécifiées dans le fichier des fonctionnalités de l'application seront installées automatiquement.

3. Soumettre une requête web pour invoquer la passerelle SOAP.
   a. Ouvrir une fenêtre de terminal, et naviguez dans le répertoire quickstart associé extrait du fichier ZIP de l'application quickstart (dans ce cas, **switchyard-bpm-service**).
   b. Exécuter la commande suivante :

```
$ mvn clean install
```



NOTE

Vous aurez besoin des référentiels suivants configurés dans votre fichier **`settings.xml`** :

- http://maven.repository.redhat.com/techpreview/all/
- http://repository.jboss.org/nexus/content/repositories/public/

c. Exécuter la commande suivante :

```
$ mvn exec:java -Pkaraf
```

4. Vous recevrez la réponse suivante :

```
SOAP Reply:
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"><SOAP-
ENV:Header xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"/><soap:Body><ns2:sub
mitOrderResponse xmlns:ns2="urn:switchyard-quickstart:bpm-
service:1.0">
    <orderId>test1</orderId>
    <accepted>true</accepted>
    <status>Thanks for your order, it has been shipped!</status>
</ns2:submitOrderResponse></soap:Body></soap:Envelope>
```

CHAPITRE 10. INTÉGRATION AVEC SPRING

10.1. CONFIGURER RED HAT JBOSS BPM SUITE AVEC SPRING

Consulter le Guide d'installation de Red Hat JBoss BPM Suite pour télécharger le module Spring. Vous aurez besoin de télécharger la version déployable de JBoss BPM Suite.

Le module de Spring se trouve dans le fichier **`jboss-bpms-engine.zip`** et s'appelle **`kie-spring-VERSION-redhat-MINORVERSION.jar`**.

Ce que vous souhaitez faire avec les modules Spring affecte la façon dont vous aurez besoin de les configurer.

En tant que Moteur de processus auto gérés

C'est la façon habituelle de commencer à utiliser Suite BPM JBoss dans votre application Spring. Vous la pré-configurez une seule fois et vous l'exécutez dans le cadre de l'application. Grâce à l'API **`RuntimeManager`**, une synchronisation parfaite entre le service de tâches et le moteur de processus est gérée en interne et l'utilisateur final n'a pas à traiter le code interne pour les faire fonctionner ensemble.

En tant que service de tâches partagées

Lorsque vous utilisez une instance unique d'un **`TaskService`**, vous avez plus de flexibilité dans la configuration de l'instance de service de tâches car elle est indépendante du **`RuntimeManager`**. Une fois configurée, elle est ensuite utilisée par le RuntimeManager à la demande.

Pour créer un RuntimeEnvironment partir de votre application Spring, vous pouvez utiliser la classe **`org.kie.spring.factorybeans.RuntimeEnvironmentFactoryBean`**. Cette classe de fabrique est chargée de produire des instances de **`RuntimeEnvironment`** qui sont consommés par le RuntimeManager lors de la création. Illustré ci-dessous, vous verrez un RuntimeEnvironment configuré le gestionnaire d'entités, le gestionnaire de transactions et les ressources pour la classe **`org.kie.spring.factorybeans.RuntimeEnvironmentFactoryBean`** :

```
<bean id="runtimeEnvironment"
class="org.kie.spring.factorybeans.RuntimeEnvironmentFactoryBean">
  <property name="type" value="DEFAULT"/>
  <property name="entityManagerFactory" ref="jbpmEMF"/>
  <property name="transactionManager" ref="jbpmTxManager"/>
  <property name="assets">
    <map>
      <entry key-ref="process"><util:constant static-
field="org.kie.api.io.ResourceType.BPMN2"/></entry>
    </map>
  </property>
</bean>
```

L'environnement de runtime suivant peut être créé ou configuré :

- DEFAULT - configuration par défaut (le plus commun) du RuntimeManager
- EMPTY - environnement complètement vide à remplir manuellement
- DEFAULT_IN_MEMORY - la même chose que DEFAULT mais sans persistence du moteur de runtime

- DEFAULT_KJAR - la même chose que DEFAULT mais les ressources de connaissance sont prises dans le KJAR identifié par releaseId ou le GAV
- DEFAULT_KJAR_CL - généré directement à partir d'une chemin de classe qui consiste en un descripteur kmodule.xml

Suivant le type sélectionné, plusieurs propriétés obligatoires sont requises. Cependant, une des propriétés de connaissance suivantes doit être fournie :

- knowledgeBase
- assets
- releaseId
- groupId, artifactId, version

Finalement, pour les types DEFAULT, DEFAULT_KJAR, DEFAULT_KJAR_CL, la persistance doit être configurée sous la forme de valeurs **`entity manager factory`** ou**`transaction manager`**. Voici ci-dessous un exemple de RuntimeManager pour **`org.kie.spring.factorybeans.RuntimeManagerFactoryBean`**:

```
<bean id="runtimeManager"
class="org.kie.spring.factorybeans.RuntimeManagerFactoryBean" destroy-
method="close">
  <property name="identifier" value="spring-rm"/>
  <property name="runtimeEnvironment" ref="runtimeEnvironment"/>
</bean>
```

CHAPITRE 11. INTÉGRATION CDI

11.1. INTÉGRATION CDI

Pour utiliser jbpm-kie-services dans votre système, vous devrez fournir certains mbeans qui puissent satisfaire toutes les dépendances des services. Il y a plusieurs mbeans qui dépendent de certains scénarios.

- Gestionnaire d'entités et Usine de gestionnaires d'entités
- Callback de groupes d'utilisateurs pour des tâches humaines
- Fournisseur d'identité pour passer les informations d'authentification de l'utilisateur aux services.

Quand vous exécutez un environnement JEE, comme JBoss Application Server, le mbean doit remplir toutes les exigences de jbpm-kie-services

```
public class EnvironmentProducer {

    @PersistenceUnit(unitName = "org.jbpm.domain")
    private EntityManagerFactory emf;

    @Inject
    @Selectable
    private UserGroupCallback userGroupCallback;

    @Produces
    public EntityManagerFactory getEntityManagerFactory() {
        return this.emf;
    }

    @Produces
    @RequestScoped
    public EntityManager getEntityManager() {
        EntityManager em = emf.createEntityManager();
        return em;
    }

    public void close(@Disposes EntityManager em) {
        em.close();
    }

    @Produces
    public UserGroupCallback produceSelectedUserGroupCalback() {
        return userGroupCallback;
    }
    @Produces

    public IdentityProvider produceIdentityProvider {
        return new IdentityProvider() {
             // implement IdentityProvider
        };
    }
}
```

Alors le fichier **deployments/business-central.war/WEB-INF/beans.xml** peut être configuré pour pouvoir modifier les paramètres de configuration actuels de l'implémentation de **usergroupcallback**.

```
<beans xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://docs.jboss.org/cdi/beans_1_0.xsd">

<alternatives>
  <class>org.jbpm.services.task.identity.JAASUserGroupCallbackImpl</class>
</alternatives>

</beans>
```

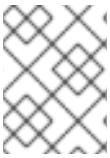
NOTE

org.jbpm.services.task.identity.JAASUserGroupCallbackImpl est juste un exemple pour démontrer les paramètres de configuration du serveur d'applications quel qu'il fût (LDAP, DB, etc).

CHAPITRE 12. PERSISTANCE

Les données de runtime du Process Engine peuvent être persistées dans les stores de données. Le mécanisme de persistance sauvegarde les données par le marshalling : les données de runtime sont converties en jeu de données et le jeu de données est sauvegardé dans le stockage de données.

Notez que la persitance n'est pas configurée par défaut et le moteur exécute sans persitance.



NOTE

Les données de runtime sont sauvegardées en utilisant le marshalling (persistance binaire). Le mécanisme de marshalling est un mécanisme de sérialisation personnalisé.

Red Hat JBoss BPM Suite persitera ce qui suit quand la persistance sera configurée :

- État de session : cela inclut l'ID de session, la date de la dernière modification, les données de session dont les règles métier peuvent avoir besoin pour l'évaluation, l'état des tâches de minuterie.
- État d'instance de processus : cela inclut l'ID d'instance de processus, l'ID de processus, la date de la dernière modification, la date du dernier accès en lecture, la date de démarrage de l'instance de processus, les données de runtime (le statut d'exécution comprenant le noeud à exécuter, les valeurs des variables, etc.) et les types d'événements.
- État de runtime d'élément de travail : inclut l'ID d'élément de travail, la date de création, le nom, l'ID d'instance de processus, et l'état de l'élément de travail lui-même.

Sur la base des données persistées, il est possible de restaurer l'état d'exécution de toutes les instances exécutant en cas d'échec ou pour supprimer les instances en cours d'éxécution de façon temporaire en les restaurant plus tard. Par défaut, aucune persitance n'est configurée.

Pour autoriser la persistance, vous devez ajouter les fichiers jar de jbpm-persistance au CLASSPATH de votre application et configurer le moteur pour pouvoir utiliser la persistance. Le moteur enregistre automatiquement l'état d'exécution dans le stockage lorsque le moteur atteint un point sûr. Les points sécurisé sont des points où l'instance de processus est suspendue. Lorsqu'une invocation d'instance de processus atteint un point sécurisé dans le moteur, le moteur stocke toutes les modifications à l'instance de processus comme un instantané des données de runtime du processus. Toutefois, lorsqu'une instance de processus est terminée, l'instantané persisté des données de runtime de l'instance de processus est automatiquement supprimé.

Si une défaillance a lieu et que vous avez besoin de restaurer le runtime du moteur à partir du stockage, les instances de processus seront automatiquement restaurées et leur exécution continuera, donc vous n'aurez pas besoin de recharger ou de déclencher les instances de processus manuellement.

Les données de persistance de runtime doivent être considérées comme étant internes au moteur. Vous ne devez pas accéder à des données de runtime persitées ou les modifier directement car cela pourrait avoir des effets inattendus.

Pour obtenir des informations sur l'état actuel d'exécution, voir le journal de l'historique. Interroger la base de données de données de temps d'exécution que si c'est absolument nécessaire.

12.1. SESSION

Les sessions sont persistées comme entités **`SessionInfo`**. Elles persistent l'état de la session KIE de runtime, et stocke les données suivantes :

Tableau 12.1.

Champ	Description	Nullable
id	Clé primaire	false
lastmodificationdate	dernière sauvegarde dans le store de données	N/A
rulesbytearray	jeu de données binaire avec état de session (blob binaire	false
startdate	démarrage de session	
optlock	numéro de version utilisé pour verrouiller la valeur d'un verrouillage optimiste	

12.2. INSTANCE DE PROCESSUS

Les instances de processus sont persistées en tant qu'entités **`ProcessInstanceInfo`**, qui persistent l'état d'une instance de processus en cours d'exécution et qui stockent les données suivantes :

Tableau 12.2.

Champ	Description	Nullable
instanceid	Clé primaire	false
lastmodificationdate	dernière sauvegarde dans le store de données	N/A
lastreaddate	dernière lecture dans le store de données	N/A
processid	ID du processus sur lequel l'instance est basée	false
processinstancebytearray	jeu de données binaire avec état d'instance de processus (blob binaire)	false
startdate	Date de démarrage de l'instance de processus	
optlock	numéro de version utilisée comme valeur de verrouillage pour le verrouillage optimiste	
état	État d'instance de processus	false

ProcessInstanceInfo possède une relation 1:N avec l'entité **EventTypes**.

L'entité **EventTypes** contient les données suivantes :

Tableau 12.3.

Champ	Description	Nullable
instanceid	référence à l'instance de processus (clé étrangère de **processinstanceinfo**)	false
élément	champ de texte lié à un événement subi par une instance de processus	

Support du verrouillage pessimiste

Le mécanisme de vérrouillage de persistance des processus par défaut est *optimiste*. Avec une haute simultanéité multi-thread pour une même instance de processus, cette stratégie de verrouillage peut entraîner de mauvaises performances.

Avec la sortie de la version 6.1 de Red Hat JBoss BPM Suite, cela peut être modifié pendant l'exécution pour permettre à l'utilisateur de définir le verrouillage sur la base d'un processus à la fois, et lui permettre d'être *pessimiste* (la modification peut être faite au niveau session KIE ou Runtime Manager et pas seulement au niveau du processus).

Pour configurer un processus qui puisse utiliser le verrouillage pessimiste, procédez ainsi dans un environnement de runtime :

```
import org.kie.api.runtime.Environment;
import org.kie.api.runtime.EnvironmentName;
import org.kie.api.runtime.manager.RuntimeManager;
import org.kie.api.runtime.manager.RuntimeManagerFactory;

...

// here env is an instance of org.kie.api.runtime.Environment
env.set(EnvironmentName.USE_PESSIMISTIC_LOCKING, true);

// now create your Runtime Manager using this enviornment
RuntimeManager manager =
RuntimeManagerFactory.Factory.get().newPerRequestRuntimeManager(environmen
t);
```

12.3. ÉLÉMENT DE TRAVAIL

Les éléments de travail sont persités comme entités **workiteminfo**, qui persistent l'état d'une instance d'élément de travail en particulier en cours d'exécution et qui stocke les données suivantes :

Tableau 12.4.

Champ	Description	Nullable
workitemid	Clé primaire	false
nom	nom de l'élément de travail	
processinstanceid	id d'instance de processus parent	false
état	entier relatif représentant un état d'élément de travail	false
optlock	numéro de version utilisée comme valeur de verrouillage pour le verrouillage optimiste	
workitembytearray	ensemble de données binaires avec état d'élément de travail (blob binaire)	false
creationDate	horodate de création de l'élément de travail	false

12.4. CONFIGURATION DE LA PERSISTANCE

12.4.1. Configuration de la persistance

Bien que la persistance ne soit pas utilisée par défaut, les dépendances nécessaires sont disponibles dans le répertoire d'exécution sous forme de fichiers jar .

La peristance est définie par session, et vous pouvez la définir en utilisant la classe **`JBPMHelper`** quand vous créez une session ou en utilisant **`JPAKnowledgeService`** pour créer votre session. La dernière option offre plus de flexibilité, tandis que **`JBPMHelper`** a une méthode pour créer une session et utilise un fichier de configuration pour configurer cette session.

12.4.2. Configurer une persistance en utilisant JBPMHelper

Pour configurer une persistance de votre session en utilisant JBPMHelper, procéder ainsi :

1. Définir votre application pour qu'elle puisse utiliser le contructeur de session JBPMHelper :
   - **`KieSession ksession = JBPMHelper.newKieSession(kbase);`**
   - **`KieSession ksession = JBPMHelper.loadKieSession(kbase, sessionId);`**
2. Configurer la persistance dans le fichier **`jBPM.properties`** .

> **Exemple 12.1. Extrait de fichier jBPM.properties avec une persistance pour la base de données H2 en-mémoire**
>
> ```
> # for creating a datasource
> persistence.datasource.name=jdbc/jbpm-ds
> persistence.datasource.user=sa
> ```

```
persistence.datasource.password=
persistence.datasource.url=jdbc:h2:tcp://localhost/~/jbpm-db
persistence.datasource.driverClassName=org.h2.Driver

# for configuring persistence of the session
persistence.enabled=true
persistence.persistenceunit.name=org.jbpm.persistence.jpa
persistence.persistenceunit.dialect=org.hibernate.dialect.H2Dialec
t

# for configuring the human task service
taskservice.enabled=true
taskservice.datasource.name=org.jbpm.task
taskservice.transport=mina
taskservice.usergroupcallback=org.jbpm.task.service.DefaultUserGro
upCallbackImpl
```

Toutes les invocations sur la session vont déclencher le processus de persistance.

Veillez à ce que la source de données fonctionne bien au démarrage du moteur. Si vous exécutez la base de données H2 en-mémoire, vous pouvez démarrer la base de données avec votre application par la méthode **`JBPMHelper.startH2Server();`** et l'enregistrer dans engine par l'appel de méthode **`JBPMHelper.setupDataSource();`**.

12.4.3. Configurer une persistance en utilisant JPAKnowledgeService

Pour créer votre session de connaissance et configurer sa persistance avec JPAKnowledgeService, procéder ainsi :

1. Définir votre application pour qu'elle utilise la session de connaissance créée par JPAKnowledgeService :
    - Définir la session sur la base d'une base de connaisances, une configuration de session de connaissance, et un environnement. L'environnement doit contenir une référence à l'usine de gestionnaires d'entités :

```
// create the entity manager factory and register it in the
environment
EntityManagerFactory emf =
Persistence.createEntityManagerFactory(
"org.jbpm.persistence.jpa" );
Environment env = KnowledgeBaseFactory.newEnvironment();
env.set( EnvironmentName.ENTITY_MANAGER_FACTORY, emf );

// create a new knowledge session that uses JPA to store the
runtime state
KieSession ksession = JPAKnowledgeService.newKieSession( kbase,
null, env );
int sessionId = ksession.getId();

// invoke methods on your method here
ksession.startProcess( "MyProcess" );
ksession.dispose();
```

- Définir la session sur la base d'un id de session spécifique

```
// recreate the session from database using the sessionId
ksession = JPAKnowledgeService.loadKieSession(sessionId, kbase,
null, env );
```

2. Configurer la persistance dans le fichier **`META-INF/persistence.xml`** : configurer JPA pour qu'il utilise Hibernate et la base de données respective.

   Les informations sur la façon de configurer la source de données sur votre serveur d'applications doivent être dans la documentation livrée avec le serveur d'applications. Pour cette information pour JBoss Enterprise Application Platform, consulter le *Guide de Configuration et d'Administration* pour ce produit.

Exemple 12.2. Extrait de fichier persistence.xml avec une persistance pour une source de données H2 `jdbc/jbpm-ds`

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<persistence
  version="1.0"
  xsi:schemaLocation=
    "http://java.sun.com/xml/ns/persistence
     http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd
     http://java.sun.com/xml/ns/persistence/orm
     http://java.sun.com/xml/ns/persistence/orm_1_0.xsd"
  xmlns:orm="http://java.sun.com/xml/ns/persistence/orm"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/persistence">
  <persistence-unit name="org.jbpm.persistence.jpa" transaction-
type="JTA">
    <provider>org.hibernate.ejb.HibernatePersistence</provider>
    <jta-data-source>jdbc/jbpm-ds</jta-data-source>
    <mapping-file>META-INF/JBPMorm.xml</mapping-file>
    <class>org.drools.persistence.info.SessionInfo</class>

<class>org.jbpm.persistence.processinstance.ProcessInstanceInfo</c
lass>
    <class>org.drools.persistence.info.WorkItemInfo</class>
    <properties>
      <property name="hibernate.dialect"
value="org.hibernate.dialect.H2Dialect"/>
      <property name="hibernate.max_fetch_depth" value="3"/>
      <property name="hibernate.hbm2ddl.auto" value="update"/>
      <property name="hibernate.show_sql" value="true"/>
      <property name="hibernate.transaction.manager_lookup_class"

value="org.hibernate.transaction.BTMTransactionManagerLookup"/>
    </properties>
  </persistence-unit>
</persistence>
```

Toutes les invocations sur la session vont déclencher le processus de persistance.

Veillez à ce que la source de données fonctionne bien au démarrage du moteur. Si vous exécutez la

base de données H2 en-mémoire, vous pouvez démarrer la base de données avec votre application par la méthode **`JBPMHelper.startH2Server();`** et l'enregistrer dans engine par l'appel de méthode **`JBPMHelper.setupDataSource();`**.

NOTE

Si vous exécutez JBoss BPM Suite dans un environnement Java simple, votre configuration de source de données ressemblera à ce qui suit :

```
PoolingDataSource ds = new PoolingDataSource();
ds.setUniqueName("jdbc/jbpm-ds");
ds.setClassName("bitronix.tm.resource.jdbc.lrc.LrcXADataSource"
);
ds.setMaxPoolSize(3);
ds.setAllowLocalTransactions(true);
ds.getDriverProperties().put("user", "sa");
ds.getDriverProperties().put("password", "sasa");
ds.getDriverProperties().put("URL",
"jdbc:h2:tcp://localhost/~/jbpm-db");
ds.getDriverProperties().put("driverClassName",
"org.h2.Driver");
ds.init();
```

CHAPITRE 13. TRANSACTIONS

13.1. TRANSACTIONS

Le moteur de processus supporte les transactions JTA : les transactions locales ne sont supportées que quand les transactions locales Spring Pure ne le sont pas.

Par défaut, chaque invocation de méthode est considérée comme une transaction. Pour changer ce comportement, par exemple, pour combiner plusieurs commandes en une transaction, vous devrez spécifier les limites de la transaction.

13.2. DÉFINIR DES TRANSACTIONS

Pour définir une transaction, procédéz ainsi :

1. Enregistrez le gestionnaire de transactions dans votre environnement.

Exemple 13.1. Codifiez par l'enregistrement du gestionnaire de transactions

```
// create the entity manager factory
EntityManagerFactory emf =
EntityManagerFactoryManager.get().getOrCreate("org.jbpm.persistenc
e.jpa");
TransactionManager tm =
TransactionManagerServices.getTransactionManager();
Environment env = EnvironmentFactory.newEnvironment();
env.set(EnvironmentName.ENTITY_MANAGER_FACTORY, emf);
env.set(EnvironmentName.TRANSACTION_MANAGER, tm);

// setup the runtime environment
RuntimeEnvironment environment =
RuntimeEnvironmentBuilder.Factory.get()
.newDefaultBuilder()
.addAsset(ResourceFactory.newClassPathResource("MyProcessDefinitio
n.bpmn2"), ResourceType.BPMN2)
    .addEnvironmentEntry(EnvironmentName.TRANSACTION_MANAGER, tm)

.addEnvironmentEntry(EnvironmentName.PERSISTENCE_CONTEXT_MANAGER,
new JpaProcessPersistenceContextManager(env))

.addEnvironmentEntry(EnvironmentName.TASK_PERSISTENCE_CONTEXT_MANA
GER, new JPATaskPersistenceContextManager(env))
    .get();
```

2. Initialiser la KieSession :

```
// get the KieSession
RuntimeManager manager =
RuntimeManagerFactory.Factory.get().newPerProcessInstanceRuntimeMana
ger(environment);
```

```
RuntimeEngine runtime =
manager.getRuntimeEngine(ProcessInstanceIdContext.get());
KieSession ksession = runtime.getKieSession();
```

3. Définir le gestionnaire de transactions dans **`jndi.properties`**.

Exemple 13.2. Definition du gestionnaire de transactions Bitronix dans jndi.properties

```
java.naming.factory.initial=bitronix.tm.jndi.BitronixInitialContex
tFactory
```

NOTE

Pour utiliser un gestionnaire de transactions différent, modifier hibernate.transaction.manager_lookup_class, la propriété de gestionnaire de transactions, dans le fichier **`persistence.xml`** pour charger votre gestionnaire de transactions.

Exemple 13.3. Le gestionnaire de transactions de JBoss comme gestionnaire de transactions

```
<property
name="hibernate.transaction.manager_lookup_class"
value="org.hibernate.transaction.JBossTransactionManage
rLookup"/>
```

4. Définir le début et la fin d'une transaction.

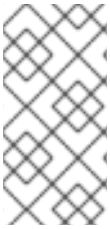
```
 // start the transaction
UserTransaction ut =
InitialContext.doLookup("java:comp/UserTransaction");
ut.begin();

// perform multiple commands inside one transaction
ksession.insert( new Person( "John Doe" ) );
ksession.startProcess("MyProcess");

// commit the transaction
ut.commit();
```

13.3. CMT (CONTAINER MANAGED TRANSACTIONS)

Dans les cas où la JBoss BPM Suite est incorporée dans une application qui est dans un conteneur qui permet de gérer les transactions lui-même (Container Managed Transactions - CMT), un gestionnaire de transactions spécialement dédié est fourni grâce à la classe **`org.jbpm.persistence.jta.ContainerManagerTransactionManager`**. C'est parce que l'implémentation par défaut du gestionnaire de transactions de JBoss BPM Suite est basée sur la

classe **`UserTransaction`** qui obtient le statut de la transaction. Cependant, certains serveurs d'applications en mode CMT ne permettent pas d'accéder à l'instance de **`UserTransaction`** à partir du JNDI.

Les opérations exécutées sur ce gestionnaire sont toutes non opérationnelles (NOOP) parce qu'elles ne peuvent pas affecter la CMT sous-jacent. La classe **`ContainerManagedTransactionManager`** s'attend à ce que la transaction soit toujours active (renvoyant **`ACTIVE`** à la méthode **`getStatus()`**).



NOTE

Même si le conteneur gère des transactions, le conteneur doit être informé de toutes les exceptions qui se produisent pendant l'exécution d'instances de processus. Les exceptions levées par le moteur doivent être propagées jusqu'au conteneur pour pouvoir restaurer correctement les transactions.

Configurer le gestionnaire de transactions

Pour pouvoir utiliser le **`ContainerManagedTransactionManager`**, il doit être inséré dans l'environnement avant que vous puissiez créer ou charger une session :

```
    Environment env = EnvironmentFactory.newEnvironment();
    env.set(EnvironmentName.ENTITY_MANAGER_FACTORY, emf);
    env.set(EnvironmentName.TRANSACTION_MANAGER, new
ContainerManagedTransactionManager());
    env.set(EnvironmentName.PERSISTENCE_CONTEXT_MANAGER, new
JpaProcessPersistenceContextManager(env));
```

Ensuite, installer votre fournisseur de service JPA dans votre fichier **`persistence.xml`**. Par exemple, si vous utilisez WebSphere d'IBM :

```
<property name="hibernate.transaction.factory_class"
value="org.hibernate.transaction.CMTTransactionFactory"/>
<property name="hibernate.transaction.manager_lookup_class"
value="org.hibernate.transaction.WebSphereExtendedJTATransactionLookup"/>
```

Disposer d'une KSession dans une CMT

Dans une CMT, vous ne devriez pas disposer d'une ksession directement (en utilisant la méthode **`dispose()`**). Ce faisant provoquerait des exceptions à la fin de la transaction car le moteur de processus doit nettoyer l'état quand l'invocation est terminée

Utilisez plutôt la méthode **`execute()`** de la classe spécialisée **`org.jbpm.persistence.jta.ContainerManagedTransactionDisposeCommand`**. Cette commande s'assure que la ksession sera supprimée lorsque la transaction sera effectivement terminée.

Cette méthode vérifie si la transaction est active. Si c'est le cas, elle délègue la phase **`afterDisposal`** de la transaction au lieu de l'exécuter directement. S'il n'y a aucune transaction active, la ksession sera supprimée immédiatement.

CHAPITRE 14. JOURNALISATION

Le mécanisme de journalisation vous permet de stocker des informations sur l'exécution d'une instance de processus. Il est fourni par un listener spécial qui écoute le moteur de processus pour être à l'affut de tout événement pertinent à journaliser, de façon à ce que l'information puisse être stockée séparémment de l'information non journlaisée dans une base de données intégrée (h2) ou dans une source de données connectée qui utilise JPA ou Hibernate.

Le module jbpm-audit fournit le listener de l'événement et vous permet également de stocker des informations liées aux processus directement dans une base de données en utilisant JPA ou Hibernate. Les données des entités suivantes sont stockées comme suit :

- Instance de processus **`processinstancelog`**
- Instance d'élément **`nodeinstancelog`**
- Instance de variable **`variableinstancelog`**

Tableau 14.1. Champs de la table de ProcessInstanceLog

Champ	Description	Nullable
id	La clé primaire de l'entité de journalisation	Non
end_date	La date de fin de l'instance du processus	Oui
processid	Le nom (id) du processus sous-jacent	Oui
processinstanceid	L'id de l'instance du processus	Non
start_date	La date de démarrage de l'instance du processus	Oui
status	Le statut de l'instance du processus	Oui
parentProcessInstanceId	L'id d'instance de processus de l'instance de processus parent si tel est le cas	Oui
résultat	Le résultat de l'instance de processus (comme des information de fin de processus, comme un code d'erreur)	Oui

Tableau 14.2. Champs de la table de NodeInstanceLog

Champ	Description	Nullable
id	La clé primaire de l'entité de journalisation	Non

Champ	Description	Nullable
log_date	La date de l'événement	Oui
nodeid	L'id du noeud de l'élément de processus sous-jacent	Oui
nodeinstanceid	L'id de l'instance de noeud	Oui
nodename	Le nom du noeud sous-jacent	Oui
processid	L'id du processus sous-jacent	Oui
processinstanceid	L'id de l'instance de processus parente	Non
type	Le type d'événement (`0` = enter event, `1` = exit event)	Non

Tableau 14.3. Les champs de la table de VariableInstanceLog

Champ	Description	Nullable
id	La clé primaire de l'entité de journalisation	Non
log_date	La date de l'événement	Oui
processid	Le nom (id) du processus sous-jacent	Oui
processinstanceid	L'id de l'instance du processus	Non
value	La valeur de la variable au moment de la journalisation	Oui
variableid	La variable est définie dans la définition du processus	Oui
variableinstanceid	l'id de l'instance de la variable	Oui
résultat	Le résultat de l'instance de processus (comme des information de fin de processus, comme un code d'erreur)	Oui

Si nécessaire, définir votre propre modèle d'informations personnalisé et utiliser les listeners d'événement de processus pour extraire l'information.

14.1. JOURNALISATION DES ÉVÉNEMENTS DANS LA BASE DE DONNÉES

Pour journaliser un événement qui a lieu pendant le runtime d'une instance de processus, une instance d'élément, ou une instance de variable, vous devez procéder ainsi :

1. Mapper les classes de journalisation dans la source de données, de façon à ce que la source de données présentée accepte les entrées de journalisation. Dans Red Hat JBoss EAP, éditez les propriétés de sources de données dans le fichier **`persistence.xml`**.

Exemple 14.1. Les classes ProcessInstanceLog, NodeInstanceLog et VariableInstanceLog activées pour processInstanceDS

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<persistence  version="1.0"  xsi:schemaLocation=
    "http://java.sun.com/xml/ns/persistence
     http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd
     http://java.sun.com/xml/ns/persistence/orm
     http://java.sun.com/xml/ns/persistence/orm_1_0.xsd"
  xmlns:orm="http://java.sun.com/xml/ns/persistence/orm"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/persistence">

  <persistence-unit name="org.jbpm.persistence.jpa">
    <provider>org.hibernate.ejb.HibernatePersistence</provider>
    <jta-data-source>jdbc/processInstanceDS</jta-data-source>
    <class>org.drools.persistence.info.SessionInfo</class>

<class>org.jbpm.persistence.processinstance.ProcessInstanceInfo</c
lass>
    <class>org.drools.persistence.info.WorkItemInfo</class>
    <class>org.jbpm.process.audit.ProcessInstanceLog</class>
    <class>org.jbpm.process.audit.NodeInstanceLog</class>
    <class>org.jbpm.process.audit.VariableInstanceLog</class>

    <properties>
      <property name="hibernate.dialect"
value="org.hibernate.dialect.H2Dialect"/>
      <property name="hibernate.max_fetch_depth" value="3"/>
      <property name="hibernate.hbm2ddl.auto" value="update"/>
      <property name="hibernate.show_sql" value="true"/>
      <property name="hibernate.transaction.manager_lookup_class"

value="org.hibernate.transaction.BTMTransactionManagerLookup"/>
    </properties>
  </persistence-unit>
</persistence>
```

2. Enregistrer in logger dans votre session Kie.

Exemple 14.2. Importer les loggers

```
import org.jbpm.process.audit.AuditLogService;
```

```
import org.jbpm.process.audit.AuditLoggerFactory;
import org.jbpm.process.audit.AuditLoggerFactory.Type;
import org.jbpm.process.audit.JPAAuditLogService;
...
```

Exemple 14.3. Enregistrer un logger dans une session Kie

```
@PersistenceUnit(unitName = PERSISTENCE_UNIT_NAME)
  private EntityManagerFactory emf;

  private AuditLogService auditLogService;
@PostConstruct
  public void configure() {

  auditLogService = new JPAAuditLogService(emf);
  ((JPAAuditLogService)
auditLogService).setPersistenceUnitName(PERSISTENCE_UNIT_NAME);

  if( emf == null ) {
  ((JPAAuditLogService)
auditLogService).setPersistenceUnitName(PERSISTENCE_UNIT_NAME);
  }

  RuntimeEngine runtime =
singletonManager.getRuntimeEngine(EmptyContext.get());
  KieSession ksession = runtime.getKieSession();
  AuditLoggerFactory.newInstance(Type.JPA, ksession, null);

  }
```

3. En option, appeler la méthode **`addFilter`** sur le logger pour filtrer les informations inutiles. Seules les informations acceptées par tous les filtres apparaissent dans la base de données.

4. Les classes de logger peuvent être visualisées dans la vue d'auditing :

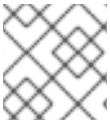
```
<dependency>
 <groupId>org.jbpm</groupId>
 <artifactId>jbpm-audit</artifactId>
 <version>6.0.1.Final</version>
</dependency>
```

14.2. FONCTIONNALITÉ LOGBACK

Red Hat JBoss BPM Suite fournit la fonctionnalité **`logback`** pour la configuration de la journalisation.

En conséquence, tout ce qui est configuré est consigné dans *Simple Logging Facade for Java*SLF4J, qui délègue à n'importe quel journal Logback, Apache Commons Logging, Log4j ou java.util.logging. Ajouter une dépendance à l'adaptateur de journalisation pour votre infrastructure de journalisation de choix. Si vous n'utilisez pas de framework de journalisation pour l'instant, vous pouvez utiliser Logback en ajoutant cette dépendance Maven :

```
<dependency>
   <groupId>ch.qos.logback</groupId>
   <artifactId>logback-classic</artifactId>
   <version>1.x</version>
</dependency>
```



NOTE

slf4j-nop and **slf4j-simple** se prêtent bien à un environnement léger.

14.3. CONFIGURER LA JOURNALISATION

Pour configurer le niveau de configuration des packages, créer un fichier **logback.xml** dans **business-central.war/WEB-INF/classes/logback.xml**. Pour définir le niveau de journalisation du package **org.drools** à "debug" pour une journalisation verbeuse, vous devez ajouter la ligne suivante au fichier :

```
<configuration>

    <logger name="org.drools" level="debug"/>

    ...

<configuration>
```

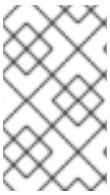
De même, vous pouvez configurer la journalisation pour des packages comme ceux qui suivent :

- org.guvnor
- org.jbpm
- org.kie
- org.slf4j
- org.dashbuilder
- org.uberfire
- org.errai
- etc...

Si vous configurez avec **log4j**, le **log4j.xml** peut être situé dans le fichier **business-central.war/WEB-INF/classes/log4j.xml** et peut être configuré de la manière suivante :

```
<log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j/">

    <category name="org.drools">
      <priority value="debug" />
    </category>
```

```
    ...

</log4j:configuration>
```



NOTE

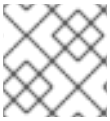
De plus, la journalisation peut être configurée dans le conteneur individuel. Pour configurer la journalisation dans JBoss Enterprise Application Platform, veuillez vous référer au Guide de configuration et d'administration de Red Hat JBoss EAP.

CHAPITRE 15. LOCALISATION ET PERSONNALISATION

15.1. LANGUES DISPONIBLES

L'interface utilisateur web Red Hat JBoss BPM Suite peut être vue en plusieurs langues :

- Anglais États-Unis (**en_US**)
- Espagnol (**es_ES**)
- Japonais (**ja_JP**)
- Chinois (**zh_CN**)
- Portuguais (**pt_BR**)
- Français (**fr_CA**)
- Allemand (**de_DE**)



NOTE

Si aucune langue n'est indiquée, l'anglais US est utilisé par défaut.

15.2. MODIFIER LES PARAMÈTRES DE LANGUE

Modifier la langue dans l'interface utilisateur de Business Central

Par défaut, Business Central utilise le paramètre régional du système. Si vous souhaitez le changer, ajouter le code correspondant au paramètre régional dans l'URL de Business Central. Par exemple, l'URL suivant indiquera le paramètre régional portuguais (pt_BR).

http://localhost:8080/business-central/?locale=pt_BR

Modifier la langue dans l'interface utilisateur de Dashbuilder

Pour modifier la langue dans l'interface utilisateur de Dashbuilder, procéder ainsi :

1. Connecter-vous à Dashbuilder une fois que le serveur a démarré en naviguant dans http://localhost:8080/dashbuilder par un navigateur web.
2. Sélectionner la langue de votre choix en cliquant sur les paramètres régionaux possibles en haut du centre de l'interface utilisateur de Dasbuilder pour modifier la langue.

Définir une langue d'interface utilisateur par défaut dans Dashbuilder

Voici un exemple pour définir la langue d'interface utilisateur par défaut dans Dashbuilder :

Procédure 15.1. Définir la langue par défaut Français

1. Naviguer dans **jboss-eap-6.1/standalone/configuration** et définir ce qui suit dans le fichier **standalone.xml**.

```
<system-properties>
    <property
name="org.jboss.dashboard.LocaleManager.installedLocaleIds"
```

```
value="en,es,de,fr,ja,pt,zh"/>
    <property
name="org.jboss.dashboard.LocaleManager.defaultLocaleId"
value="fr"/>
</system-properties>
```

2. La langue d'interface utilisateur par défaut de Dashbuilder est maintenant le français.

Définir les paramètres régionaux installés dans Dashbuilder

Voici un exemple sur la façon de définir les paramètres régionaux dans Dashbuilder :

Procédure 15.2. Définir le paramètre régional installé

- Naviguer dans **`jboss-eap-6.1/standalone/configuration`** et définir ce qui suit dans le fichier **`standalone.xml`**.

```
<system-properties>
    <property
name="org.jboss.dashboard.LocaleManager.installedLocaleIds"
value="en,es,de,fr,ja,pt"/>
    <property
name="org.jboss.dashboard.LocaleManager.defaultLocaleId"
value="fr"/>
</system-properties>
```

Dans cet exemple, la langue chinoise (zh) a été supprimée de la liste des paramètres régionaux installés pour les utilisateurs, donc les utilisateurs ne seront pas en mesure de passer au chinois dans Dashbuilder. Dashbuilder affichera le contenu en Français, qui est le paramètre régional par défaut. Les utilisateurs seront en mesure de sélectionner d'autres langues définies (fr, es, de, ja, pt) dans ce fichier.

NOTE

Dans Business Central, le serveur d'applications n'a pas besoin d'être démarré à nouveau après le changement de paramètre régional si vous avez déjà ajouté ce paramètre à l'URL de Business Central. Cependant, dans Dashbuilder, le serveur d'applications doit être démarré à nouveau après que les fichiers de configuration ont été modifiés.

15.3. EXÉCUTER UNE JVM AVEC LA CODIFICATION UTF-8

Red Hat JBoss BPM Suite a été conçu pour fonctionner avec un codage UTF-8. Si un système de codage différent est utilisé par une JVM sous-jacente, des erreurs inattendues peuvent se produire.

Pour vous assurer que UTF-8 est utilisé par la JVM, utilisez la propriété de système suivante "-Dfile.encoding=UTF-8"

PARTIE IV. EXÉCUTION

CHAPITRE 16. SERVEUR D'EXÉCUTION

16.1. RÈGLES D'ATTRIBUTION

Les règles d'attribution sont des règles exécutées automatiquement lorsqu'une tâche humaine est créée ou terminée. Ce mécanisme peut être utilisé, par exemple, pour affecter une tâche humaine automatiquement à un utilisateur particulier appartenant à un groupe ou pour empêcher un utilisateur d'effectuer une tâche si des données sont manquantes.

16.1.1. Définir les règles d'attribution

Pour définir les règles d'attribution, procéder ainsi :

1. Créer un fichier qui contienne la définition de la règle sur le chemin d'accès de Business Central (l'emplacement recommandé est ***$DEPLOY_DIR*/standalone/deployments/business-central.war/WEB-INF/classes/**):
   - **default-add-task.drl** avec les règles à vérifier quand la tâche humaine est créée.
   - **default-complete-task.drl** avec les règles à vérifier quand la tâche humaine est terminée.
2. Pour définir les règles dans le fichier.

Exemple 16.1. Le contenu de `default-add-task.drl`

```
package defaultPackage

import org.kie.api.task.model.Task;
import org.kie.api.task.model.User;
import org.kie.api.task.model.Status;
import org.kie.api.task.model.PeopleAssignments;
import org.jbpm.services.task.rule.TaskServiceRequest;
import org.jbpm.services.task.exception.PermissionDeniedException;
import org.jbpm.services.task.impl.model.*;
import java.util.HashMap;
import java.util.List;


global TaskServiceRequest request;

rule "Don't allow Mary to complete task when rejected"
    when
        $task : Task()
        $actualOwner : User( id == 'mary') from
$task.getTaskData().getActualOwner()
        $params : HashMap(this["approved"] == false)
    then
        request.setAllowed(false);
        request.setExceptionClass(PermissionDeniedException.class);
        request.addReason("Mary is not allowed to complete task with
approved false");
end
```

Si les propriétaires potentiels d'une tâche humaine contiennent l'utilisateur **Mary**, la tâche sera assignée automatiquement à l'utilisateur **mary**.

Exemple 16.2. Le contenu de `default-complete-task.drl`

```
package defaultPackage

import org.kie.api.task.model.Task;
import org.kie.api.task.model.User;
import org.kie.api.task.model.Status;
import org.kie.api.task.model.PeopleAssignments;
import org.jbpm.services.task.rule.TaskServiceRequest;
import org.jbpm.services.task.exception.PermissionDeniedException;
import org.jbpm.services.task.impl.model.*;
import java.util.HashMap;
import java.util.List;


global TaskServiceRequest request;

rule "Don't allow Mary to complete task when rejected"
    when
        $task : Task()
        $actualOwner : User( id == 'mary') from
$task.getTaskData().getActualOwner()
        $params : HashMap(this["approved"] == false)
    then
        request.setAllowed(false);
        request.setExceptionClass(PermissionDeniedException.class);
        request.addReason("Mary is not allowed to complete task without
approval.");
end
```

Si les propriétaires potentiels d'une tâche humaine contiennent l'utilisateur **Mary**, la tâche sera assignée automatiquement à l'utilisateur **mary**.

16.2. SESSION MAIL

Une session mail définit les propriétés de serveur mail utilisées pour envoyer des emails quand l'application l'exige, comme les mécanismes de notification ou d'escalation (voir le *Red Hat JBoss BPM Suite User Guide*).

16.2.1. Mise en place d'une session mail

Pour mettre en place une session mail pour votre moteur d'exécution, procédez ainsi :

1. Ouvrir le fichier de configuration du profil respectif (**`standalone.xml`** ou **`host.xml`**) pour effectuer les modifications.
2. Ajouter la session mail au sous-système **`urn:jboss:domain:mail:1.1`**.

   Exemple 16.3. Nouvelle session mail sur le localhost

```
<subsystem xmlns="urn:jboss:domain:mail:1.1">
    <!-- omitted code -->

    <mail-session jndi-name="java:/mail/bpmsMailSession"
debug="true" from="bpms@company.com">
        <smtp-server outbound-socket-binding-ref="bpmsMail"/>
    </mail-session>
</subsystem>
```

3. Définir un socket sortant dans le fichier de configuration du profil.

Exemple 16.4. Définition du socket sortant

```
<outbound-socket-binding name="bpmsMail">
    <remote-destination host="localhost" port="12345"/>
</outbound-socket-binding>
```

CHAPITRE 17. PLUG-IN DE RED HAT JBOSS DEVELOPER STUDIO

17.1. PLUG-IN

PARTIE V. MONITORING

CHAPITRE 18. MONITORING DES PROCESSUS

18.1. RÉSEAU JBOSS OPERATIONS NETWORK

Un plug-in de réseau d'exploitation JBoss Operations Network peut être utilisé pour surveiller les sessions de règles pour Red Hat JBoss . Le plug-in utilise Java Management Extensions (JMX) pour surveiller les sessions de règles.

En raison d'une limitation dans la possibilité de passer des arguments de monitoring JVM par la ligne de commande Maven, tous les paramètres **`com.sun.management.jmxremote.*`** doivent être passés à l'application JBoss par le fichier de configuration **`pom.xml`**.

Veuillez consulter **JBoss Operations Network** *Installation Guide* pour obtenir des instructions d'installation pour le serveur JBoss ON.

18.2. INSTALLER LE PLUG-IN JBOSS BRMS DANS JBOSS ON

Le plug-in Red Hat JBoss BRMS de JBoss Operations Network peut être installé, soit en copiant les fichiers JAR du plug-in dans le répertoire du plug-in JBoss Operations Network, soit par le GUI de JBoss Operations Network.

La procédure suivante guide un utilisateur à copier les fichiers JAR du plug-in dans le répertoire du plug-in JBoss Operations Network

Procédure 18.1. Copier les fichiers JAR du plug-in JBoss BRMS

1. Extraire l'archive du pack de plug-in JBoss BRMS dans un lieu temporaire. Cela crée un sous-répertoire nommé jon-plugin-pack-brms-bpms-3.3.0.GA. Exemple :

```
[root@server rhq-agent]# unzip jon-plugin-pack-brms-bpms-
3.3.0.GA.zip -d /tmp
```

2. Copier les fichiers JAR du plug-in de JBoss BRMS du répertoire jon-plugin-pack-brms-bpms-3.3.2.GA/ dans le répertoire du plug-in du serveur JBoss ON. Exemple :

```
[root@server rhq-agent]# cp /tmp/jon-plugin-pack-brms-bpms-
3.3.0.GA/*.jar /opt/jon/jon-server-3.3.0.GA1/plugins
```

3. Démarrer le serveur JBoss Operations Network pour mettre à jour le plug-in de JBoss BRMS.

Pour télécharger le plug-in de JBoss BRMS dans le GUI de JBoss Operations Network, suivre la procédure suivante :

Procédure 18.2. Télécharger le plug-in de JBoss BRMS par le GUI

1. Démarrer le serveur de JBoss Operations Network et connectez-vous pour accéder au GUI.
2. Dans la navigation supérieure du GUI, ouvrez le menu **`Administration`**.
3. Dans la zone **`Configuration`** sur la gauche, sélectionner le lien **`Plugins Serveur`**.
4. Au bas de la liste des plug-ins de serveur téléchargés, cliquer sur le bouton **`Télécharger un plugin`** et choisissez le plug-in BRMS.

5. Le plug-in JBoss BRMS de JBoss Operations Network est maintenant téléchargé.

18.3. CONTRÔLE DES KIEBASES ET DES KIESESSIONS

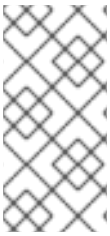
Pour que JBoss Operations Network puisse contrôler les KieBases et les KieSessions, les MBeans doivent être activés.

Les MBeans peuvent être activés si on leur passe le paramètre :

```
-kie.mbeans = enabled
```

Ou via l' API :

```
KieBaseConfiguration kbconf =
KieServices.Factory.get().newKieBaseConfiguration();
    kbconf.setOption(MBeansOption.ENABLED);
```



NOTE

Les **`Kie Services`** ont été créés pour JBoss BRMS 6; pour JBoss BRMS 5, **`Drools Services`** était la convention de nommage utilisée et elle a des mensurations différentes suivant les sessions. Par exemple, le renommage de **activation** → **correspondance** est apparu dans la version mise à jour.

Veuillez consulter le guide *JBoss Operations Network Resource Monitoring and Operations Reference*pour obtenir des informations sur la façon d'importer des Sessions Kie dans la vue Inventaire à des fins de monitoring.

CHAPITRE 19. GESTION DE LA SÉCURITÉ DANS RED HAT JBOSS BPM SUITE DASHBUILDER

19.1. ACCÉDER À RED HAT JBOSS BPM SUITE DASHBUILDER

Dashbuilder est l'interface utilisateur basée web de Red Hat JBoss BPM Suite pour le Business Activity Monitoring. Pour accéder au Dashbuilder à partir de Business Central, aller dans **Tableaux de bord** → **Tableaux de processus et de tâches**.

Le tableau de bord qui s'affiche fournit des statistiques sur les données de runtime sélectionnées à gauche. Vous pouvez créer votre propre tableau de bord dans le Dashbuilder. Pour cela, afficher le Dashbuilder en cliquant sur **Tableaux de bord** → **Tableaux de processus**.

19.2. GESTION DE LA SÉCURITÉ

Pour gérer la sécurité, vous devez définir des polices d'autorisation personnalisées pour pouvoir octroyer ou refuser un accès à un espace de travail, une page, ou des instances de panneau par rôle.

Vous trouverez ci-dessous une liste des rôles disponibles de Dashbuilder :

- admin - administre le système Red Hat JBoss BPM Suite. A tous les droits de procéder aux changements nécessaires. A également la possibilité d'ajouter ou de supprimer des utilisateurs du système.
- developer - implémente le code requis pour faire fonctionner les processus. Utilise surtout la connexion JBDS pour voir les processus, mais peut utiliser l'outil web de temps en temps :
- analyst - est responsable de créer ou de concevoir des processus dans le système. Crée des flux de processus et est chargé des demandes de changement de processus. Doit tester les processus qu'ils créent. Crée également des formulaires et des tableaux de bord.
- user - utilisateur quotidien du système qui réalise les tâches d'entreprise requises pour que les processus puissent avancer. Travaille principalement à partir d'une liste de tâches.
- manager - observateur du système intéressé par les statistiques autour des processus métier et leur performance, les indicateurs d'entreprise, les autres rapports sur le système, et les personnes en rapport avec le système.

Grâce au système d'autorisations, vous pouvez créer une structure d'espace de travail de plusieurs pages, menus et panneaux, et définir quelles pages et panneaux d'une page seront visibles pour chaque rôle. Vous pouvez également définir des types spéciaux d'utilisateurs et leur donner un accès restreint à certaines fonctionnalités, ou même un accès restreint à un sous-ensemble de page.

19.3. PERMISSIONS D'ESPACE DE TRAVAIL

Procédure 19.1. Accéder à des permissions d'espace de travail

1. Connectez-vous aux Tableaux de bord d'entreprise de Business Central (comme décrit dans le sujet Accéder à Red Hat JBoss BPM Suite Dashbuilder)
2. Sélectionner le tableau de bord qui convient à partir du menu déroulant de l'espace de travail.

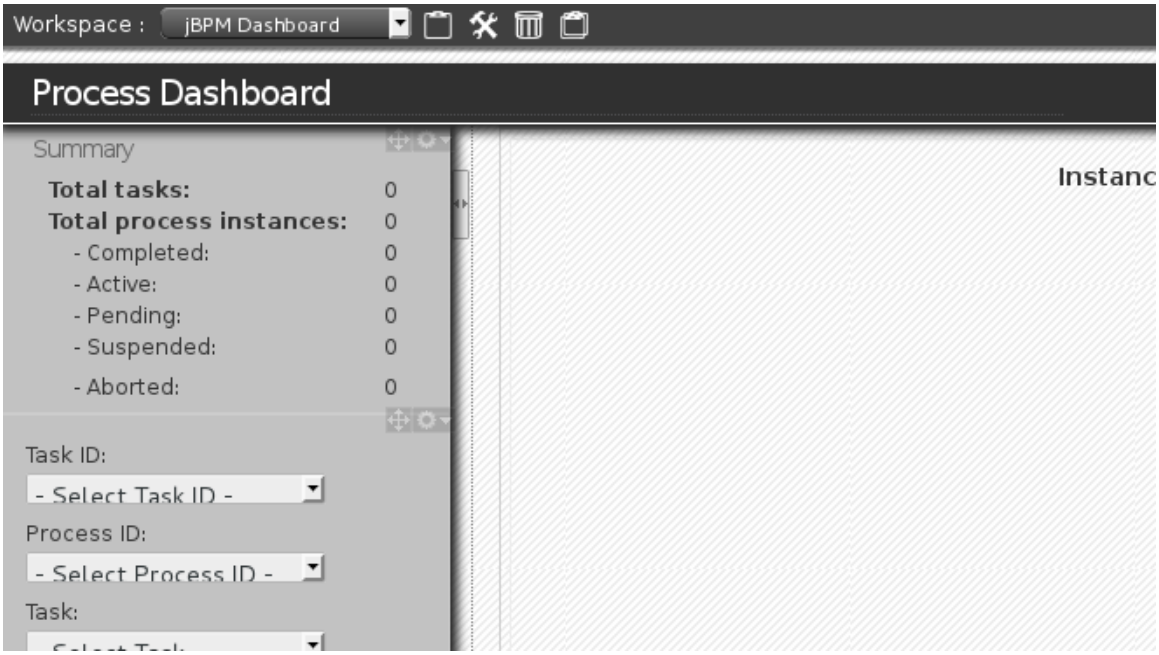


Figure 19.1. Espace de travail de Dashbuilder

3. Cliquer sur le bouton **`Edit selected workspace properties`**  pour accéder au Tableau de bord d'espace de travail.

4. Cliquer sur l'étiquette **`Permissions`** pour voir l'écran de gestion des permissions.

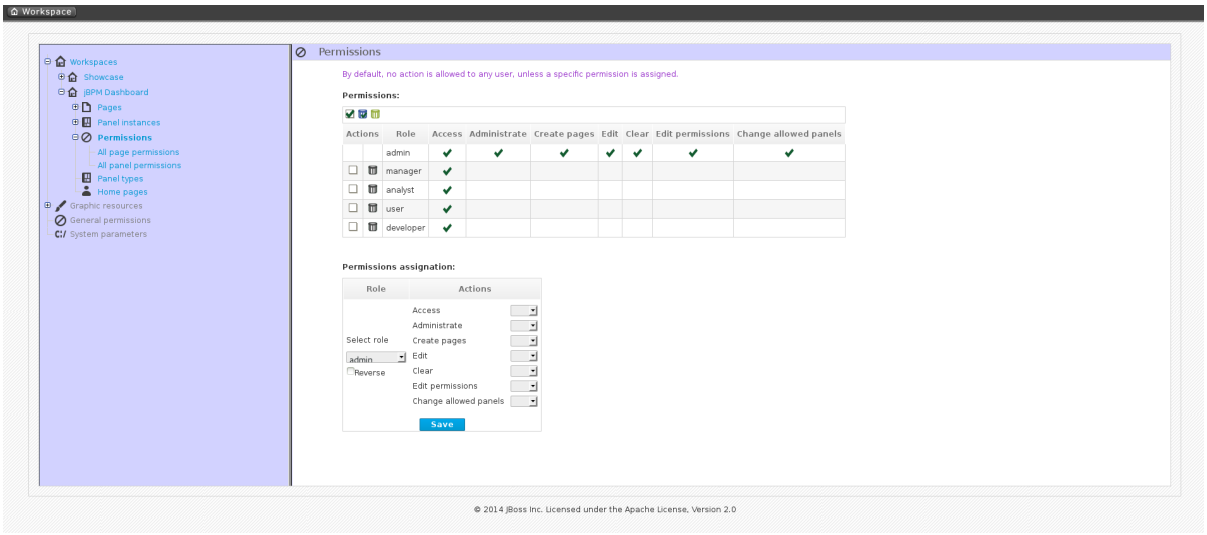


Figure 19.2. Écran des permissions

Dans la section **`Allocation de permissions`**, vous trouverez une liste des actions autorisées s'appliquant au rôle sélectionné :

- **Access**: permission de vous connecter à l'application
- **Administrate**: permission d'accéder à la barre d'outils et aux fonctionnalités de configuration du système.
- **Create pages**: possibilité de créer des nouvelles pages de projet.
- **Edit** : permission de modifier des propriétés d'espace de travail.

- **Clear** : possibilité de supprimer l'espace de travail.
- **Edit permissions** : possibilité d'octroyer/refuser des permissions.
- **Change allowed panels** : permission de limiter le type de panneaux qui peuvent être utilisés dans cet espace de travail.

Pour assigner une permission, vous devez sélectionner le rôle cible et la liste des actions autorisées pour une ressource donnée.

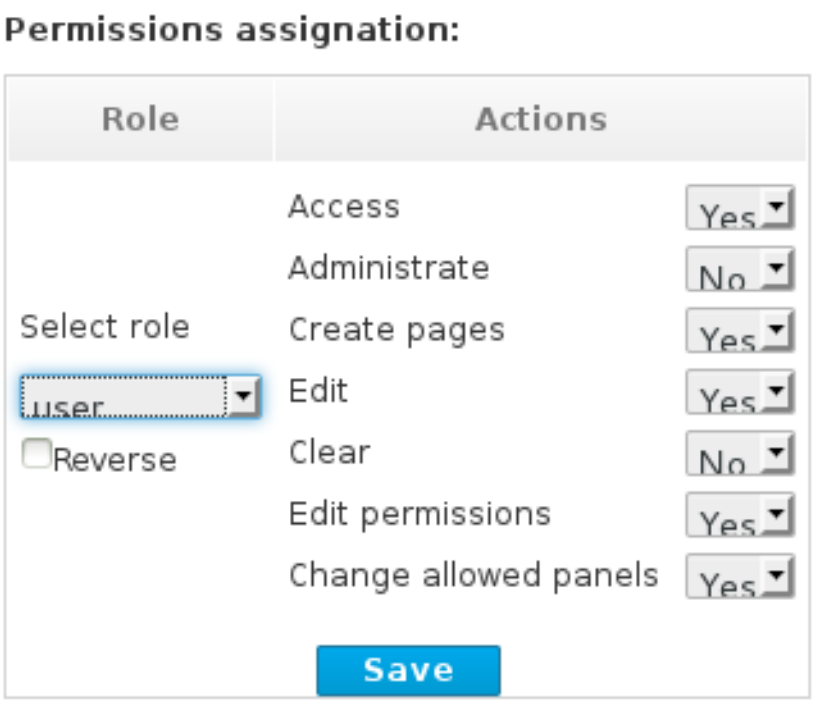


Figure 19.3. Attribution des permissions

- *Target roles (who)* : quel utilisateur recevra/ne recevra pas les permissions définies.
- *Allowed actions* : suivant le type de ressource, nous pouvons activer/désactiver ce que l'utilisateur peut faire avec cette ressource.
- *Reverse (optional)* : quand on a un groupe de rôles et que l'on veut donner/refuser une permission à tous les rôles sauf un.

NOTE

Par défaut, l'ensemble des permissions vont au rôle *admin*. Cela rend les choses plus faciles pour créer un utilisateur qui peut tout faire, tant que le rôle *admin* est attribué.

19.4. PERMISSIONS DE PAGE

1. Pour accéder aux **`Page permissions`**, chercher le menu déroulant **`Pages`** qui se trouve sous le Tableau de bord jBPM (ou un autre tableau de bord que vous aurez sélectionné).
2. Étendre le menu **`Pages`**, puis l'option **`Tableau de bord de processus`**.
3. Sélectionner l'option **`Permissions de page`**.

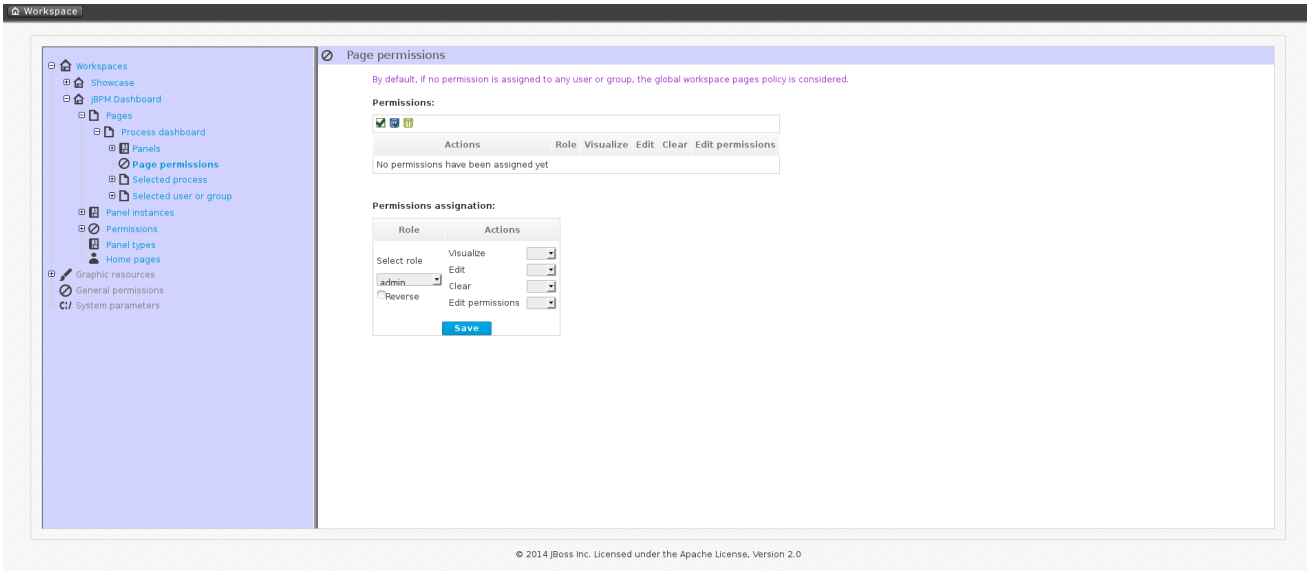


Figure 19.4. Permissions de page

Dans la section **Allocation de permissions**, vous trouverez une liste des actions autorisées s'appliquant au rôle sélectionné :

- **Visualize** : permission pour rendre la page visible.
- **Edit** : possibilité de modifier les propriétés d'une page.
- **Clear** : possibilité de supprimer la page.
- **Edit permissions** : possibilité d'octroyer/refuser des permissions pour une page.

19.5. PANNEAU DE PERMISSIONS

1. Pour accéder à la page **Permissions de panneau**, étendre l'option **Instnaces de panneau** sous le Tableau de bord jBPM (ou un autre tableau de bord que vous utilisez).
2. Étendre l'option **Tableau de bord**, puis étendre **Tableau de bord de processus**.
3. Étendre le choix **Panneaux** et sélectionner le processus qui convient.
4. Ouvrir la page **Permissions de panneau**.

Vous trouverez, ci-dessous, un écran de gestion des permissions pour un panneau donné (dans cet exemple, le panneau de processus) :

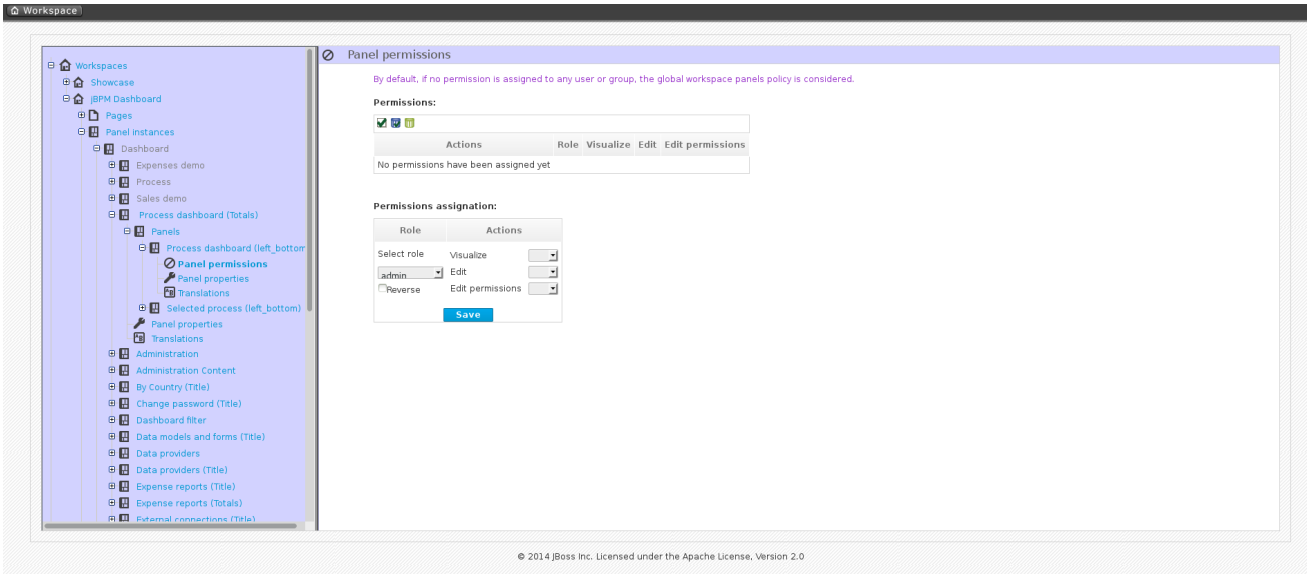


Figure 19.5. Écran de configuration des permissions du panneau

Les actions autorisées sont les suivantes :

- **Visualize** : pour rendre la page visible.
- **Edit** : pour modifier des propriétés d'espace de travail.
- **Edit permissions** : octroyer/refuser des permissions pour une page.

ANNEXE A. HISTORIQUE DES VERSIONS

Version 1.0.0-1.2 **Mon Feb 15 2016** **Corina Roe**

Updating French translations

Version 1.0.0-1.1 **Tue Jan 5 2016** **Red Hat Localization Services**

Fichiers de traduction synchronisés avec les sources XML 1.0.0-1

Version 1.0.0-1 **Wed Aug 05 2015** **Petr Penicka**

Créé à partir de Content Specification: 22829, Revision: 765028 par ppenicka