IBM Tivoli Workload Automation

Guide du développeur : Gestion de Tivoli Workload Automation

Version 9.2

SC11-6398-03

IBM Tivoli Workload Automation

Guide du développeur : Gestion de Tivoli Workload Automation

Version 9.2

SC11-6398-03

Important
Avant d'utiliser le présent document et le produit associé, prenez connaissance des informations générales figurant à la section «Remarques», à la page 83.

Réf. US : SC23-9608-03

Ce document est mis à jour périodiquement. Chaque nouvelle édition inclut les mises à jour. Les informations qui y sont fournies sont susceptibles d'être modifiées avant que les produits décrits ne deviennent eux-mêmes disponibles. En outre, il peut contenir des informations ou des références concernant certains produits, logiciels ou services non annoncés dans ce pays. Cela ne signifie cependant pas qu'ils y seront annoncés.

Pour plus de détails, pour toute demande d'ordre technique, ou pour obtenir des exemplaires de documents IBM, référez-vous aux documents d'annonce disponibles dans votre pays, ou adressez-vous à votre partenaire commercial.

Vous pouvez également consulter les serveurs Internet suivants :

- http://www.fr.ibm.com (serveur IBM en France)
- http://www.can.ibm.com (serveur IBM au Canada)
- http://www.ibm.com (serveur IBM aux Etats-Unis)

Compagnie IBM France
Direction Qualité
17, avenue de l'Europe
92275 Bois-Colombes Cedex

La présente édition s'applique à la version 9.2.0 de Tivoli Workload Scheduler (référence de logiciel 5698-WSH), ainsi qu'à toutes les éditions et modifications ultérieures, sauf mention contraire dans les nouvelles éditions.

Table des matières

Avis aux lecteurs canadiens

Le présent document a été traduit en France. Voici les principales différences et particularités dont vous devez tenir compte.

Illustrations

Les illustrations sont fournies à titre d'exemple. Certaines peuvent contenir des données propres à la France.

Terminologie

La terminologie des titres IBM peut différer d'un pays à l'autre. Reportez-vous au tableau ci-dessous, au besoin.

IBM France	IBM Canada
ingénieur commercial	représentant
agence commerciale	succursale
ingénieur technico-commercial	informaticien
inspecteur	technicien du matériel

Claviers

Les lettres sont disposées différemment : le clavier français est de type AZERTY, et le clavier français-canadien de type QWERTY.

OS/2 et Windows - Paramètres canadiens

Au Canada, on utilise :

- les pages de codes 850 (multilingue) et 863 (français-canadien),
- le code pays 002,
- le code clavier CF.

Nomenclature

Les touches présentées dans le tableau d'équivalence suivant sont libellées différemment selon qu'il s'agit du clavier de la France, du clavier du Canada ou du clavier des États-Unis. Reportez-vous à ce tableau pour faire correspondre les touches françaises figurant dans le présent document aux touches de votre clavier.

France	Canada	Etats-Unis
(Pos1)		Home
Fin	Fin	End
(PgAr)		PgUp
(PgAv)		PgDn
Inser	Inser	Ins
Suppr	Suppr	Del
Echap	Echap	Esc
Attn	Intrp	Break
Impr écran	ImpEc	PrtSc
Verr num	Num	Num Lock
Arrêt défil	Défil	Scroll Lock
(Verr maj)	FixMaj	Caps Lock
AltGr	AltCar	Alt (à droite)

Brevets

Il est possible qu'IBM détienne des brevets ou qu'elle ait déposé des demandes de brevets portant sur certains sujets abordés dans ce document. Le fait qu'IBM vous fournisse le présent document ne signifie pas qu'elle vous accorde un permis d'utilisation de ces brevets. Vous pouvez envoyer, par écrit, vos demandes de renseignements relatives aux permis d'utilisation au directeur général des relations commerciales d'IBM, 3600 Steeles Avenue East, Markham, Ontario, L3R 9Z7.

Assistance téléphonique

Si vous avez besoin d'assistance ou si vous voulez commander du matériel, des logiciels et des publications IBM, contactez IBM direct au 1 800 465-1234.

Figures

Tableaux

A propos de ce guide

Fournit une présentation du guide, avec des informations concernant les changements apportés depuis la dernière édition et le public visé. Fournit également des informations concernant le mode d'obtention des ressources et l'assistance technique d'IBM®.

IBM Tivoli Workload Automation : guide du développeur de logiciel - prise en charge de Tivoli Workload Automation présente les interfaces de programme d'application disponibles pour prendre en charge les produits Tivoli Workload Automation à partir de vos propres applications. Ce guide explique également comment utiliser le Integration Workbench fourni avec le produit pour développer et implémenter ces interfaces de programme d'application.

Par exemple, grâce aux informations de cette publication et à l'aide associée contenues dans Integration Workbench, votre application peut générer un ou plusieurs travaux et un flot de travaux, soumettre le flot de travaux dans le plan, surveiller sa progression et supprimer les objets créés lors d'une exécution réussie, le tout sans exécuter manuellement **composer**, **conman** ou Dynamic Workload Console.

Nouveautés

Prenez connaissance des nouveautés de cette édition.

Pour plus d'informations à propos des fonctions nouvelles ou modifiées de cette édition, consultez la section *Récapitulatif des améliorations* dans *Tivoli Workload Automation - Présentation*.

Pour plus d'informations sur les APAR de cette édition, consultez les Notes sur l'édition de Tivoli Workload Scheduler à l'adresse http://www-01.ibm.com/support/docview.wss?rs=672&uid=swg27041032 et les Notes sur l'édition de Dynamic Workload Console à l'adresse http://www-01.ibm.com/support/docview.wss?rs=672&uid=swg27041033.

Public visé

Décrit le type d'utilisateur auquel cette documentation est destinée.

Cette publication fournit des informations sur la création d'interfaces de Tivoli Workload Scheduler et Tivoli Workload Scheduler for z/OS autres que celles fournies avec le produit.

Le lecteur de ce manuel doit être un spécialiste de *programmation d'application* de services Java™ ou Web (suivant le cas) ayant une bonne compréhension de l'infrastructure Tivoli Workload Automation et de ses interactions entre les composants ou le responsable de cette personne qui souhaite savoir ce qui peut être réalisé à l'aide des interfaces de programme d'application.

La publication suppose que le programmeur d'application possède une expérience significative dans la création de ce type d'interface. Elle suppose également que toute connaissance nécessaire pour programmer l'interface de programme d'application ou interface des services Web provienne de la documentation produit.

Cette publication ne tente pas d'expliquer les concepts, procédures et pratiques Tivoli Workload Automation auxquels elle se rapporte.

Ce manuel comporte également des informations utiles à la planification pour l'*administrateur informatique* et l'*Tivoli Workload Automationadministrateur informatique*.

Publications

Le produit Tivoli Workload Automation fait l'objet de plusieurs publications.

Pour connaître la liste des publications disponibles dans la bibliothèque du logiciel Tivoli Workload Automation, voir *Publications* sous *Référence* dans la documentation du produit.

Pour connaître la liste des termes utilisés dans le produit Tivoli Workload Automation, voir *Glossaire* sous *Référence* dans la documentation du produit.

Accessibilité

Les fonctions d'accessibilité permettent aux personnes souffrant d'un handicap physique (par exemple, une mobilité réduite ou une déficience visuelle) de pouvoir utiliser les logiciels.

Avec ce produit, vous pouvez utiliser les technologies d'assistance pour parcourir l'interface à l'aide de messages sonores. Vous pouvez également utiliser le clavier au lieu de la souris pour toutes les fonctions de l'interface graphique.

Pour des informations complètes concernant Dynamic Workload Console, consultez l'annexe correspondante du document *IBM Tivoli Workload Scheduler - Guide d'utilisation et de référence*.

Formation technique Tivoli

Tivoli propose une formation technique.

Pour plus d'informations sur la formation technique Tivoli, consultez le site Web IBM Tivoli Education à l'adresse :

http://www.ibm.com/software/tivoli/education

Informations de support

IBM vous propose plusieurs façons d'obtenir de l'aide lorsque vous êtes confronté à un problème.

Si vous rencontrez un incident avec un logiciel IBM, vous pouvez le résoudre rapidement. IBM vous permet d'obtenir l'assistance que vous souhaitez de plusieurs manières :

- En faisant des recherches dans les bases de connaissances : elles contiennent un grand nombre d'incidents recensés et de solutions, de remarques d'ordre technique et autres informations adéquates.
- En vous procurant des correctifs : vous trouverez les versions les plus récentes disponibles pour votre produit.

- En contactant le service de support logiciel IBM : si les deux solutions ci-dessus ne vous ont pas permis de résoudre votre incident, vous pouvez contacter directement un technicien IBM de plusieurs manières.

Pour plus d'informations sur ces trois manières de résoudre un incident, voir l'annexe relative aux informations de support dans le manuel *Tivoli Workload Scheduler - Guide d'identification et de résolution des problèmes*.

Chapitre 1. Présentation de la prise en charge de Tivoli Workload Automation

Fournit une présentation de l'intégralité de la publication.

IBM Tivoli Workload Automation : guide du développeur de logiciel - prise en charge de Tivoli Workload Automation décrit deux interfaces de programme d'application que vous pouvez utiliser pour prendre en charge les produits Tivoli Workload Automation à partir de vos propres applications :

- Utilisez Java interface de programme d'application pour créer votre propre interface graphique ou interface de ligne de commande afin d'exécuter toutes les fonctions des programmes de ligne de commande **composer**, **conman** et **planman** ainsi que Dynamic Workload Console. Cela comprend également exécuter les tâches suivantes dans Tivoli Workload Scheduler et Tivoli Workload Scheduler for z/OS :
  - Modifier les objets de la base de données
  - Soumettre la charge de travail
  - Surveiller le plan
  - Exécuter des actions dans le plan, telles que des actions correctives en cas d'échec du travail
- Utiliser interface des services Web pour créer votre propre application client afin d'exécuter un sous-ensemble des fonctions Tivoli Workload Scheduler et Tivoli Workload Scheduler for z/OS pour gérer les travaux et flots de travaux dans le plan. Ni les actions de la base de données, ni les autres actions du plan ne peuvent être implémentées et appelées à l'aide de cette interface.

Vous pouvez utiliser le manuel Tivoli Workload Automation Software Development Kit'sIntegration Workbench fourni avec le produit pour développer et implémenter ces interfaces de programme d'application.

Les informations concernant interfaces de programme d'application s'organisent de cette façon :

- Chapitre 2, «Integration Workbench», à la page 3
- Chapitre 3, «Prise en charge de Tivoli Workload Automation avec l'interface de programme d'application Java», à la page 5
- Chapitre 4, «Gestion de Tivoli Workload Automation avec l'interface de services Web», à la page 33

Chapitre 2. Integration Workbench

Décrit le module Integration Workbench du composant Software Development Kit.

IBM Tivoli Workload Automation : Software Development Kit est livré avec le module Integration Workbench que vous pouvez utiliser pour exploiter le système Java interface de programme d'application et le interface des services Web pour développer vos propres applications.

Cette section vous explique comment installer et utiliser l'aide d'Integration Workbench. L'aide contient des informations détaillées sur les tâches vous pouvez exécuter avec Integration Workbench ainsi que les informations de référence détaillées concernant les méthodes et classes disponibles :

Installation d'Integration Workbench

Fournit une présentation de l'installation d'Integration Workbench.

Integration Workbench (composant de Software Development Kit - SDK) est exécuté sous Eclipse. L'installation, décrite de manière détaillée dans *Tivoli Workload Scheduler - Guide de planification et d'installation*, *SC11-2008*, vous donne la possibilité d'installer Integration Workbench et une version fournie et prise en charge d'Eclipse en une seule fois ou d'installer Integration Workbench en tant que *site Eclipse* à l'aide d'une version existante et prise en charge d'Eclipse, disponible sur votre réseau.

Dans les deux cas, à la fin de l'installation, il existe une option pour afficher le fichier `readmefirst.html` qui contient les informations sur le plan de travail et son exécution dans le panneau où vous cliquez sur **Terminer**. Ces informations sont également fournies dans «Utilisation de l'aide d'Integration Workbench».

Pour plus d'informations sur Eclipse, allez à http://www.eclipse.org/.

Utilisation de l'aide d'Integration Workbench

Explique comment accéder à la fonction d'aide d'Integration Workbench pour les projets de plug-in et d'interface de programme d'application Tivoli Workload Scheduler.

Pour utiliser l'aide d'Integration Workbench, procédez comme suit :

1. Lancez le plan de travail comme suit :

   Integration Workbench installé avec Eclipse

   UNIX Lancez le fichier suivant : `<TWS_home>/TWS/IntegrationWorkbench/eclipse/eclipse`

   Windows
   Allez dans **Démarrer → Tivoli Workload Scheduler → Integration Workbench**

   Integration Workbench installé en tant que site Eclipse
   Ouvrez votre version d'Eclipse comme à l'accoutumée.

2. Sélectionnez l'emplacement où sauvegarder votre espace de travail Eclipse. Eclipse le demande chaque fois que vous l'exécutez ou que vous y exécutez le plan de travail, sauf si vous cochez l'option pour sauvegarder un emplacement spécifique par défaut.
3. Lorsque la fenêtre Eclipse s'ouvre, sélectionnez **Aide → Table des matières**
4. cliquez sur **IBM Tivoli Workload Scheduler Integration Workbench**
5. Les options affichées proposent un grand nombre d'informations concernant Tivoli Workload Scheduler Integration Workbench. Par exemple, pour plus d'informations concernant toutes les classes et méthodes utilisées dans l'interface de programme d'application, cliquez sur **Référence** et sélectionnez **API reference**.

Remarque : Les informations ci-dessus peuvent être également consultées en ouvrant le document suivant dans le navigateur Web : *<TWS_home>*/TWS/IntegrationWorkbench/readmefirst.html

Chapitre 3. Prise en charge de Tivoli Workload Automation avec l'interface de programme d'application Java

Ce chapitre décrit l'interface de programme d'application J2EE qui utilise Enterprise Java Beans pour prendre en charge Tivoli Workload Scheduler et Tivoli Workload Scheduler for z/OS.

Vous pouvez utiliser l'API Java pour exécuter toutes les tâches disponibles dans :

- Dynamic Workload Console
- composer
- conman
- planman

Conventions de dénomination

Les conventions d'attribution de nom des objets Java sont assez simples. Par exemple, pour déterminer si une définition de travail spécifique de la base de données utilise une commande ou un script, utilisez une méthode appeléeisCommand dans une classe appelée JobDefinition.

La convention la plus importante à retenir c'est qu'un objet de la base de données se différencie d'un objet du plan par le suffixe "InPlan" du nom de la classe d'objets.

Spécification détaillée de l'interface de programme d'application

Fournit des informations concernant l'accès à l'aide de référence de l'interface de programme d'application Javadoc.

La spécification complète des beans Java peut être trouvée dans l'aide en ligne de référence de l'API Javadoc. Il s'agit du document décrivant en détail toutes les classes et méthodes. La spécification complète des beans Java peut être consultée de l'une des manières suivantes :

- Dans l'aide de **Tivoli Workload Scheduler Integration Workbench**, cliquez sur **Référence** et sélectionnez la **référence de l'interface de programme d'application**
- Ouvrez le fichier HTML suivant : <*TWA_home*>/TWS/APIs/doc/Javadoc/index.html

Pour obtenir une description de l'utilisation des différentes sous-fenêtres du panneau de interface de programme d'application Javadoc, cliquez sur **Aide**.

Il n'est pas conseillé d'utiliser les classes de la catégorie *Deprecated* (Obsolète).

Projets de l'interface de programme d'application Tivoli Workload Scheduler

Décrit les projets de l'interface de programme d'application.

Projets de l'interface de programme d'application

Les projets décrits dans ce document sont conçus pour une connexion à une instance de la version correspondante de Tivoli Workload Scheduler et pour une interaction avec cette instance à l'aide des méthodes proposées par l'interface de programme d'application Java.

Structure d'un projet de l'interface de programme d'application

Les assistants du projet d'interface de programme d'application proposent une structure contenant tout ce dont vous avez besoin pour vous connecter à l'instance Tivoli Workload Scheduler requise :

«Arbre source Java (src)»
Répertoires distincts de la source et des fichiers classe.

«Bibliothèques Java», à la page 7
Une bibliothèque système JRE et des bibliothèques distinctes sont disponibles pour l'objet Tivoli Workload Scheduler et les fichiers JAR d'exécution.

Répertoires clé
Répertoire contenant un fichier *.jks nécessaire pour accéder à la connexion sécurisée Tivoli Workload Scheduler.

Répertoire config
Répertoire contenant tous les fichiers de configuration que vous devez définir pour une connexion à Tivoli Workload Scheduler.

Une ou plusieurs unités de compilation Java
Une unité de compilation contient une classe qui implémente l'interface Java pour la connexion à Tivoli Workload Scheduler. Une autre est une unité de compilation vide avec le chemin d'accès aux classes déjà configuré et prêt pour une exécution avec la logique de programme dont vous avez besoin.

«build.xml», à la page 8
Fichier de génération ANT standard que vous modifiez pour répondre à vos besoins.

Création de projets d'interface de programme d'application

Pour créer des projets d'interface de programme d'application, procédez de l'une des deux manières suivantes :

«Création d'un projet de A à Z», à la page 8
Pour exécuter un assistant, fournissez des informations relatives au type de projet que vous souhaitez créer.

«Création d'un projet à partir d'un exemple d'interface de programme d'application», à la page 8
Dans une liste d'exemples, sélectionnez un projet d'interface de programme d'application identique au projet que vous souhaitez créer.

Editez ensuite le nouveau projet pour qu'il exécute la tâche requise.

Arbre source Java (src)

Décrit l'arbre source Java.

Lorsque vous créez un projet d'interface de programme d'application, il est configuré comme un projet Java avec des dossiers distincts pour les fichiers source et classe. Le dossier source est appelé `src`. Il contient le code Java de l'application.

Une série de classes Java est également créée avec le nouveau projet.

Reportez-vous aux informations de référence de l'API Java de Tivoli Workload Scheduler pour connaître la méthode à implémenter. Pour la consignation et le traçage de votre code, utilisez les interfaces de programme d'application de consignation Java JSR-047 standard.

Bibliothèques Java

Décrit les bibliothèques Java.

Les projets de l'interface de programme d'application Tivoli Workload Scheduler sont créés avec les bibliothèques suivantes :

Bibliothèque système JRE par défaut
Même si l'environnement JRE est paramétré par défaut, n'oubliez pas que le processeur d'événement IBM Tivoli Workload Scheduler est exécuté à l'aide de IBM JDK version 1.5.

Il est conseillé d'utiliser IBM JDK version 1.5 pour les projets d'interface de programme d'application Tivoli Workload Scheduler.

Bibliothèque IBM Tivoli Workload Scheduler
Cette bibliothèque contient tous les fichiers JAR Tivoli Workload Scheduler nécessaires pour implémenter les plug-in Tivoli Workload Scheduler ou pour utiliser les interfaces de programme d'application Tivoli Workload Scheduler.

La bibliothèque définit également les règles d'accès pour les classes dans les fichiers JAR : les fichiers JAR publiques sont définis comme étant accessibles alors que les classes internes sont définies comme désactivées (Discouraged).

Bibliothèque d'exécution IBM Tivoli Workload Scheduler

Cette bibliothèque contient tous les fichiers JAR Tivoli Workload Scheduler nécessaires pour une exécution assurée par les applications basées sur l'interface de programme d'application

L'utilisation de classes désactivées est accompagnée par défaut d'avertissements de compilateur. L'utilisation de ces classes n'est pas prise en charge dans tous les cas.

Les bibliothèques supplémentaires nécessaires pour le code Java de plug-in peuvent être copiées dans le dossier `lib` et ajoutées au chemin de génération Java.

Pour plus d'informations sur ces bibliothèques, reportez-vous à l'aide de Integration Workbench.

Liaisons connexes

Autres dossiers du projet

Autres dossiers du projet

Décrit les autres dossiers du projet.

config Utilisez ce dossier pour stocker les fichiers de configuration

supplémentaires (tels que les fichiers de propriétés) que l'administrateur Tivoli Workload Scheduler devra éditer pour l'opération.

keys Ce dossier permet de stocker les fichiers de clés *.jks nécessaires pour se connecter à l'aide de la connexion sécurisée Tivoli Workload Scheduler.

chemin de génération Java

build.xml

Décrit le fichier `build.xml` créé pour un projet d'interface de programme d'application.

Il s'agit d'un fichier de génération ANT standard. Vous pouvez le modifier selon vos besoins.

Liaisons collectées

«Autres dossiers du projet», à la page 7
Décrit les autres dossiers du projet.

Création d'un projet de A à Z

Explique comment crée un projet d'interface de programme d'application depuis le début.

A l'aide d'Integration Workbench, suivez les étapes ci-dessous pour créer intégralement des projets d'interface de programme d'application :

1. A partir d'Integration Workbench, sélectionnez **Help → Help Contents (Aide → Table des matières)**
2. Cliquez sur **Tivoli Workload Scheduler Integration Workbench**, puis sur **Tâches**
3. Développez **Projets d'API Tivoli Workload Scheduler**
4. Les étapes requises pour créer le projet sont répertoriées. Lisez-les pour savoir ce qu'il faut faire.
5. Suivez les instructions indiquées pour créer le projet. Integration Workbench crée une bibliothèque contenant tous les fichiers JAR du produit. Faites appel à vos connaissances des produits Java pour créer tout le codage et l'infrastructure nécessaires pour exécuter l'interface de programme d'application.

Création d'un projet à partir d'un exemple d'interface de programme d'application

Explique comment crée un projet d'interface de programme d'application à partir d'un exemple.

Integration Workbench vous permet de créer des projets basés sur les exemples fournis. Grâce à cette méthode, vous n'avez pas besoin de créer la totalité de la structure de projet de l'interface de programme d'application. Sélectionnez un exemple qui s'approche le plus de vos exigences, puis vous le modifiez en conséquence.

Pour savoir comment utiliser cette fonction, procédez comme suit :

1. A partir d'Integration Workbench, sélectionnez **Help → Help Contents (Aide → Table des matières)**
2. Cliquez sur **Tivoli Workload Scheduler Integration Workbench**, puis sur **Tâches**
3. Développez **Projets d'API Tivoli Workload Scheduler**

4. Sélectionnez **Creating a TWS API project by example**
5. Créez le projet en suivant les instructions. Integration Workbench crée un projet contenant toute l'infrastructure et le code de l'interface de programme d'application sélectionnée.
6. Pour en savoir plus sur l'interface de programme d'application, ouvrez le projet, sélectionnez **Doc** et cliquez deux fois sur **index.htm**

Exemples que vous pouvez sélectionner

Décrit de manière détaillée les exemples de projets de l'interface de programme d'application que vous pouvez utiliser comme modèles.

Les exemples que vous pouvez sélectionner sont les suivants :

AddEventRule
: Utilisation des règles d'événement dans la base de données

MakeQueryJobsOnPlan
: Utilisez les instances de travail de planification dynamique du plan.

MakeQueryOnPlan
: Utilisez les objets du plan.

MakeZOSQueryOnPlan
: Utilisez les objets du plan z/OS.

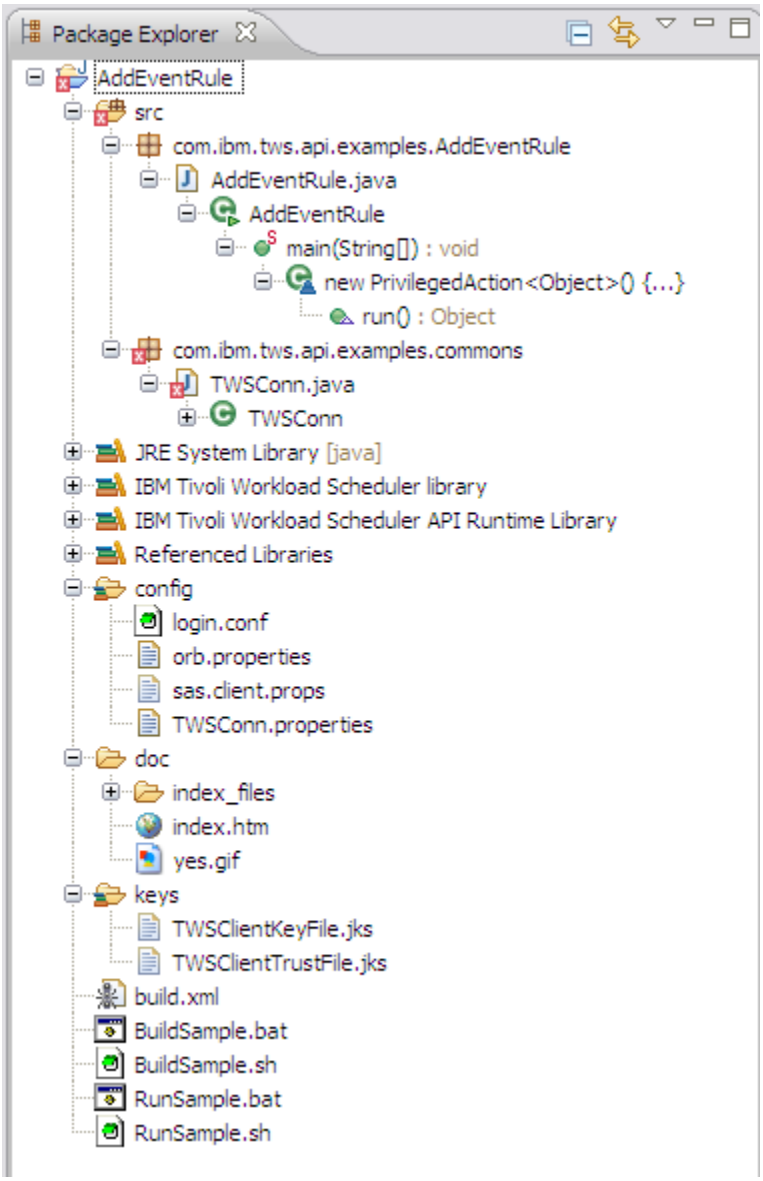
RerunJobInPlan
: Soumettez une instance de flot de travaux dans la réexécution du plan en cours.

SubmitJobInPlan
: Soumettez une instance de flot de travaux dans le plan en cours.

Exemple de projet d'interface de programme d'application Tivoli Workload Scheduler

Fournit un exemple de projet d'interface de programme d'application Tivoli Workload Scheduler.

La figure suivante présente la structure classique d'un nouveau projet d'interface de programme d'application Java Tivoli Workload Scheduler :



La structure présentée s'applique à un projet d'interface de programme d'application Java appelé `AddEventRule`.

Les classes Java contenues dans le dossier `src` sont décrites dans l'arbre source Java (src).

Le chemin de génération Java du projet de plug-in est décrit dans Chemin de génération Java.

Le rôle et le contenu des autres dossiers de projet sont décrits dans les dossiers des autres projets.

Le fichier ANT `build.xml` est décrit dans build.xml.

- Arbre source **Java (src)**
- Chemin de génération **Java**

- **Autres dossiers du projet**
- **build.xml**

Connexion aux produits

Explique comment implémenter une connexion aux produits Tivoli Workload Automation à l'aide de l'interface de programme d'application API.

Pour vous connecter aux produits, vous devez définir les paramètres de connexion de manière à vous connecter au gestionnaire de domaine maître Tivoli Workload Scheduler (dans lequel un connecteur est installé) ou au Tivoli Workload Scheduler for z/OS Connector.

Si vous avez créé votre projet à partir d'un exemple, procédez comme suit suivant le moteur auquel vous vous connectez :

Connexion au gestionnaire de domaine maître Tivoli Workload Scheduler

1. Dans Integration Workbench, sélectionnez votre projet
2. Développez **Config**
3. Editez le fichier TWSConfig.properties pour obtenir les paramètres corrects de votre environnement. Les paramètres sont les suivants :

   ```
   TWSConfig.serverName=
   TWSConfig.serverPort=
   TWSConfig.userID=
   TWSConfig.password=
   TWSConfig.useSecureConnection=
   TWSConfig.serverSecurePort=
   ```

   où les paramètres se présentent comme suit :

   TWSConfig.serverName
   Nom de réseau de l'adresse IP du système dans lequel le gestionnaire de domaine maître Tivoli Workload Scheduler est exécuté (sur lequel un connecteur est installé). La valeur par défaut est "localhost".

   TWSConfig.serverPort
   Port utilisé par le connecteur sur le gestionnaire de domaine maître. Par défaut, la valeur est 31115.

   TWSConfig.userID
   ID utilisateur avec lequel le plug-in doit s'authentifier. La valeur par défaut est "twsuser".

   TWSConfig.password
   Mot de passe de l'ID utilisateur.

   TWSConfig.useSecureConnection
   Entrez "true" pour utiliser une connexion sécurisée.

   TWSConfig.serverSecurePort
   Si `TWSConfig.useSecureConnection` est paramétré sur "true", port sécurisé utilisé par le connecteur sur le gestionnaire de domaine maître.

Connexion au connecteur Tivoli Workload Scheduler for z/OS

1. Dans Integration Workbench, sélectionnez votre projet
2. Développez **Config**

3. Editez le fichier TWSConn.properties pour obtenir les paramètres corrects de votre environnement. Les paramètres sont les suivants :

```
TWSConn.serverName=
TWSConn.serverPort=
TWSConn.userID=
TWSConn.password=
TWSConn.remoteServerName=
```

où les paramètres se présentent comme suit :

TWSConn.serverName
Nom de réseau de l'adresse IP du système dans lequel le connecteur z/OS est installé. La valeur par défaut est "localhost".

TWSConn.serverPort
Port utilisé par le connecteur z/OS. Par défaut, la valeur est 31115.

TWSConn.userID
ID utilisateur avec lequel le plug-in doit s'authentifier. La valeur par défaut est "twsuser".

TWSConn.password
Mot de passe de l'ID utilisateur.

TWSConn.remoteServerName
Nom du moteur z/OS auquel vous souhaitez vous connecter.

Si vous avez créé un projet depuis le début, créez une structure de paramètres de connexion analogue.

Exemples de Tivoli Workload Scheduler

Propose une présentation des exemples possibles d'utilisation de l'interface de programme d'application Java pour Tivoli Workload Scheduler.

Les exemples suivants vous aident à comprendre comment les beans sont utilisés. Les exemples sont annotés à l'aide de commentaires explicatifs. Dans la référence javadoc (voir «Spécification détaillée de l'interface de programme d'application», à la page 5), pour plus d'informations, recherchez les objets utilisés dans les exemples.

Les exemples sont disponibles dans ces groupements :

Utilisation des objets dans la base de données

Fournit des exemples d'utilisation de l'interface de programme d'application Java pour utiliser des objets dans la base de données.

Les exemples suivants indiquent comment employer les classes pour utiliser les objets dans la base de données :

Exemple 1 : Ajout d'un poste de travail à la base de données

```
//Object definition
String wksName = "MYWS";
Workstation wks = new Workstation();
wks.setName(wksName);
wks.setType(WorkstationType.FTA);
wks.setOs(OperatingSystem.UNIX);
```

```
wks.setAutoLink(true);
wks.setNodeName("node.ibm.com");
wks.setSecurityLevel(SecurityLevel.NONE);


ConnModel myModel;
//Get an instance of ConnModel interface...
...

//Add the object
try
{
   myModel.addTWSObject(wks, null);
}
catch (ConnException e)
{
   //Do something to recover...
}
```

Exemple 2 : Récupération d'un poste de travail de la base de données

```
Workstation wksRead = new Workstation();
//Get the same workstation from the DB
try
{
   wksRead = (Workstation) myModel.getTWSObject(Workstation.class,
             new FlowTargetKey(wksName), false, null);
}
catch (ConnException e)
{
   //Do something to recover...
 }
```

Exemple 3 : Suppression d'un poste de travail de la base de données

```
//Remove a workstation from the DB
 try
{
   myModel.removeTWSObject(Workstation.class, wksRead.getId(), null);
}
catch (ConnException exc)
{
   //Do something to recover...
}
```

Utilisation des objets dans le plan

Fournit des exemples d'utilisation de l'interface de programme d'application Java pour utiliser des objets dans le plan.

Les exemples suivants indiquent comment employer les classes pour utiliser des objets dans le plan :

Exemple 4 : Soumission d'une instance de flot de travaux dans le plan en cours

Cette procédure requiert les principales étapes suivantes :

1. Obtenir la définition de flot de travaux requise de la base de données

   ```
   ConnPlan myPlan;
   //Get an instance of ConnPlan interface...
   ...
   ```

```
String alias = "SBJBF1_1";
JobStream js = null;
JobStreamInPlan jsip = null;
//If you already have a JobStream in the DB with Identifier jsDbID...
try
{
   //get it from the DB
   js = (JobStream)(myPlan.getTWSObject(JobStream.class,jsDbID,false,null));
```

2. Transform it into a JobStreamInPlan:

```
//Transform it in a JobStreamInPlan.
//TODAY est une variable représentant l'heure planifiée
   jsip = myPlan.makeJobStreamInPlan(jsDbID, TODAY, alias, null);
}
catch (ConnException e)
{
              //Something went wrong...
}
catch (ConnEngineNotMasterException e)
{
   //Since the makeJobStreamInPlan is available also on FTAs
   //(it's on the Plan interface), an exception must be thrown
   //if it is called on an engine that is not the master
}
```

3. Add the JobStreamInPlan to the plan:

```
List idList = new ArrayList();
try
{
   //Add the job stream to the plan.
   //This method returns a list of Identifiers because the job stream can be
   //defined on a Workstation class, so you have an ID for each workstation
   //of the class
   idList = (ArrayList)myPlan.addJobStreamInstance(jsip, null);
}
catch (ConnException e)
{
   //...
}
catch (ConnEngineNotMasterException e)
{
   //...
}
```

Exemple 5 : Exécution une requête sur le plan

L'exemple suivant répertorie les cinq premiers travaux commençant par la lettre A :

```
String nameFilter = "A*";
int howMany = 5;

QueryFilter qf = new QueryFilter();
qf.setFilter(JobInPlanFilters.JOB_NAME, nameFilter);

QueryResult qr = null;
 try
{
          qr = myPlan.queryPlanObject(JobInPlan.class, qf, howMany, null);
}
catch (ConnException e)
{
   //...
}
```

Utilisation des règles d'événement dans la base de données

Fournit des exemples d'utilisation de l'interface de programme d'application Java pour utiliser des règles d'événement dans la base de données.

Les exemples suivants indiquent comment employer les classes pour utiliser des règles d'événement dans la base de données :

Exemple 6 : Ajout d'une règle d'événement à la base de données

Procédez comme suit :

1. Définissez la règle d'événement :

```
String eventRuleName = "SampleEventRule";
String eventRuleDescription =
   "Define Event Rule; test MessageLoggerPlugIn and TWSObjectsMonitorPlugIn";
Date today = new Date(System.currentTimeMillis());
Date tomorrow = new Date(System.currentTimeMillis() + 86400000L);

//EventRule definition

EventRule er = new EventRule();
er.setName(eventRuleName);
er.setDescription(eventRuleDescription);
er.setRuleType(EventRuleType.FILTER);
er.setDraft(false);
er.setValidFrom(today);
er.setValidTo(tomorrow);
```

2. Définissez la condition de l'événement. Dans cet exemple, la condition est la soumission d'un travail :

```
EventCondition evCond = new EventCondition();
evCond.setPluginName(TWSObjectsMonitorPlugIn.PLUGIN_NAME);
evCond.setEventType(JobUtil.EVENT_JOB_SUBMIT);
```

3. Définissez les conditions que la condition d'événement doit satisfaire pour déclencher l'action associée (le prédicat de filtrage) :

```
String filterPred =   "<attributeFilter name=\"JobStreamWorkstation\"
                                        operator=\"eq\">"
   + "<value>MYWS</value>"
   + "</attributeFilter>"

   + "<attributeFilter name=\"JobStreamName\" operator=\"eq\">"
   + "<value>JS1</value>"
   + "</attributeFilter>"

   + "<attributeFilter name=\"JobName\" operator=\"eq\">"
   + "<value>JOB1</value>"
   + "</attributeFilter>"

   + "<attributeFilter name=\"Workstation\" operator=\"eq\">"
   + "<value>MYHOST</value>"
   + "</attributeFilter>"

   + "<attributeFilter name=\"Priority\" operator=\"range\">"
   + "<value>10</value>"
   + "<value>30</value>"
   + "</attributeFilter>"

   + "<attributeFilter name=\"Monitored\" operator=\"eq\">"
   + "<value>TRUE</value>"
   + "</attributeFilter>"

   + "<attributeFilter name=\"EstimatedDuration\" operator=\"ge\">"
   + "<value>400</value>"
   + "</attributeFilter>"
```

```
    + "<attributeFilter name=\"Login\" operator=\"eq\">"
    + "<value>TWSUser</value>"
    + "</attributeFilter>"

    + "<attributeFilter name=\"EveryFrequency\" operator=\"ge\">"
    + "<value>400</value>"
    + "</attributeFilter>";
```

4. Renseignez la condition d'événement :

```
evCond.setFilteringPredicate(filterPred);
```

5. Ajoutez la condition d'événement à la règle d'événement :

```
er.getTriggerEvents().add(evCond);
```

6. Définissez l'action associée. Dans cet exemple, l'action associée consigne un message dans la base de données :

```
RuleAction action = new RuleAction();
action.setPluginName(MessageLoggerPlugIn.PLUGIN_NAME);
action.setActionType(MessageLoggerPlugInConstants.ACTION_TYPE_MESSAGE_LOG);
action.setDescription("Adding the Message logger Plugin");
action.setResponseType(RuleResponseType.ON_DETECTION);
```

7. Définissez la valeur du paramètre de l'action associée :

```
Map parameterMap = new HashMap();
parameterMap.put(MessageLoggerPlugInConstants.MESSAGE, "message");
parameterMap.put(MessageLoggerPlugInConstants.OBJECT_KEY, "object key");
```

8. Renseignez l'action associée :

```
action.getParameterMap().putAll(parameterMap);
```

9. Ajoutez l'action associée à la règle d'événement :

```
er.getActions().add(action);
```

10. Ajoutez la règle d'événement à l'interface ConnModel :

```
ConnModel myModel = null;
//Get an instance of ConnModel interface...
//...

//Add the object

Identifier erId = null;
try
{
 erId = myModel.addTWSObject(er, null);
}
catch (ConnException e)
{
 //Do something to recover...
}
```

Exemple 7 : Récupération d'une règle d'événement de la base de données par ID

Procédez comme suit :

1. Obtenez l'ID règle d'événement à récupérer par tout moyen adapté à votre interface
2. Récupérez la règle d'événement :

```
EventRule eRuleRead = new EventRule();
try
{
 eRuleRead =
      (EventRule) myModel.getTWSObject(EventRule.class, erId, false, null);
}
```

```
catch (ConnException e)
{
 //Do something to recover...
}
```

Exemple 8 : Récupération d'une règle d'événement de la base de données par clé (nom)

Procédez comme suit :

1. Obtenez la clé (nom) de règle d'événement à récupérer par tout moyen adapté à votre interface
2. Récupérez la règle d'événement :

```
EventRule eRuleRead = new EventRule();
try
{
 eRuleRead =
    (EventRule) myModel.getTWSObject(EventRule.class,
                                     new EventRuleKey(eventRuleName), false, null);
}
catch (ConnException e)
{
 //Do something to recover...
}
```

Exemple 9 : Suppression d'une règle d'événement de la base de données par ID

Procédez comme suit :

1. Récupérez la règle d'événement à supprimer par son ID, comme indiqué dans l'exemple 7.1
2. Si la règle d'événement a été récupérée avec succès, supprimez-la :

```
{
 myModel.removeTWSObject(EventRule.class, eRuleRead.getId(), null);
}
catch (ConnException exc)
{
 //Do something to recover...
}
```

Exemple 10 : Suppression d'une règle d'événement de la base de données par clé (nom)

Procédez comme suit :

1. Récupérez la règle d'événement à supprimer par sa clé, comme indiqué dans l'exemple 7.2
2. Si la règle d'événement a été récupérée avec succès, supprimez-la :

```
{
 myModel.removeTWSObject(EventRule.class,
                                       new EventRuleKey(eventRuleName), null);
}
catch (ConnException exc)
{
 //Do something to recover...
}
```

Exemples de Tivoli Workload Scheduler for z/OS

Propose une présentation des exemples possibles d'utilisation de l'interface de programme d'application Java pour Tivoli Workload Scheduler for z/OS.

Les exemples suivants vous aident à comprendre comment les beans sont utilisés dans l'environnement Tivoli Workload Scheduler for z/OS :

Exemple 1 : Ajout d'un poste de travail à la base de données

```
// Create the connection to the server
TWSZConn connection = new TWSZConn();
// Get an instance of the interface ZConnModel
final ZConnModel model = connection.getModelBean();

// Define the workstation properties
String wksName = "CPU1";
String wksDescription = "Added by API";
String wksPrintoutRouting = "SYSOUT";
Workstation wks = new Workstation();
wks.setName(wksName);
wks.setDescription(wksDescription);
wks.setType(WorkstationType.COMPUTER);
wks.setReportingAttribute(WorkstationReportingAttribute.AUTOMATIC);
WorkstationZOSAttributes wksAttr = new WorkstationZOSAttributes();
wksAttr.setDefaultTransportTime(3600);
wksAttr.setDefaultJobDuration(60);
wksAttr.setPrintoutRouting(wksPrintoutRouting);
wksAttr.setStartedTaskSupported(true);
wks.setZosAttributes(wksAttr);

try {
 Context context = new Context();
 // Add the workstation to the database
 model.addTWSObject(wks, context);
}
catch (ConnException e) {
 // Do something to recover
}
```

Exemple 2 : Ajout d'un flot de travaux (application) à la base de données

```
// Create the connection to the server
TWSZConn connection = new TWSZConn();
// Get an instance of the interface ZConnModel
final ZConnModel model = connection.getModelBean();

final static Date TODAY =
new Date((System.currentTimeMillis()/86400000L) * 86400000L);

// Define the job stream properties
String jsName = "APPL";
String jsOwnerName = "API";
JobStream js = new JobStream();
js.setName(jsName);
js.setOwnerName(jsOwnerName);
js.setValidFrom(TODAY);
js.setPriority(5);
// Define a JCL job to add to the job stream
String jobName = "1";
String wksName = "CPU1";
String jclName = "MYJCL";
Job jclJob = new Job();
jclJob.setName(jobName);
jclJob.setEstimatedDuration(1000);
```

```
jclJob.setPriority(-1);
ZOSJobDefinition jobDef = new ZOSJobDefinition();
jobDef.setFlowTargetKey(new FlowTargetKey(wksName));
jobDef.setTaskType(TaskTypes.ZOS_JOB_TASK);
jobDef.setJclName(jclName);
jclJob.setJobDefinition(jobDef);
// Add the JCL job to the job stream
js.getJobs().add(jclJob);
// Define a Printer job to add to the job stream
jobName = "2";
wksName = "PRNT";
Job printerJob = new Job();
printerJob.setName(jobName);
printerJob.setEstimatedDuration(1000);
printerJob.setPriority(-1);
jobDef = new ZOSJobDefinition();
jobDef.setFlowTargetKey(new FlowTargetKey(wksName));
jobDef.setTaskType(TaskTypes.ZOS_ PRINTER_TASK);
jobDef.setJclName(jclName);
jobDef.setLimitForFeedback(102);
printerJob.setJobDefinition(jobDef);
// Add to the Printer job the dependency from the JCL job
List printerJobDeps = printerJob.getInternalDependencies();
InternalDependency depFromJclJob =
new InternalDependency(null, (JobKey)jclJob.getKey());
printerJobDeps.add(depFromJclJob);
// Add the Printer job to the job stream
js.getJobs().add(printerJob);

try {
 Context context = new Context();
 // Add the job stream to the database
```

Exemple 3 : Modification d'un flot de travaux (application) dans la base de données

```
// Create the connection to the server
TWSZConn connection = new TWSZConn();
// Get an instance of the interface ZConnModel
final ZConnModel model = connection.getModelBean();

final static Date TODAY =
new Date((System.currentTimeMillis()/86400000L) * 86400000L);

final static Date TOMORROW =
new Date(((System.currentTimeMillis()/86400000L) * 86400000L) + 86400000L);

final static long HOUR = 3600000;

// Make the job stream key
String jsName = "APPL";
JobStreamKey jsKey = new JobStreamKey(jsName, TODAY, false, false);

try {
 Context context = new Context();
 // Get the job stream by its key
 JobStream js =
         (JobStream)model.getTWSObject(JobStream.class, jsKey, false, context);

 // Define a run cycle to add to the job stream
  String rcName = "RCRULE";
  String rcDescription = "Added by API";
  RunCycle rcRule = new RunCycle();
  rcRule.setName(rcName);
  rcRule.setDescription(rcDescription);
  rcRule.setType(RunCycleType.RULE);
  rcRule.setValidFrom(TODAY);
```

```
  rcRule.setValidTo(TOMORROW);
  rcRule.getTimeRestrictions().setStartOffset(10*HOUR);
  rcRule.getTimeRestrictions().setDeadlineOffset(12*HOUR + 24* HOUR);
  rcRule.setFreeDaysRule(FreeDaysRule.DO_NOT_SELECT);
  rcRule.setICalendar("ADRULE ONLY(001 003) LAST(001 002) DAY(DAY
MONDAY THURSDAY) MONTH(FEBRUARY APRIL JUNE SEPTEMBER NOVEMBER) YEAR  ");

  // Add the run cycle to the job stream
  js.getRunCycles().add(rcRule);
  // Modify the job stream in the database
  Identifier jsId = model.setTWSObject(js, true, true, context);
}
catch (ConnException e) {
 // Do something to recover
}
```

Exemple 4 : Ajout d'une ressource spéciale à la base de données

```
// Create the connection to the server
TWSZConn connection = new TWSZConn();
// Get an instance of the interface ZConnModel
final ZConnModel model = connection.getModelBean();

final static long HOUR = 3600000;

// Define the resource properties
String resName = "RES";
String resDescription = "Added by API";
String wksName1 = "CPU1";
String wksName2 = "CPU2";
Resource res = new Resource();
res.setName(resName);
res.setDescription(resDescription);
FlowTargetKey wksKey1 = new FlowTargetKey(wksName1);
FlowTargetKey wksKey2 = new FlowTargetKey(wksName2);
res.getConnectedWorkstationLinks().add(new WorkstationLink(wksKey1));
res.getConnectedWorkstationLinks().add(new WorkstationLink(wksKey2));
ResourceBaseConstraints resCon = new ResourceBaseConstraints();
resCon.setQuantity(30);
resCon.setUsedFor(ResourceUsage.CONTROL);
resCon.setActionOnError(ResourceActionOnError.KEEP);
resCon.setAvailable(YesNoDefaultOption.NO);
res.setDefaultConstraints(resCon);
ResourceAvailabilityInterval resInt =
new ResourceAvailabilityInterval();
resInt.setIntervalValidityDayOfWeek(Calendar.MONDAY);
resInt.setIntervalStartTime(10*HOUR);
resInt.setIntervalEndTime(21*HOUR);
resInt.setQuantity(20);
resInt.setAvailable(YesNoDefaultOption.YES);
res.getResourceAvailabilityIntervals().add(resInt);

try {
 Context context = new Context();
 // Add the resource to the database
 model.addTWSObject(res, context);
}
catch (ConnException e) {
 // Do something to recover
}
```

Exemple 5 : Ajout d'un flot de travaux au plan après modification de son contenu

Le programme utilise des API Java pour modifier un flot de travaux et effectuer une substitution de variables avant de le soumettre au plan.

Le programme suivant ajoute un flot de travaux STREAM1 au plan. Avant d'effectuer cette tâche, le programme effectue les opérations suivantes :

1. Ajoute deux travaux à STREAM1 après avoir extrait leurs attributs de deux travaux provenant des flots de travaux STREAM2 et STREAM3.
2. Configure un certain nombre de variables paramétrables JCL avant de soumettre le flot de travaux STREAM1.
3. Ajoute/modifie les attributs propres aux travaux avant de libérer les travaux dans le plan.

Après avoir importé l'ensemble des classes et des objets nécessaires, établi une connexion avec le planificateur et déclaré les variables requises, le programme :

- Extrait un travail du flot de travaux STREAM2 dans la base de données, extrait et place ses propriétés dans un conteneur ZosJobInfo appelé jZos1 et définit d'autres informations, telles que l'heure d'arrivée d'entrée, le poste de travail et les ressources associées.
- Répète ces opérations pour le travail doté du numéro 10 dans le flot de travaux STREAM3 et stocke les propriétés dans un conteneur ZosJobInfo appelé jZos2.
- Ajoute deux travaux au flot de travaux STREAM1 et modifie leurs propriétés pour définir leurs numéros (qui représentent leur ID au sein du flot de travaux) et leurs noms.
- Définit les premières variables JCL générales pour les quatre travaux du flot de travaux STREAM1 (à l'aide de l'API variablesToBeSubstituted), puis des variables spécifiques pour chaque travail (à l'aide de l'API jobVariablesToBeSubstituted).
- Ajoute le flot de travaux STREAM1 au plan.
- Modifie une dernière fois le flot de travaux pour ajouter des propriétés supplémentaires à l'un des travaux (le nom de travail étendu et une ressource spéciale, par exemple), puis libère le travail.

```
package com.ibm;

import java.rmi.RemoteException;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.List;


import com.ibm.tws.conn.exception.ConnEngineNotMasterException;
import com.ibm.tws.conn.exception.ConnException;
import com.ibm.tws.conn.exception.ConnLockingException;
import com.ibm.tws.conn.exception.ConnNotFoundException;
import com.ibm.tws.conn.exception.ConnSecurityException;
import com.ibm.tws.conn.exception.ConnTransportException;
import com.ibm.tws.conn.exception.ConnValidationException;
import com.ibm.tws.conn.util.Context;
import com.ibm.tws.conn.util.QueryResult;
import com.ibm.tws.objects.filter.JobStreamFilters;
import com.ibm.tws.objects.filter.QueryFilter;
import com.ibm.tws.objects.filter.WorkstationFilters;
import com.ibm.tws.objects.model.Job;
import com.ibm.tws.objects.model.JobStream;
import com.ibm.tws.objects.model.JobStreamHeader;
import com.ibm.tws.objects.model.ResourceDependency;
import com.ibm.tws.objects.model.Workstation;
import com.ibm.tws.objects.model.WorkstationHeader;
import com.ibm.tws.objects.model.ZOSJobDefinition;
import com.ibm.tws.objects.plan.JobInPlan;
import com.ibm.tws.objects.plan.JobStreamInPlan;
import com.ibm.tws.objects.plan.ResourceDependencyInPlan;
import com.ibm.tws.objects.plan.ResourceInPlanKey;
import com.ibm.tws.objects.plan.WorkstationInPlanKey;
import com.ibm.tws.objects.plan.ZOSJobDefinitionInPlan;
import com.ibm.tws.objects.plan.types.DependenciesResolutionOption;
```

```
import com.ibm.tws.objects.plan.types.JobInPlanZOSAttributes;
import com.ibm.tws.objects.plan.utils.ZosJobInfo;
import com.ibm.tws.objects.types.Identifier;
import com.ibm.tws.zconn.model.ZConnModel;
import com.ibm.tws.zconn.plan.ZConnPlan;
import com.ibm.tws.zconn.plan.dao.impl.util.ResourceInPlanHelper;
import com.ibm.tws.zconn.plan.dao.impl.util.WorkstationInPlanHelper;

@SuppressWarnings("restriction")
public class MainWitRes {

 public static void main(String[] args) {
  //Create a connection to the server
  TWSZConn connection=new TWSZConn();

  //Get Model or Plan Bean
  final ZConnModel model=connection.getModelBean();
  final ZConnPlan plan=connection.getPlanBean();
  final Context context=new Context();

  com.ibm.websphere.security.auth.WSSubject.doAs
    (connection.getSubject(), new java.security.PrivilegedAction<Object>() {
   /* (non-Javadoc)
    * @see java.security.PrivilegedAction#run()
    */
   @SuppressWarnings("unchecked")
   public Object run() {
    //Variable Declaration
    HashMap<String, List<Integer>> dependencyToDelete = new HashMap<String, List<Integer>>();
    HashMap<String, List<Integer>> dependencyToAdd = new HashMap<String, List<Integer>>();
    HashMap<Integer, String[][]> jobVariablesToBeSubstituted = new HashMap<Integer, String[][]>();
    List<ZosJobInfo> jobsToDelete=new ArrayList<ZosJobInfo>();
    List<ZosJobInfo>jobsToAdd = new ArrayList<ZosJobInfo>();
    List<Identifier> identifierList=null;
    List<Integer> successorList=new ArrayList<>();
    List<JobStreamHeader> jobStreamHeaderList;
    List<WorkstationHeader> workstationHeaderList;
    List<ZosJobInfo>jobsToModify = new ArrayList<ZosJobInfo>();
    List<ResourceDependencyInPlan> resourceDependencyInPlanList;
    Date startTime=new Date();
    Date deadlineTime=new Date();
    Long time1,time2;
    int seconds,minutes,hours;
    int priority=5;
    int jobNum=0;
    Integer succ;
    String [][] variablesMap1 = new String [3][2];
    String[][] variablesToBeSubstituted = new String [1][2];
    String [][] variablesMap2 = new String [2][2];
    String resourceName;
    String authorityGroup=null;
    String JSName="STREAM1";
    String description = "";
    String group=null;
    String owner=null;
    String ownerDescription=null;
    String variableTable = "STREAM1";
    String jobStreamName1="STREAM2";
    String jobStreamName2="STREAM3";
    boolean holdAll=true;
    JobStream jobStream1=null;
    JobStream jobStream2=null;
    Job jobFM1=null;
    Job jobFM2=null;
    JobStreamInPlan jobStreamInPlan=null;
    ZOSJobDefinition zosJobDefinition1=null;
    ZOSJobDefinition zosJobDefinition2=null;
    ZosJobInfo zosJobInfo1=new ZosJobInfo();
    ZosJobInfo zosJobInfo2=new ZosJobInfo();
    ZosJobInfo zosJobInfo3=new ZosJobInfo();
    ZosJobInfo zosJobInfo4=new ZosJobInfo();
    QueryFilter queryFilter;
    QueryResult queryResult;
    Calendar calendar1,calendar2;
    Workstation workstation = null;
    ZOSJobDefinitionInPlan jobDefinitionInPlan;
    ResourceDependencyInPlan resourceDependencyInPlan;
    ResourceInPlanKey resourceInPlanKey;
    WorkstationInPlanKey workstationInPlanKey;
```

```
JobInPlanZOSAttributes jobInPlanZosAtt;
     //Getting jobStream="Stream2" from the DB, first the header then the full jobStream
queryFilter=new QueryFilter();
queryFilter.setFilter(JobStreamFilters.JOB_STREAM_NAME, jobStreamName1);
try {
    queryResult=model.queryTWSObject(JobStream.class, queryFilter, 1, context);
   jobStreamHeaderList=(List<JobStreamHeader>) queryResult.getList();
   if(jobStreamHeaderList.size()>0){
   jobStream1=(JobStream) model.getTWSObject(JobStream.class,jobStreamHeaderList.get(0).getKey() , false, context);
   }
     } catch (ConnTransportException e) {
 //TODO Auto-generated catch block
 e.printStackTrace();
} catch (ConnValidationException e) {
 //TODO Auto-generated catch block
 e.printStackTrace();
} catch (ConnSecurityException e) {
 //TODO Auto-generated catch block
 e.printStackTrace();
} catch (ConnException e) {
 //TODO Auto-generated catch block
 e.printStackTrace();
} catch (RemoteException e) {
 //TODO Auto-generated catch block
 e.printStackTrace();
}

//Getting the first job from Stream2 to add to STREAM1
if(jobStream1.getJobs().size()>0){
 jobFM1=(Job) jobStream1.getJobs().get(0);
 zosJobDefinition1=(ZOSJobDefinition) jobFM1.getJobDefinition();
}
     //Creation of ZosJobInfo for the first job
ZosJobInfo jZos1=new ZosJobInfo();

//Setting the inputArrivalTime
calendar1=Calendar.getInstance();
time1=jobFM1.getTimeRestrictions().getStartOffset();
if(time1!=null && time1>0){
 seconds=(int) (time1 / 1000) % 60 ;
 minutes=(int) ((time1 / (1000*60)) % 60);
 hours=(int) ((time1 / (1000*60*60)) % 24);
 calendar1.set(Calendar.HOUR,hours);
 calendar1.set(Calendar.MINUTE, minutes);
 calendar1.set(Calendar.SECOND, seconds);
 jZos1.setInputArrivalTime(calendar1);
}
//Setting the zosJobInfo with data in zosJobDefinition and job (&timeRestriction)
     String workstationN=zosJobDefinition1.getFlowTargetKey().getName();
 jZos1.setJobName(zosJobDefinition1.getJclName());
 jZos1.setTextDescription(jobFM1.getDescription());
 jZos1.setWorkstationName(workstationN);
  jZos1.setDuration(jobFM1.getEstimatedDuration());
 jZos1.setJobNumber(44);
 jZos1.setAutoSubmit(zosJobDefinition1.getAutoSubmit());
 jZos1.setTaskType(zosJobDefinition1.getTaskType());
 jZos1.setTimeDependent(jobFM1.getTimeRestrictions().isTimeDependent());
 jZos1.setCentralizedScript(zosJobDefinition1.getHasCentralizedScript());

 //Getting the workstation from the DB -> to get the type associated, first get the header and then the full workstation.
 queryFilter.setFilter(WorkstationFilters.WORKSTATION_NAME, workstationN);
 try {
       queryResult=model.queryTWSObject(Workstation.class, queryFilter, 1, context);
       workstationHeaderList=(List<WorkstationHeader>) queryResult.getList();
       if(workstationHeaderList.size()>0){
         workstation=(Workstation) model.getTWSObject(Workstation.class,workstationHeaderList.get(0).getKey() , false, context);
   }
     } catch (ConnTransportException e) {
 //TODO Auto-generated catch block
 e.printStackTrace();
     } catch (ConnValidationException e) {
 //TODO Auto-generated catch block
 e.printStackTrace();
     } catch (ConnSecurityException e) {
 //TODO Auto-generated catch block
 e.printStackTrace();
     } catch (ConnException e) {
 //TODO Auto-generated catch block
 e.printStackTrace();
```

```
    } catch (RemoteException e) {
 //TODO Auto-generated catch block
 e.printStackTrace();
    }

jZos1.setWorkstationType(workstation.getType());

//Setting the Resource value
for(ResourceDependency resourceD:(List<ResourceDependency>) jobFM1.getResourceDependencies()){
 String rName=resourceD.getResourceKey().getName();
 if(rName!=null && rName.compareToIgnoreCase("Resource1")==0){
  jZos1.setR1(resourceD.getQuantity());
 }
 if(rName!=null && rName.compareToIgnoreCase("Resource2")==0){
  jZos1.setR2(resourceD.getQuantity());
 }
 if(rName!=null && rName.compareToIgnoreCase("ParallelServers")==0){
  jZos1.setParallelServer(resourceD.getQuantity());
 }
}

//Getting jobStream="STREAM3" from the Db, first the header then the full jobStream
queryFilter.setFilter(JobStreamFilters.JOB_STREAM_NAME, jobStreamName2);
    try {
queryResult=model.queryTWSObject(JobStream.class, queryFilter, 1, context);
jobStreamHeaderList=(List<JobStreamHeader>) queryResult.getList();
 if(jobStreamHeaderList.size()>0){
  jobStream2=(JobStream) model.getTWSObject(JobStream.class,jobStreamHeaderList.get(0).getKey() , false, context);
 }
} catch (ConnTransportException e) {
 //TODO Auto-generated catch block
 e.printStackTrace();
} catch (ConnValidationException e) {
 //TODO Auto-generated catch block
 e.printStackTrace();
} catch (ConnSecurityException e) {
 //TODO Auto-generated catch block
 e.printStackTrace();
} catch (ConnException e) {
 //TODO Auto-generated catch block
 e.printStackTrace();
} catch (RemoteException e) {
 //TODO Auto-generated catch block
 e.printStackTrace();
}

//Starting with second job to add to STREAM1
//Getting the job with number=10 from STREAM3
for(Job job:(List<Job>)jobStream2.getJobs()){
 if(job.getName().compareTo("10")==0){
  jobFM2=job;
  zosJobDefinition2=(ZOSJobDefinition) jobFM2.getJobDefinition();
 }
}
//Creation of ZosJobInfo for the second job
ZosJobInfo jZos2=new ZosJobInfo();

//Setting the inputArrivalTime
calendar2=Calendar.getInstance();
time2=jobFM2.getTimeRestrictions().getStartOffset();
if(time2!=null && time2>0){
 seconds=(int) (time2 / 1000) % 60 ;
 minutes=(int) ((time2 / (1000*60)) % 60);
 hours=(int) ((time2 / (1000*60*60)) % 24);
 calendar2.set(Calendar.HOUR,hours);
 calendar2.set(Calendar.MINUTE, minutes);
 calendar2.set(Calendar.SECOND, seconds);
 jZos2.setInputArrivalTime(calendar2);
}

    //Setting the zosJobInfo for the second job
//with data in zosJobDefinition and job (@timeRestriction)
workstationN=zosJobDefinition2.getFlowTargetKey().getName();
jZos2.setJobName(zosJobDefinition2.getJclName());
jZos2.setTextDescription(jobFM2.getDescription());
jZos2.setWorkstationName(workstationN);
jZos2.setDuration(jobFM2.getEstimatedDuration());
jZos2.setJobNumber(45);
jZos2.setAutoSubmit(zosJobDefinition2.getAutoSubmit());
```

```
jZos2.setTaskType(zosJobDefinition2.getTaskType());
jZos2.setTimeDependent(jobFM2.getTimeRestrictions().isTimeDependent());
jZos2.setCentralizedScript(zosJobDefinition2.getHasCentralizedScript());

//Getting the workstation -> for get the type associated. Header then full workstation.
queryFilter.setFilter(WorkstationFilters.WORKSTATION_NAME, workstationN);
try {
 queryResult=model.queryTWSObject(Workstation.class, queryFilter, 1, context);
 workstationHeaderList=(List<WorkstationHeader>) queryResult.getList();
 if(workstationHeaderList.size()>0){
  workstation=(Workstation) model.getTWSObject(Workstation.class,workstationHeaderList.get(0).getKey() , false, context);
 }
} catch (ConnTransportException e) {
 //TODO Auto-generated catch block
 e.printStackTrace();
} catch (ConnValidationException e) {
 //TODO Auto-generated catch block
 e.printStackTrace();
} catch (ConnSecurityException e) {
 //TODO Auto-generated catch block
 e.printStackTrace();
} catch (ConnException e) {
 //TODO Auto-generated catch block
 e.printStackTrace();
} catch (RemoteException e) {
 //TODO Auto-generated catch block
 e.printStackTrace();
}

jZos2.setWorkstationType(workstation.getType());

     //Setting the Resource value
for(ResourceDependency resourceD:(List<ResourceDependency>) jobFM2.getResourceDependencies()){
 String rName=resourceD.getResourceKey().getName();
 if(rName!=null && rName.compareToIgnoreCase("Resource1")==0){
  jZos2.setR1(resourceD.getQuantity());
 }
 if(rName!=null && rName.compareToIgnoreCase("Resource2")==0){
  jZos2.setR2(resourceD.getQuantity());
 }
 if(rName!=null && rName.compareToIgnoreCase("ParallelServers")==0){
  jZos2.setParallelServer(resourceD.getQuantity());
 }
}

//Setting predecessor and successor for the jobs.
//Otherwise we cannot add them to STREAM1

succ= new Integer(jZos1.getJobNumber());
successorList.add(succ);
succ=new Integer(jZos2.getJobNumber());
successorList.add(succ);
dependencyToAdd.put(String.valueOf(1),successorList);

//Adding the zosJobInfo to STREAM1
jobsToAdd.add(jZos1);
jobsToAdd.add(jZos2);

//Example of how to modify a job in STREAM1

zosJobInfo1.setJobNumber(5);
zosJobInfo1.setJobName("EXEC1");
jobsToModify.add(zosJobInfo1);

zosJobInfo2.setJobNumber(10);
zosJobInfo2.setJobName("EXEC2");
jobsToModify.add(zosJobInfo2);

zosJobInfo4.setJobNumber(15);
zosJobInfo4.setJobName("EXEC3");
jobsToModify.add(zosJobInfo4);

zosJobInfo3.setJobNumber(20);
zosJobInfo3.setJobName("EXEC4");
jobsToModify.add(zosJobInfo3);

     //Default JCL variables values (for all jobs)

variablesToBeSubstituted [0][0] = "VAR1";
```

```
variablesToBeSubstituted [0][1] = "ValVar1ForExec2And3";


DependenciesResolutionOption resolutionOption = DependenciesResolutionOption.RESOLUTION_ALL;

//Job level JCL variables values

variablesMap1 [0][0] = "VAR1";
variablesMap1 [0][1] = "ValVar1ForExec1";
variablesMap1 [1][0] = "VAR2";
variablesMap1 [1][1] = "ValVar2ForExec1";
variablesMap1 [2][0] = "VAR3";
variablesMap1 [2][1] = "ValVar3ForExec1";
jobNum = 5;


jobVariablesToBeSubstituted.put(jobNum, variablesMap1);


variablesMap2 [0][0] = "VAR2";
variablesMap2 [0][1] = "ValVar2ForExec2";
variablesMap2 [1][0] = "VAR4";
variablesMap2 [1][1] = "ValVar4ForExec2";
jobNum = 10;

jobVariablesToBeSubstituted.put(jobNum, variablesMap2);

String [][] variablesMap3 = new String [2][2];
variablesMap3 [0][0] = "VAR2";
variablesMap3 [0][1] = "ValVar2ForExec3";
variablesMap3 [1][0] = "VAR4";
variablesMap3 [1][1] = "ValVar4ForExec3";
jobNum = 15;

jobVariablesToBeSubstituted.put(jobNum, variablesMap3);

String [][] variablesMap4 = new String [4][2];
variablesMap4 [0][0] = "VAR3";
variablesMap4 [0][1] = "ValVar3ForExec4";
variablesMap4 [1][0] = "VAR4";
variablesMap4 [1][1] = "ValVar4ForExec4";
variablesMap4 [2][0] = "VAR5";
variablesMap4 [2][1] = "ValVar5ForExec4";
variablesMap4 [3][0] = "VAR6";
variablesMap4 [3][1] = "ValVar6ForExec4";
jobNum = 20;

jobVariablesToBeSubstituted.put(jobNum, variablesMap4);

     //Add STREAM1 in the plan with all the modified and new jobs (zosJobInfo)

try {
 identifierList = plan.editAddJobStreamInstanceWithVariableSubstitution(
   JSName, startTime, deadlineTime, priority, description, group, owner,
   ownerDescription, variableTable, jobsToDelete, jobsToAdd, jobsToModify,
   dependencyToDelete, dependencyToAdd, variablesToBeSubstituted, authorityGroup,
   holdAll, resolutionOption, jobVariablesToBeSubstituted, context);
 Identifier jsid = identifierList.get(0);

 // get the jobStream (STREAM1) from the plan
 jobStreamInPlan = (JobStreamInPlan) plan.getPlanObject(JobStreamInPlan.class, jsid, context);
} catch (ConnLockingException e) {
 //TODO Auto-generated catch block
 e.printStackTrace();
} catch (ConnNotFoundException e) {
 //TODO Auto-generated catch block
 e.printStackTrace();
} catch (ConnSecurityException e) {
 //TODO Auto-generated catch block
 e.printStackTrace();
} catch (ConnTransportException e) {
 //TODO Auto-generated catch block
 e.printStackTrace();
} catch (ConnValidationException e) {
 //TODO Auto-generated catch block
 e.printStackTrace();
} catch (ConnEngineNotMasterException e) {
 //TODO Auto-generated catch block
 e.printStackTrace();
```

```
} catch (ConnException e) {
 //TODO Auto-generated catch block
 e.printStackTrace();
} catch (RemoteException e) {
 //TODO Auto-generated catch block
 e.printStackTrace();
}

     //Modify the job in JobStreamInPlan to add data missing in ZosJobInfo
//Like extendedName(job 44) and specialResource(job=45)
for(JobInPlan jobInPlan:(List<JobInPlan>) jobStreamInPlan.getJobs()){
 jobDefinitionInPlan=(ZOSJobDefinitionInPlan) jobInPlan.getJobDefinition();

 if(jobInPlan.getName().compareTo("44")==0){
  //Setting of some extended name and HORC
  jobDefinitionInPlan.setHighestOkReturnCode(0);
  jobDefinitionInPlan.setExtendedName(zosJobDefinition1.getExtendedName());
  try {
   plan.setJobInstance(jobInPlan, context);
  } catch (ConnLockingException e) {
   //TODO Auto-generated catch block
   e.printStackTrace();
  } catch (ConnNotFoundException e) {
   //TODO Auto-generated catch block
   e.printStackTrace();
  } catch (ConnSecurityException e) {
   //TODO Auto-generated catch block
   e.printStackTrace();
  } catch (ConnTransportException e) {
   //TODO Auto-generated catch block
   e.printStackTrace();
  } catch (ConnValidationException e) {
   //TODO Auto-generated catch block
   e.printStackTrace();
  } catch (ConnException e) {
   //TODO Auto-generated catch block
   e.printStackTrace();
  } catch (RemoteException e) {
   //TODO Auto-generated catch block
   e.printStackTrace();
  }
 }

       if(jobInPlan.getName().compareTo("45")==0){
  jobDefinitionInPlan.setHighestOkReturnCode(0);

  jobInPlanZosAtt=jobInPlan.getZosSpecificAttributes();
  resourceDependencyInPlanList=jobInPlan.getResourceDependencies();
  //Adding a special Resource to the job in a jobStreamInPlan
  for(ResourceDependency rD:(List<ResourceDependency>)jobFM2.getResourceDependencies()){
   resourceName=rD.getResourceKey().getName();
   if(resourceName!=null && !resourceName.isEmpty() && resourceName.compareToIgnoreCase("Resource1")!=0 &&
     resourceName.compareToIgnoreCase("Resource2")!=0 &&  resourceName.compareToIgnoreCase("ParallelServers")!=0){
    //Creation and setting of a ResourceDepInPlan
    resourceDependencyInPlan=new ResourceDependencyInPlan();
    resourceDependencyInPlan.setActionOnComplete(rD.getActionOnComplete());
    resourceDependencyInPlan.setAllocationType(rD.getAllocationType());
    resourceDependencyInPlan.setQuantity(rD.getQuantity());
    workstationInPlanKey=new WorkstationInPlanKey();
    workstationInPlanKey.setName(zosJobDefinition2.getFlowTargetKey().getName());
    resourceInPlanKey=new ResourceInPlanKey(resourceName,workstationInPlanKey);
    resourceDependencyInPlan.setId(ResourceInPlanHelper.keyToId(resourceInPlanKey));
    resourceDependencyInPlan.setWorkstationId(WorkstationInPlanHelper.keyToId(workstationInPlanKey));
    resourceDependencyInPlan.setKey(resourceInPlanKey);
    resourceDependencyInPlanList.add(resourceDependencyInPlan);
    jobInPlanZosAtt.setNumberOfSpecialResources(1);
    break;
           }

       }

       try {
   plan.setWholeJobInstance(jobInPlan, false, null, null, null, null, null, jobInPlan.getResourceDependencies(), null,
              null, null, null, context);
  } catch (ConnLockingException e) {
   //TODO Auto-generated catch block
   e.printStackTrace();
  } catch (ConnNotFoundException e) {
   //TODO Auto-generated catch block
```

```
       e.printStackTrace();
      } catch (ConnSecurityException e) {
       //TODO Auto-generated catch block
       e.printStackTrace();
      } catch (ConnTransportException e) {
       //TODO Auto-generated catch block
       e.printStackTrace();
      } catch (ConnValidationException e) {
       //TODO Auto-generated catch block
       e.printStackTrace();
      } catch (ConnException e) {
       //TODO Auto-generated catch block
       e.printStackTrace();
      } catch (RemoteException e) {
       //TODO Auto-generated catch block
       e.printStackTrace();
      }

     }
    }


    try {
     plan.holdJobStreamInstanceJobs((Identifier)identifierList.get(0), false, context);
    } catch (ConnLockingException e) {
     //TODO Auto-generated catch block
     e.printStackTrace();
    } catch (ConnNotFoundException e) {
     //TODO Auto-generated catch block
     e.printStackTrace();
    } catch (ConnSecurityException e) {
     //TODO Auto-generated catch block
     e.printStackTrace();
    } catch (ConnTransportException e) {
     //TODO Auto-generated catch block
     e.printStackTrace();
    } catch (ConnValidationException e) {
     //TODO Auto-generated catch block
     e.printStackTrace();
    } catch (ConnException e) {
     //TODO Auto-generated catch block
     e.printStackTrace();
    } catch (RemoteException e) {
     //TODO Auto-generated catch block
     e.printStackTrace();
    }


    return null;
   }
  }); // end doAs
  ;
 }

}
```

Utilisation de l'interface de programme d'application pour utiliser le JCL z/OS

Explique comment utiliser l'interface de programme d'application pour utiliser le JCL z/OS.

Vous devez généralement définir les travaux JCL dans Tivoli Workload Scheduler for z/OS en utilisant les panneaux z/OS Program Interface. Cette section vous explique comment créer une interface Java pour gérer le JCL dans la bibliothèque adéquate. Vous pouvez ajouter, lire, modifier et supprimer un travail JCL et pour chacune de ces activités, vous devez être en mesure de vous connecter au Tivoli Workload Scheduler for z/OS Connector qui implémente l'API.

Définition de la connexion à Tivoli Workload Scheduler for z/OS Connector

Explique comment définir une connexion à Tivoli Workload Scheduler for z/OS Connector pour utiliser le JCL z/OS.

Pour définir la connexion au Tivoli Workload Scheduler for z/OS Connector, utilisez une syntaxe identique à la syntaxe suivante :

```
final ZConnModel model = connection.getModelBean();
```

Ajout d'un travail JCL

Explique comme ajouter un travail JCL à l'aide de l'interface de programme d'application (API).

Les paramètres d'ajout d'un travail JCL sont les suivants :

- Le travail lui-même
- La clé de travail composée du nom de la bibliothèque et du nom du travail JCL.

La commande retourne l'ID du travail créé.

L'exemple de code suivant crée un travail JCL et l'ajoute à une bibliothèque de travail spécifique :

```
JCL jcl = new JCL();
JCLKey jobk1 = new JCLKey("TWSSD.CWSD64.JOBLIB","MYJCL");
jcl.setKey(jobk1);
jcl.getTextLines().add("//"+name+" JOB (876903,D07),'AAAAAAA',MSGLEVEL=(1,1), 00010000");
jcl.getTextLines().add("//     MSGCLASS=A,CLASS=A,NOTIFY=CARDELL              00020000");
jcl.getTextLines().add("//STEP1    EXEC PGM=IEFBR14                           00030001");
jcl.getTextLines().add("//SYSPRINT DD SYSOUT=*                                00060000");

id = model.addTWSObject(jcl,null);
```

Lecture du travail JCL

Explique comme lire un travail JCL à l'aide de l'interface de programme d'application (API).

Les paramètres de lecture d'un travail JCL sont les suivants :

- La clé de travail composée du nom de la bibliothèque et du nom du travail JCL
- Une valeur booléenne permettant de déterminer s'il faut verrouiller le travail après l'avoir lu

La commande retourne le travail JCL identifié par la clé de travail

L'exemple de code suivant lit un travail JCL spécifique dans une bibliothèque de travail spécifique :

```
JCLKey jobk1 = new JCLKey("TWSSD.CWSD64.JOBLIB","MYJCL1");

JCL jobJCL = (JCL)model.getTWSObject(JCL.class, jobk1, false, null);
```

Modification du travail JCL

Explique comme modifier un travail JCL à l'aide de l'interface de programme d'application (API).

Lisez tout d'abord le travail JCL et verrouillez-le :

```
JCLKey jobk1 = new JCLKey("TWSSD.CWSD64.JOBLIB","MYJCL1");

JCL jobJCL = (JCL)model.getTWSObject(JCL.class, jobk1, true, null);
```

Les paramètres de modification d'un travail JCL sont les suivants :

- La clé de travail composée du nom de la bibliothèque et du nom du travail JCL
- Le JCL modifié.

La commande retourne l'ID du travail modifié.

L'exemple de code suivant modifie un travail JCL déjà lu et verrouillé :

```
jcl.getTextLines().add("//"+name+" JOB (876903,D07),'AAAAAAA',MSGLEVEL=(1,1), 00010000");
jcl.getTextLines().add("//     MSGCLASS=A,CLASS=A,NOTIFY=PIPPO               00020000");

id = model.setTWSObject(jcl, true, true, null);
```

Suppression du travail JCL

Explique comme supprimer un travail JCL à l'aide de l'interface de programme d'application API.

Lisez tout d'abord le travail JCL et verrouillez-le :

```
JCLKey jobk1 = new JCLKey("TWSSD.CWSD64.JOBLIB","MYJCL1");

JCL jobJCL = (JCL)model.getTWSObject(JCL.class, jobk1, true, null);
```

Les paramètres de suppression d'un travail JCL sont les suivants :

- La clé de travail composée du nom de la bibliothèque et du nom du travail JCL

L'exemple de code suivant supprime un travail JCL déjà lu et verrouillé :

```
model.removeTWSObject(JCL.class, jobk1, null);
```

Matériel de référence

Explique comment accéder au matériel de référence de l'interface de programme d'application.

L'aide Integration Workbench contient tout le matériel de référence dont vous avez besoin.

Pour accéder à ce matériel, procédez comme suit :

1. A partir d'Integration Workbench, sélectionnez **Aide** > **Table des matières**
2. Cliquez sur **Tivoli Workload Scheduler Integration Workbench**, puis sur **Référence**
3. Procurez-vous le matériel de référence de l'un des éléments suivants :
   - Les informations nécessaires pour exécuter les assistants qui créent les projets de l'interface de programme d'application, soit de A à Z, soit à partir d'un exemple
   - Les informations concernant les bibliothèques de l'objet et les fichiers JAR d'exécution
   - Description des schémas XML
   - Un lien vers les informations concernant Tivoli Event Integration Facility
   - Référence complète de chaque classe et méthode Java

Informations complémentaires

Fournit des liens vers des informations supplémentaires concernant l'utilisation des interfaces de programme d'application Java.

Redbooks

Pour plus d'informations sur la programmation de ce type d'API, voir les IBM Redbooks suivants :

Redbooks *IBM : EJB 2.0 Development with WebSphere Studio Application Developer, SG24-6819*

Ce Redbook IBM fournit des informations détaillées pour utiliser efficacement WebSphere Studio Application Developer afin de développer des applications basées sur l'architecture EJB (Enterprise JavaBeans) et de déployer ces applications sur WebSphere Application Server.

Pour accéder à cette publication, suivez ce lien : http://www.redbooks.ibm.com/abstracts/sg246819.html.

Redbooks *IBM : Programmation des interfaces de programme d'application J2EE avec WebSphere Advanced, SG24-6124*

Ce Redbook IBM fournit des exemples de programmation des nouvelles API J2EE à l'aide de VisualAge for Java et de déploiement sur WebSphere Advanced.

Pour accéder à cette publication, suivez ce lien : http://www.redbooks.ibm.com/abstracts/sg246819.html.

Chapitre 4. Gestion de Tivoli Workload Automation avec l'interface de services Web

Utilisez Interface des services Web pour exécuter un sous-ensemble des fonctions Tivoli Workload Scheduler et Tivoli Workload Scheduler for z/OS pour gérer les travaux et les flots de travaux dans le plan à partir de votre propre application Web client. Ni les actions de la base de données, ni les autres actions du plan ne peuvent être implémentées et appelées à l'aide de cette interface.

Dès que vous installez un composant Tivoli Workload Scheduler comprenant WebSphere Application Server, les fichiers WSDL suivants sont automatiquement installés :

SchedulingFactory.wsdl
Utilisez ces services pour soumettre les travaux ou les flots de travaux dans le plan et identifier les travaux et les flots de travaux qui respectent les critères définis.

JobService.wsdl
Utilisez ces services pour afficher les propriétés des travaux dans le plan, afficher la sortie de travaux, définir les propriétés sélectionnées, publier toutes les dépendances, annuler et supprimer les travaux du plan.

JobStreamService.wsdl
Utilisez ces services pour afficher les propriétés des flots de travaux dans le plan, afficher la sortie de flot de travaux, définir les propriétés sélectionnées, publier toutes les dépendances, annuler les flots de travaux du plan.

Ces composants sont installés dans le chemin suivant :
`<chemin_profil_WAS>/installedApps/DefaultNode/<chemin_composant>`

où la valeur par défaut de *<chemin_profil_WAS>* est `<rép_racine_TWA>/WAS/TWSprofile` et *<chemin_composant>* dépend des composants installés :

Connecteur z/OS
`ZConnector.ear/PlanServicesWeb.war/WEB-INF/wsdl`

Autres composants Tivoli Workload Scheduler
`TWSEngineModel.ear/PlanServicesWeb.war/WEB-INF/wsdl`

Si vous disposez à la fois du connecteur z/OS et d'un composant Tivoli Workload Scheduler, les fichiers sont présents deux fois (mais présentent le même contenu).

Ouvrez ces fichiers WSDL avec un outil de développement de service Web. Ces fichiers fournissent :

- La partie serveur qui sert d'interface avec le gestionnaire de domaine maître pour exécuter le sous-ensemble pris en charge d'opérations de planification sur les travaux et les flots de travaux en production.
- Moyen de créer votre propre interface client à partir de l'endroit où vos demandeurs de service peuvent demander à exécuter un sous-ensemble d'opérations à partir de tout système de votre environnement

Le même ensemble de services est fourni pour les deux environnements Tivoli Workload Scheduler et Tivoli Workload Scheduler for z/OS. Cependant, certains services ne peuvent être utilisés que dans un environnement et un grand nombre de ces services possède différents paramètres dans différents environnements, c'est pourquoi ils sont décrits séparément pour les deux environnements.

Interface des services Web est décrit dans les rubriques suivantes :

Services Web de Tivoli Workload Scheduler

Fournit une présentation et les liens vers les descriptions détaillées des services Web fournis pour Tivoli Workload Scheduler.

Voici les détails complets (cliquez sur les liens de ces tables pour obtenir la description du service) :

SchedulingFactory.wsdl

Tableau 1. Services disponibles dans l'interface de services Web SchedulingFactory de Tivoli Workload Scheduler

Nom du service	Actions effectuées
submitJob	Soumet les travaux définis dans la base de données dans le plan de production (environnement distribué uniquement).
submitAdHocJob	Soumet les travaux ad hoc dans le plan de production (environnement distribué uniquement).
queryJobs	Retourne toutes les instances de travail respectant les critères suivants.
submitJobStream	Soumet des flots de travaux dans le plan de production.
queryJobStreams	Retourne toutes les instances de flot de travaux qui correspondent aux critères de filtrage.

JobService.wsdl

Tableau 2. Services disponibles dans l'interface de services Web JobService de Tivoli Workload Scheduler

Nom du service	Actions effectuées
getProperties	Affiche les propriétés du travail.
setProperties	Définit les propriétés spécifiées pour une instance de travail.
getOutput	Extrait la sortie d'une instance de travail.
kill	Supprime une instance de travail.
cancel	Annule une instance de travail.
releaseAllDependencies	Libère toutes les dépendances d'une instance de travail.

JobstreamService.wsdl

Tableau 3. Services disponibles dans l'interface de services Web JobStreamService de Tivoli Workload Scheduler

Nom du service	Actions effectuées
getProperties	Affiche les propriétés du flot de travaux.
setProperties	Indique les propriétés définies d'une instance de flot de travaux.
getJobsList	Affiche la liste de tous les travaux contenus dans un flot de travaux.
cancel	Annule une instance de flot de travaux.
releaseAllDependencies	Libère toutes les dépendances d'une instance de flot de travaux.

Services Web de Tivoli Workload Scheduler for z/OS

Fournit une présentation et les liens vers les descriptions détaillées des services Web fournis pour Tivoli Workload Scheduler for z/OS.

Voici les détails complets (cliquez sur les liens de ces tables pour obtenir la description du service) :

SchedulingFactory.wsdl

Tableau 4. Services disponibles dans l'interface de services Web SchedulingFactory de Tivoli Workload Scheduler for z/OS

Nom du service	Actions effectuées
queryJobs (z/OS)	Retourne toutes les instances de travail respectant les critères suivants.
submitJobStream (z/OS)	Soumet des flots de travaux dans le plan de production.
submitJobStreamWithVarSub (z/OS)	Soumet les flots de travaux avec une substitution de variable dans le plan de production.
editSubmitJobStreamWithVarSub (z/OS)	Edite et soumet les flots de travaux avec la substitution de variable dans le plan de production.
queryJobStreams (z/OS)	Retourne toutes les instances de flot de travaux qui correspondent aux critères de filtrage.

JobService.wsdl

Tableau 5. Services disponibles dans l'interface de services Web JobService de Tivoli Workload Scheduler for z/OS

Nom du service	Actions effectuées
getProperties (z/OS)	Affiche les propriétés du travail.
setProperties (z/OS)	Définit les propriétés spécifiées pour une instance de travail.
getOutput (z/OS)	Extrait la sortie d'une instance de travail.
cancel (z/OS)	Annule une instance de travail.

JobstreamService.wsdl

Tableau 6. Services disponibles dans l'interface de services Web JobStreamService de Tivoli Workload Scheduler for z/OS

Nom du service	Actions effectuées
getProperties (z/OS)	Affiche les propriétés du flot de travaux.
setProperties (z/OS)	Indique les propriétés définies d'une instance de flot de travaux.
getJobsList (z/OS)	Affiche la liste de tous les travaux contenus dans un flot de travaux.
cancel (z/OS)	Annule une instance de flot de travaux.

Gestion des services Web

Explique comment gérer votre environnement de service Web.

La présente section explique comment gérer les services Web à partir de votre application. Elle contient les rubriques suivantes :

- «Accès aux services»
- «Gestion des erreurs», à la page 38

Accès aux services

Accès aux services Web en appelant un proxy.

Les services Web sont accessibles en appelant un proxy contenant le chemin que votre application utilisera pour accéder aux fichiers wsdl de services Web dans l'emplacement suivant :

Environnement distribué (non z/OS)

`http(s)://<localhost>:port_number/PlanServicesWeb/services/<service_name>`

Environnement z/OS

`http(s)://<localhost>:port_number/zPlanServicesWeb/services/<service_name>`

Où :

localhost
Nom d'hôte du gestionnaire de domaine maître

numéro_port
Les numéros de port pour http ou https sont définis dans votre installation WebSphere Application Server dans :

Tivoli Workload Scheduler
`<chemin_profil_WAS>/config/cells/ TWSNodeCell/nodes/TWSNode/servers/server1/server.xml`

où la valeur par défaut de *<chemin_profil_WAS>* est `<rép_racine_TWA>/WAS/TWSprofile`

Dynamic Workload Console
`<rép_profil_JazzSM>/config/cells/JazzSMNode01Cell/nodes/JazzSMNode01/servers/server1/`

où la valeur par défaut de *<rép_profil_JazzSM>* est :

Sur les systèmes d'exploitation Windows
C:\Program Files\IBM\JazzSM\profile

Sur les systèmes d'exploitation UNIX
/opt/IBM/JazzSM/profile

sous :

- `WC_defaulthost` pour http
- `WC_defaulthost_secure` pour https

nom_service
Nom du service que vous appelez : `SchedulingFactory`, `JobService` ou `JobStreamService`.

Voici un exemple d'accès aux services Web dans l'environnement z/OS :

```
SchedulingFactoryProxy proxy = new SchedulingFactoryProxy("http://111.222.333.444:
                               31126/zPlanServicesWeb/services/SchedulingFactory");

String[] jstreams = proxy.submitJobStreamWithVarSub(
                    ENGINE_NAME,
                    JOB_STREAM_KEY,
                    schedTime,
                    deadlineTime,
                    null,
                    null,
                    null,
                    null,
                    null,
                    null,
                    null,
                    variablesToBeSubstituted);
```

Identification du gestionnaire de domaine maître correct

Explique comment identifier le gestionnaire de domaine maître correct lors de l'appel de services Web.

Pour vérifier que votre service Web est exécuté sur le gestionnaire de domaine maître correct, procédez comme suit :

Environnement distribué (non z/OS)

1. Lorsque vous créez le proxy d'accès pour le wsdl dans lequel se trouve le service dont vous avez besoin, accédez auwsdl *sur le même gestionnaire de domaine maître* (moteur) que celui où vous souhaitez qu'il soit exécuté.
2. Lorsque vous appelez le service, paramétrez l'élément `engineName` de la méthode de service sur null.

Le service est exécuté sur le gestionnaire de domaine maître installé en local sur le wsdl auquel vous accédez.

Environnement z/OS

1. Lorsque vous créez le proxy d'accès pour le wsdl dans lequel se trouve le service dont vous avez besoin, accédez au wsdl *sur le système sur lequel le connecteur z/OS est installé*
2. Lorsque vous appelez le service, paramétrez l'élément `engineName` de la méthode de service sur le nom de moteur z/OS sur lequel vous souhaitez exécuter le service.

Le service est exécuté sur le gestionnaire de domaine maître identifié par le nom de moteur z/OS.

Gestion des erreurs

Explique comment gérer les erreurs lors de l'utilisation des services Web.

Cette section indique les erreurs pouvant se produire, suivant le service individuel utilisé. Dans des circonstances normales, chaque erreur est accompagnée d'un autre message d'erreur décrivant le problème de manière plus détaillée. Les erreurs sont les suivantes :

InvalidArguments
La syntaxe de la demande de service n'est pas correcte. Corrigez la syntaxe et recommencez l'opération.

Locking
Une erreur s'est produite lorsque le service a tenté de verrouiller un objet avant de l'utiliser. Cette erreur est généralement due au fait qu'un autre utilisateur possède un objet impliqué dans l'opération verrouillée pour l'édition. Il suffit généralement d'attendre et de retenter l'action. Un échec répété de ce type peut indiquer un problème plus grave sur lequel il faut attirer l'attention de l'administrateur Tivoli Workload Scheduler.

ObjectNotFound
L'objet identifié dans la requête est introuvable et ne peut ainsi pas être soumis. Avant de soumettre une requête modifiant ou supprimant un objet existant, vous pouvez choisir de lancer tout d'abord une requête sur l'objet et d'effectuer uniquement la modification ou la suppression de l'objet ou des objets retournés par la requête. Notez qu'il est possible qu'un objet soit supprimé par un autre utilisateur entre la récupération d'une liste d'objets à l'aide d'une requête et l'implémentation d'une modification ou d'une suppression.

EngineNotMaster
Ce message est uniquement retourné par z/OS Connector et indique que le moteur identifié dans la demande n'est pas le gestionnaire de domaine maître. Cela peut entraîner une erreur lors de l'attribution du nom de moteur.

Transport
Une erreur de communication s'est produite. Lisez le message d'accompagnement pour savoir comment résoudre le problème.

OperationFailed
L'opération a échoué. Lisez le message d'accompagnement pour en connaître la raison et savoir comment résoudre le problème.

Security
L'opération n'a pas pu être exécutée car l'utilisateur exécutant la requête ne dispose pas de l'accès sécurisé adéquat à un ou plusieurs objets soumis à la requête. Changez les droits d'accès sécurisé de l'utilisateur dans le fichier Security ou soumettez les opérations en tant qu'utilisateur disposant des droits adéquats

Services Web SchedulingFactory

Décrit les services Web disponibles dans le fichier SchedulingFactory.wsdl.

Cette section décrit les services Web que vous pouvez utiliser à partir du fichier `SchedulingFactory.wsdl`. Des exemples d'utilisation de certains services Web de la liste sont fournis.

Les services Web sont décrits séparément pour Tivoli Workload Scheduler et Tivoli Workload Scheduler for z/OS.

Services SchedulingFactory de Tivoli Workload Scheduler

Décrit les services Web disponibles dans SchedulingFactory.wsdl pour Tivoli Workload Scheduler.

Les services Web suivants peuvent être utilisés pour servir d'interface avec le plan de Tivoli Workload Scheduler.

submitJob

Décrit le service Web submitJob de Tivoli Workload Scheduler.

Description

Utilisez ce service pour soumettre un travail dans le plan Tivoli Workload Scheduler.

Paramètres d'entrée

engineName
Non utilisé ; paramétré sur null.

jobKey
Clé identifiant le travail dans la base de données Tivoli Workload Scheduler : *workstationName#jobName*.

alias Nom d'alias du travail. Accepte également la valeur `null`.

Résultat

La réponse `submitJobResponse` retourne une chaîne `submitJobReturn` contenant l'identificateur du travail en cours de soumission dans le plan.

submitAdHocJob

Décrit le service Web submitAdHocJob de Tivoli Workload Scheduler.

Description

Utilisez ce service pour soumettre un travail dans le plan Tivoli Workload Scheduler. Le travail n'est pas défini dans la base de données.

Paramètres d'entrée

engineName
Non utilisé ; paramétré sur null.

job Instruction `jobToSubmit` décrivant le travail. L'instruction `jobToSubmit` est définie et présentée dans le fichier `TWS-Types.xsd` et dans «Définition du travail ad hoc».

Résultat

La réponse `submitAdHocJobResponse` retourne une chaîne `submitAdHocJobReturn` contenant l'identificateur du travail en cours de soumission dans le plan.

Définition du travail ad hoc :

Pour soumettre un travail ad hoc, vous devez fournir une instruction `jobToSubmit` décrivant le travail. L'instruction présente le format suivant :

jobName
Nom du travail. Les caractères génériques sont admis, auquel cas, tous les travaux éligibles sont soumis.

workstationName
Nom du poste de travail sur lequel le travail doit être exécuté. Si la zone est vide, le poste de travail sur lequel la commande est exécutée devient la valeur par défaut.

userLogin
Connexion de l'utilisateur exécutant le travail.

taskType
Type de tâche. La valeur affectée peut être l'une des suivantes :
- UNIX
- Windows
- Broker
- Autre

taskString
Chaîne contenant les paramètres contrôlant l'exécution du travail.

priority
Priorité d'exécution du travail. Sa valeur doit être l'une des valeurs suivantes :
- Nombre compris entre 0 et 99.
- hi
- go

command
Indique si le travail est une commande. Sa valeur peut être paramétrée sur `yes` ou `no`.

monitored
Indique si le travail doit être contrôlé par des règles d'événement. Sa valeur peut être paramétrée sur `yes` ou `no`.

requiredConfirmation
Indique si l'exécution (avec ou sans réussite) de ce travail requiert une confirmation de l'utilisateur. Sa valeur peut être paramétrée sur `yes` ou `no`.

recoveryJobName
Nom d'un travail de reprise à exécuter si ce travail se termine avec une anomalie.

recoveryJobWorkstationName
Nom du poste de travail sur lequel le travail de reprise doit être exécuté. La valeur par défaut est `workstationName`.

recoveryOption
Options de reprise du travail. Les valeurs peuvent être les suivantes:
- Arrêter
- Continuer
- Réexécuter

La valeur par défaut est `stop` sans aucun travail de reprise et aucune invite de reprise.

recoveryPromptText
Indique le texte d'une invite de reprise, figurant entre guillemets, qui doit être affiché dans le cas d'une fin anormale du travail. Les règles sont les suivantes :
- Ce texte peut comporter jusqu'à 64 caractères.
- S'il commence par un signe deux-points (:), l'invite s'affiche mais aucune réponse n'est requise pour la poursuite du traitement

- Si le texte commence par un signe d'exclamation (!), l'invite s'affiche mais n'est pas enregistrée dans le fichier journal.

returnCodeMapping
Expression définissant le statut final d'un travail (avec ou sans réussite) en fonction d'une condition sur le code de retour de l'exécution du programme ou du script du travail.

Le code retour peut également être fourni au travail de reprise associé dans la définition de travail. Le travail de reprise effectue alors divers traitements, fondés sur le code retour.

estimatedDuration
Durée prévue d'un travail. Sa valeur est basée sur les statistiques collectées depuis des exécutions antérieures du travail. Si le travail n'a jamais été exécuté auparavant, la valeur par défaut de une minute est utilisée.

Ce paramètre est adapté si le travail est un prédécesseur d'un travail critique.

startTime
Date et heure auxquelles le travail doit démarrer.

latestStartTime
Dernières date et heure auxquelles le travail peut démarrer.

latestStartAction
Action à exécutée sur le travail si la dernière heure de départ s'est écoulée. Les valeurs peuvent être les suivantes:
- Annuler
- Continuer
- Supprimer

deadlineTime
Date et heure auxquelles le travail doit avoir démarré. Après cette heure, le travail est considéré comme en retard.

repeatInterval
Fréquence de répétition pour le travail exprimée en heures et en minutes, au format *hhmm*. Le travail est relancé à chaque fois que le fin de l'intervalle est atteinte. L'intervalle peut dépasser les 24 heures.

Pour plus d'informations, voir la description de la commande `submit job (sbj)` dans Tivoli Workload Scheduler *Tivoli Workload Scheduler: User's Guide and Reference*.

queryJobs

Décrit le service Web queryJobs de Tivoli Workload Scheduler.

Description
Utilisez ce service pour exécuter les requêtes sur les instances de travail Tivoli Workload Scheduler.

Paramètres d'entrée

engineName
Non utilisé ; paramétré sur null.

filter Liste de critères de filtrage représentée par un tableau d'instructions `filterCriteria`. Les instructions `filterCriteria` sont définies et présentées dans le fichier `TWS-Types.xsd` et dans «Définition des critères de filtrage des travaux de requête», à la page 42.

Résultat

La réponse `queryJobsResponse` retourne un tableau `queryJobsReturn` des identificateurs de travaux correspondant à la requête.

Définition des critères de filtrage des travaux de requête :
Chaque instruction `filterCriteria` présente la forme suivante :

details
Non utilisé ; paramétré sur null.

value Tableau de valeurs pour dataType.

minimum
Valeur unique représentant le minimum d'une plage.

maximum
Valeur unique représentant le maximum d'une plage.

dataType
Chaîne identifiant la zone pour laquelle vous avez fourni une valeur, plusieurs valeurs ou une plage. Les zones sont définies et présentées dans le fichier `SchedulingFactory.wsdl` et également comme suit :

JOB_ID
Identificateur du travail.

JOB_NAME
Nom du travail.

JOB_STREAM_NAME
Nom du flot de travaux dans lequel le travail a été exécuté.

WORKSTATION_NAME
Nom du poste de travail exécutant le travail.

STATUS_LIST
Peut prendre l'une des valeurs suivantes :
- BLOCKED
- CANCELLED
- COMPLETED
- ERROR
- HELD
- READY
- STARTED
- WAITING
- UNDECIDED

INTERNAL_STATUS_LIST
Peut prendre l'une des valeurs suivantes :
- ABEND
- ABEND_P
- ADDING
- CANCEL
- CANT_STREAM
- CNPEND
- END_P
- ERROR_STAT
- EXEC
- EXEC_BM
- EXTRN
- FENCE
- HOLD

- MPE_INTRO
- MPE_INTRO_BM
- MPE_SCHED
- MPE_SUSP
- MPE_WAIT
- MPE_WAITD
- PRET
- RESTART_JOB
- SUCC
- SUCC_P
- UNKNOWN
- USER_HELD
- USER_STREAM

PRIORITY
Priorité d'exécution du travail. Peut être l'une des valeurs (ou plage de valeurs) suivantes :
- Nombre compris entre 0 et 99.
- hi
- go

PRIORITY_RANGE
Permet d'analyser les travaux dans une plage de valeurs de priorité. Dans ce cas, le `minimum` et le `maximum` doivent également être fournis.

CONFIRMED
Détermine si seuls les travaux confirmés doivent être sélectionnés. Peut être `true` ou `false`.

RERUN
Détermine si seuls les travaux réexécutés doivent être sélectionnés. Peut être `true` ou `false`.

START_TIME
Heure de début définie pour le travail au format `AAAAMMJJ HHMM` ou en millisecondes.

START_TIME_RANGE
Permet d'analyser les travaux dans une plage d'heures de début au format `AAAAMMJJ HHMM` ou en millisecondes. Dans ce cas, le `minimum` et le `maximum` doivent également être fournis.

UNTIL_TIME
Heure de fin définie pour le travail au format `AAAAMMJJ HHMM` ou en millisecondes.

UNTIL_TIME_RANGE
Permet d'analyser les travaux dans une plage d'heures de fin au format `AAAAMMJJ HHMM` ou en millisecondes. Dans ce cas, le `minimum` et le `maximum` doivent également être fournis.

FINISH_TIME
Heure de fin définie pour le travail au format `AAAAMMJJ HHMM` ou en millisecondes.

FINISH_TIME_RANGE
Permet d'analyser les travaux dans une plage d'heures de fin au format `AAAAMMJJ HHMM` ou en millisecondes. Dans ce cas, le `minimum` et le `maximum` doivent également être fournis.

RECOVERY_OPTION_LIST
Peut prendre l'une des valeurs suivantes :
- STOP
- CONTINUE
- RERUN

MONITORED_JOB
Non utilisé ; paramétré sur null.

TASK Nom de la tâche.

USER_LOGIN
Connexion de l'utilisateur ayant exécuté le travail.

ERROR_CODE
Non utilisé ; paramétré sur null.

submitJobStream

Décrit le service Web submitJobStream de Tivoli Workload Scheduler.

Description
Utilisez ce service pour soumettre un flot de travaux (application) dans le plan Tivoli Workload Scheduler.

Paramètres d'entrée

engineName
Non utilisé ; paramétré sur null.

jsKey Clé identifiant le flot de travaux dans la base de données de planificateur : *workstationName#jobStreamName*

schedTime
Heure de soumission planifiée pour le flot de travaux.

alias Nom d'alias du flot de travaux.

Résultat
La réponse `submitJobStreamResponse` retourne un tableau `submitJobStreamReturn` des identificateurs de flots de travaux en cours de soumission dans le plan.

queryJobStreams

Décrit le service Web queryJobStreams de Tivoli Workload Scheduler.

Description
Utilisez ce service pour exécuter les requêtes sur les instances de flot de travaux Tivoli Workload Scheduler.

Paramètres d'entrée

engineName
Non utilisé ; paramétré sur null.

filter Liste de critères de filtrage représentée par un tableau d'instructions `filterCriteria`. Les instructions `filterCriteria` sont définies et présentées dans le fichier `TWS-Types.xsd` et dans «Définition des critères de filtrage d'analyse des flots de travaux», à la page 45.

Résultat
La réponse `queryJobStreamsResponse` retourne un tableau `queryJobStreamsReturn` des identificateurs de flots de travaux correspondant à la requête.

Définition des critères de filtrage d'analyse des flots de travaux :
Chaque instruction `filterCriteria` présente la forme suivante :

details
Non utilisé ; paramétré sur null.

value Tableau de valeurs pour dataType.

minimum
Valeur unique représentant le minimum d'une plage.

maximum
Valeur unique représentant le maximum d'une plage.

dataType
Chaîne identifiant la zone pour laquelle vous avez fourni une valeur, plusieurs valeurs ou une plage. Les zones sont définies et présentées dans le fichier `SchedulingFactory.wsdl` et également comme suit :

JOB_STREAM_ID
Identificateur du flot de travaux.

JOB_STREAM_NAME
Nom du flot de travaux.

WORKSTATION_NAME
Nom du poste de travail exécutant le flot de travaux.

STATUS_LIST
Peut prendre l'une des valeurs suivantes :
- BLOCKED
- CANCELLED
- COMPLETED
- ERROR
- HELD
- READY
- STARTED
- WAITING
- UNDECIDED

INTERNAL_STATUS_LIST
Peut prendre l'une des valeurs suivantes :
- ABEND
- ADDING
- CANCEL
- CNPEND
- EXEC
- HOLD
- PRET
- SUCC
- SUSP
- USER_HELD

LIMIT
Valeur limite avec laquelle le flot de travaux a été exécuté. Peut prendre l'une des valeurs suivantes :
- Nombre compris entre 0 et 1024.
- hi
- go

LIMIT_RANGE
Permet d'analyser les flots de travaux dans une plage de valeurs limite. Dans ce cas, le `minimum` et le `maximum` doivent également être fournis.

PRIORITY
Priorité d'exécution avec laquelle le flot de travaux a été exécuté. Peut prendre l'une des valeurs suivantes :
- Nombre compris entre 0 et 99.
- hi
- go

PRIORITY_RANGE
Permet d'analyser les flots de travaux dans une plage de valeurs de priorité. Dans ce cas, le `minimum` et le `maximum` doivent également être fournis.

CARRIED_FORWARD
Détermine si seuls les flots de travaux exécutés doivent être sélectionnés. Peut être `true` ou `false`.

CARRIED_FORWARD
Détermine si seuls les flots de travaux exécutés doivent être sélectionnés. Peut être `true` ou `false`.

START_TIME
Heure de début définie pour le flot de travaux au format `AAAAMMJJ HHMM` ou en millisecondes.

START_TIME_RANGE
Permet d'analyser les flots de travaux dans une plage d'heures de début au format `AAAAMMJJ HHMM` ou en millisecondes. Dans ce cas, le `minimum` et le `maximum` doivent également être fournis.

UNTIL_TIME
Heure de fin définie pour le flot de travaux au format `AAAAMMJJ HHMM` ou en millisecondes.

UNTIL_TIME_RANGE
Permet d'analyser les flots de travaux dans une plage d'heures de fin au format `AAAAMMJJ HHMM` ou en millisecondes. Dans ce cas, le `minimum` et le `maximum` doivent également être fournis.

DEADLINE_TIME
Heure d'échéance définie pour le travail au format `AAAAMMJJ HHMM` ou en millisecondes.

DEADLINE_TIME_RANGE
Permet d'analyser les flots de travaux dans une plage d'heures d'échéance au format `AAAAMMJJ HHMM` ou en millisecondes. Dans ce cas, le `minimum` et le `maximum` doivent également être fournis.

Services SchedulingFactory de Tivoli Workload Scheduler for z/OS

Décrit les services Web disponibles dans SchedulingFactory.wsdl pour Tivoli Workload Scheduler for z/OS.

Les services Web suivants peuvent être utilisés pour servir d'interface avec le plan de Tivoli Workload Scheduler for z/OS.

queryJobs (z/OS)

Décrit le service Web queryJobs de Tivoli Workload Scheduler for z/OS.

Description

Utilisez ce service pour exécuter les requêtes sur les instances de travail Tivoli Workload Scheduler for z/OS.

Paramètres d'entrée

engineName
Nom du moteur Tivoli Workload Scheduler for z/OS.

filter Liste de critères de filtrage représentée par un tableau d'instructions `filterCriteria`. Les instructions `filterCriteria` sont définies et présentées dans le fichier `TWS-Types.xsd` et dans «Définition des critères de filtrage d'analyse des travaux z/OS».

Résultat

La réponse `queryJobsResponse` retourne un tableau `queryJobsReturn` des identificateurs de travaux correspondant à la requête. Les zones dans la matrice sont les suivantes :

occurrenceToken
Identificateur de l'occurrence contenant l'opération.

extendedStatus
Code de statut étendu affecté à l'opération.

errorCode
Code d'erreur terminant l'opération.

operationCommand
Peut être : BD = Bind shadow; job EX = Execute operation; KJ = Kill operation; KR = Kill recovery job; MH = Hold operation; MR = Release operation; NP = NOP operation; PN = Prompt reply no; PY = Prompt reply yes; UN = Un-NOP operation.

authorityGroup
Nom du groupe de droits d'accès auquel appartient l'opération.

cleanUpStatus
Peut être : Blank=none C=Completed E=Ended in error I=Initiated O=Avail opinfo R=Request opinfo S=Started W=Waiting opinfo

latestOutPassed
L'opération a dépassé sa dernière heure de début et est en retard. Peut être `true` ou `false`.

latestOut
Heure la plus tardive à laquelle, calculée à partir de l'heure de création de plan, à laquelle l'opération peut être lancée pour respecter l'échéance (HHMM).

actualArrival
Heure d'exécution réelle de l'opération (HHMM).

actualEnd
Heure à laquelle l'opération est signalée comme exécutée ou terminée par une erreur (HHMM).

Définition des critères de filtrage d'analyse des travaux z/OS :
Chaque instruction `filterCriteria` présente la forme suivante :

details
Non utilisé. Paramétré sur null.

value Tableau de valeurs pour dataType.

minimum
Valeur unique représentant le minimum d'une plage.

maximum
Valeur unique représentant le maximum d'une plage.

dataType
Chaîne identifiant la zone pour laquelle vous avez fourni une valeur, plusieurs valeurs ou une plage. Les zones sont définies et présentées dans le fichier `SchedulingFactory.wsdl` et également comme suit :

JOB_ID
Non utilisé ; paramétré sur null.

JOB_NAME
Nom du travail. Doit être numérique.

JOB_STREAM_NAME
Nom du flot de travaux dans lequel le travail a été exécuté.

WORKSTATION_NAME
Nom du poste de travail exécutant le travail.

STATUS_LIST
Peut prendre l'une des valeurs suivantes :
- BLOCKED
- CANCELLED
- COMPLETED
- ERROR
- HELD
- READY
- STARTED
- WAITING
- UNDECIDED

INTERNAL_STATUS_LIST
Peut prendre l'une des valeurs suivantes :
- COMPLETE
- DELETED
- ERROR
- WAITING
- STARTED
- UNDECIDED
- READY
- PENDINGPRED
- WAITINGFORINPUT
- INTERRUPTED
- NOREPORTINGPRED

PRIORITY
Priorité d'exécution du travail. Peut être l'une des valeurs (ou plage de valeurs) suivantes :
- Nombre compris entre 0 et 99.
- hi
- go

PRIORITY_RANGE
Permet d'analyser les travaux dans une plage de valeurs de priorité. Dans ce cas, le `minimum` et le `maximum` doivent également être fournis.

CONFIRMED
Non utilisé ; paramétré sur null.

RERUN
Non utilisé ; paramétré sur null.

START_TIME
Heure de début définie pour le travail au format `AAAAMMJJ HHMM` ou en millisecondes.

START_TIME_RANGE
Permet d'analyser les travaux dans une plage d'heures de début au format `AAAAMMJJ HHMM` ou en millisecondes. Dans ce cas, le `minimum` et le `maximum` doivent également être fournis.

UNTIL_TIME
Non utilisé ; paramétré sur null.

UNTIL_TIME_RANGE
Non utilisé ; paramétré sur null.

FINISH_TIME
Heure de fin définie pour le travail au format `AAAAMMJJ HHMM` ou en millisecondes.

FINISH_TIME_RANGE
Permet d'analyser les travaux dans une plage d'heures de fin au format `AAAAMMJJ HHMM` ou en millisecondes. Dans ce cas, le `minimum` et le `maximum` doivent également être fournis.

RECOVERY_OPTION_LIST
Non utilisé ; paramétré sur null.

MONITORED_JOB
Détermine si seuls les travaux surveillés doivent être sélectionnés. Peut être `true` ou `false`.

TASK Nom de la tâche.

USER_LOGIN
Non utilisé ; paramétré sur null.

ERROR_CODE
Code d'erreur du travail.

submitJobStream (z/OS)

Décrit le service Web submitJobStream de Tivoli Workload Scheduler for z/OS.

Description
Utilisez ce service pour soumettre un flot de travaux (application) dans le plan Tivoli Workload Scheduler for z/OS.

Paramètres d'entrée

engineName
Nom du moteur Tivoli Workload Scheduler for z/OS.

jsKey Clé identifiant le flot de travaux dans la base de données de planificateur : *jobStreamName*

schedTime
Non utilisé ; paramétré sur null.

alias Non utilisé ; paramétré sur null.

Résultat
La réponse `submitJobStreamResponse` retourne un tableau `submitJobStreamReturn` des identificateurs de flots de travaux en cours de soumission dans le plan. Notez que les identificateurs d'objet du plan retournés contiennent le caractère `'\0'`. Caractère non valide qui doit être remplacé par un blanc.

submitJobStreamWithVarSub

Décrit le service Web submitJobStreamWithVarSub de Tivoli Workload Scheduler for z/OS.

Description
Utilisez ce service pour soumettre un flot de travaux (application) dans le plan Tivoli Workload Scheduler for z/OS et pour affecter des valeurs à toutes les variables présentes dans les travaux inclus. La substitution de variable est effectuée uniquement sur les variables paramétrables soumises au cours de la phase de configuration des travaux. Les autres variables sont remplacées au cours de la soumission.

Remarque : Le flot de travaux ne doit contenir aucun travail de configuration. S'il en contient, le message suivant s'affiche :

```
EQQM229E
   JCL BROWSE/EDIT CAN ONLY BE SELECTED FOR PROCESSOR WORKSTATIONS.
```

Paramètres d'entrée

engineName
Nom du moteur Tivoli Workload Scheduler for z/OS.

jsKey Clé identifiant le flot de travaux dans la base de données Tivoli Workload Scheduler for z/OS. Le format valide est *jobStreamName*.

schedTime
Heure d'entrée des données pour le flot de travaux.

deadlineTime
Heure de fin de l'exécution pour le flot de travaux.

priority
Valeur de priorité du flot de travaux.

description
Description du flot de travaux. Maximum 24 caractères.

groupName
Nom du groupe de flot de travaux. Maximum 16 caractères.

ownerName
Nom du propriétaire du flot de travaux. Maximum 16 caractères.

ownerDescription
Description du propriétaire du flot de travaux. Maximum 24 caractères.

authorityGroup
Groupe de droits d'accès du flot de travaux. Maximum 8 caractères.

dependenciesResolution
Indique le type de dépendances à résoudre. La valeur peut être la suivante :

Y Résoudre les dépendances remplacées et remplaçantes.

N Ignorer toutes les dépendances.

P Résoudre uniquement les dépendances remplacées.

S Résoudre uniquement les dépendances remplaçantes.

Si aucune valeur n'est saisie, la valeur par défaut est **N**.

variableTable
Nom de la table de variables associée au flot de travaux. Maximum 16 caractères.

variableToBeSubstituted
Tableau d'instructions `Property` indiquant les variables soumises dans le flot de travaux. L'instruction`Property` est définie et présentée dans le fichier `TWS-Types.xsd` et prend la forme suivante :

valeur Tableau de valeurs pour dataType.

dataType
Chaîne identifiant la variable pour laquelle vous fournissez la ou les valeurs et qui peuvent être détectées dans la table de variable définie.

Résultat
Le service retourne un tableau des identificateurs des flots de travaux en cours de soumission dans le plan. Notez que les identificateurs d'objet du plan retournés contiennent le caractère `'\0'`. Caractère non valide qui doit être remplacé par un blanc.

Exemple :
Voici un exemple de codage de `submitJobStreamWithVarSub`, dans lequel les variables à remplacer sont appelées SURNAME et NAME et se trouvent dans la table de variables par défaut pour le flot de travaux :

```
long now = System.currentTimeMillis();

 /**
  * inputArrivalTime
  */
 Calendar schedTime = new GregorianCalendar();
 Date schedDate = new Date(now);
 schedTime.setTime(schedDate);

 /**
  * deadlineTime
  */
 Calendar deadlineTime = new GregorianCalendar();
 Date deadlineDate = new Date(now + HOUR_8);
 deadlineTime.setTime(deadlineDate);

 /**
  * variablesToBeSubstituted
  */
 Property prop1 = new Property();
 prop1.setDataType("SURNAME");
 String[] valueSurname = {"Robinson"};
 prop1.setValue(valueSurname);
```

```
 Property prop2 = new Property();
 prop2.setDataType("NAME");
 String[] valueName = {"John"};
 prop2.setValue(valueName);

 Property[] variablesToBeSubstituted = new Property[2];
 variablesToBeSubstituted[0] = prop1;
 variablesToBeSubstituted[1] = prop2;

 SchedulingFactoryProxy proxy = new SchedulingFactoryProxy("http://111.222.333.444:
                               55555/zPlanServicesWeb/services/SchedulingFactory");

String[] jstreams = proxy.submitJobStreamWithVarSub(
                    ENGINE_NAME,
                    JOB_STREAM_KEY,
                    schedTime,
                    deadlineTime,
                    null,
                    null,
                    null,
                    null,
                    null,
                    null,
                    null,
                    variablesToBeSubstituted);
```

editSubmitJobStreamWithVarSub

Décrit le service Web editSubmitJobStreamWithVarSub de Tivoli Workload Scheduler for z/OS.

Description

Utilisez ce service pour soumettre un flot de travaux (application) dans le plan Tivoli Workload Scheduler for z/OS et pour exécuter certaines ou toutes les actions suivantes :

- Affectez les valeurs aux variables présentes dans les travaux contenus dans le flot de travaux
- Ajoutez, modifiez ou supprimez les travaux et leurs dépendances internes dans le flot de travaux

La substitution de variable est effectuée uniquement sur les variables paramétrables soumises au cours de la phase de configuration des travaux. Les autres variables sont remplacées au cours de la soumission.

Remarque : Le flot de travaux ne doit contenir aucun travail de configuration. S'il en contient, le message suivant s'affiche :

```
EQQM229E
   JCL BROWSE/EDIT CAN ONLY BE SELECTED FOR PROCESSOR WORKSTATIONS.
```

Paramètres d'entrée

engineName
Nom du moteur Tivoli Workload Scheduler for z/OS.

jsKey Clé identifiant le flot de travaux dans la base de données Tivoli Workload Scheduler for z/OS. Le format valide est *jobStreamName*.

schedTime
Heure d'entrée des données pour le flot de travaux.

deadlineTime
Heure de fin de l'exécution pour le flot de travaux.

priority
Valeur de priorité du flot de travaux.

description
Description du flot de travaux. Maximum 24 caractères.

groupName
Nom du groupe de flot de travaux. Maximum 16 caractères.

ownerName
Nom du propriétaire du flot de travaux. Maximum 16 caractères.

ownerDescription
Description du propriétaire du flot de travaux. Maximum 24 caractères.

authorityGroup
Groupe de droits d'accès du flot de travaux. Maximum 8 caractères.

dependenciesResolution
Indique le type de dépendances à résoudre. La valeur peut être la suivante :

Y Résoudre les dépendances remplacées et remplaçantes.

N Ignorer toutes les dépendances.

P Résoudre uniquement les dépendances remplacées.

S Résoudre uniquement les dépendances remplaçantes.

Si aucune valeur n'est saisie, la valeur par défaut est **N**.

variableTable
Nom de la table de variables associée au flot de travaux. Maximum 16 caractères.

jobsList
Tableau d'instructions `ZOSJob` indiquant les travaux ajoutés, édités et supprimés dans le flot de travaux. Le type `ZOSJob` est défini et présenté dans le fichier `TWS-Types.xsd` et dans «Ajoutez, modifiez ou supprimez les travaux dans le flot de travaux que vous soumettez», à la page 59

dependencyList
Tableau d'instructions `Dependency` indiquant les dépendances ajoutées et supprimées dans le flot de travaux. Le type `Dependency` est défini et présenté dans le fichier `TWS-Types.xsd` et dans «Ajout ou suppression des dépendances internes dans le plan», à la page 62

variablesToBeSubstituted
Tableau d'instructions `Property` indiquant les variables soumises dans le flot de travaux. L'instruction`Property` est définie et présentée dans le fichier `TWS-Types.xsd` et prend la forme suivante :

valeur Tableau de valeurs pour dataType.

dataType
Chaîne identifiant la variable pour laquelle vous fournissez la ou les valeurs et qui peuvent être détectées dans la table de variable définie.

Résultat

Le service retourne un tableau des identificateurs des flots de travaux en cours de soumission dans le plan. Notez que les identificateurs d'objet du plan retournés contiennent le caractère '\0'. Caractère non valide qui doit être remplacé par un blanc.

Exemple :

Voici un exemple de codage de editSubmitJobStreamWithVarSub :

```
long now = System.currentTimeMillis();

 /**
  * inputArrivalTime
  */
 Calendar schedTime = new GregorianCalendar();
 Date schedDate = new Date(now);
 schedTime.setTime(schedDate);

 /**
  * deadlineTime
  */
 Calendar deadlineTime = new GregorianCalendar();
 Date deadlineDate = new Date(now + HOUR_8);
 deadlineTime.setTime(deadlineDate);

 /**
  * variablesToBeSubstituted
  */
 Property prop1 = new Property();
 prop1.setDataType("SURNAME");
 String[] valueSurname = {"Robinson"};
 prop1.setValue(valueSurname);

 Property prop2 = new Property();
 prop2.setDataType("NAME");
 String[] valueName = {"John"};
 prop2.setValue(valueName);

 Property[] variablesToBeSubstituted = new Property[2];
 variablesToBeSubstituted[0] = prop1;
 variablesToBeSubstituted[1] = prop2;

 ZOSJob[] zj = new ZOSJob[1];
 zj[0] = new ZOSJob();
 zj[1] = new ZOSJob();
 zj[2] = new ZOSJob();

 zj[0].setAction("ADD");
 zj[0].setJobNumber(50);
 zj[0].setWorkstationName("VALL");
 zj[0].setDuration(10000);
 zj[0].setParallelServer(1);
 zj[0].setAutoSubmit("true");
 zj[0].setJobName("VPCEN1");
 zj[0].setInternalStatus("W");

 zj[1].setAction("MODIFY");
 zj[1].setJobNumber(10);
 zj[1].setWorkstationName("VAL1");
 zj[1].setJobName("VPCEN2");
 zj[1].setInternalStatus("R");

 zj[2].setAction("DELETE");
 zj[2].setJobNumber(50);

 Dependency[] dj = new Dependency[2];
```

```
 dj[0] = new Dependency();
 dj[0].setAction("ADD");
 dj[0].setType("PREDECESSOR");
 dj[0].setJobNumber(50);
 dj[0].setDependencyNumber(40);

 dj[0] = new Dependency();
 dj[0].setAction("DELETE");
 dj[0].setType("PREDECESSOR");
 dj[0].setJobNumber(40);
 dj[0].setDependencyNumber(20);

String[] jstreams = proxy.editSubmitJobStreamWithVarSub(ENGINE_NAME,
            JOB_STREAM_KEY,
              schedTime,
              deadlineTime,
              null,
              null,
                    null, null, null, null, null, zj, dj, variablesToBeSubstituted);
```

editSubmitJobStreamWithJobVarSub

Décrit le service Web editSubmitJobStreamWithJobVarSub de Tivoli Workload Scheduler for z/OS.

Description

Utilisez ce service pour soumettre un flot de travaux (application) dans le plan Tivoli Workload Scheduler for z/OS et pour exécuter certaines ou toutes les actions suivantes :

- Affectez des valeurs aux variables présentes dans les travaux inclus dans le flot de travaux avec l'option qui spécifie un ensemble de variables différent pour les travaux individuels.
- Ajoutez, modifiez ou supprimez des travaux et leurs dépendances internes dans le flot de travaux.

La substitution de variable est effectuée uniquement sur les variables paramétrables soumises au cours de la phase de configuration des travaux. Les autres variables sont remplacées au cours de la soumission.

Remarque : Le flot de travaux ne doit contenir aucun travail de configuration. S'il en contient, le message suivant s'affiche :

```
EQQM229E
   JCL BROWSE/EDIT CAN ONLY BE SELECTED FOR PROCESSOR WORKSTATIONS.
```

Paramètres d'entrée

engineName
Nom du moteur Tivoli Workload Scheduler for z/OS.

jsKey Clé identifiant le flot de travaux dans la base de données Tivoli Workload Scheduler for z/OS. Le format valide est *jobStreamName*.

schedTime
Heure d'entrée des données pour le flot de travaux.

deadlineTime
Heure de fin de l'exécution pour le flot de travaux.

priority
Valeur de priorité du flot de travaux.

description
Description du flot de travaux. Maximum 24 caractères.

groupName
Nom du groupe de flot de travaux. Maximum 16 caractères.

ownerName
Nom du propriétaire du flot de travaux. Maximum 16 caractères.

ownerDescription
Description du propriétaire du flot de travaux. Maximum 24 caractères.

authorityGroup
Groupe de droits d'accès du flot de travaux. Maximum 8 caractères.

dependenciesResolution
Indique le type de dépendances à résoudre. La valeur peut être la suivante :

Y Résoudre les dépendances remplacées et remplaçantes.

N Ignorer toutes les dépendances.

P Résoudre uniquement les dépendances remplacées.

S Résoudre uniquement les dépendances remplaçantes.

Si aucune valeur n'est saisie, la valeur par défaut est **N**.

variableTable
Nom de la table de variables associée au flot de travaux. Maximum 16 caractères.

jobsList
Tableau d'instructions `ZOSJob` indiquant les travaux ajoutés, édités et supprimés dans le flot de travaux. Le type `ZOSJob` est défini et présenté dans le fichier `TWS-Types.xsd` et dans «Ajoutez, modifiez ou supprimez les travaux dans le flot de travaux que vous soumettez», à la page 59

dependencyList
Tableau d'instructions `Dependency` indiquant les dépendances ajoutées et supprimées dans le flot de travaux. Le type `Dependency` est défini et présenté dans le fichier `TWS-Types.xsd` et dans «Ajout ou suppression des dépendances internes dans le plan», à la page 62

variablesToBeSubstituted
Tableau d'instructions `Property` indiquant les variables soumises dans le flot de travaux. L'instruction`Property` est définie et présentée dans le fichier `TWS-Types.xsd` et prend la forme suivante :

value Tableau de valeurs pour dataType.

dataType
Chaîne identifiant la variable pour laquelle vous fournissez la ou les valeurs et qui peuvent être détectées dans la table de variable définie.

jobVariablesToBeSubstituted
Itération des éléments suivants :

Tableau d'instructions `Property` indiquant les variables

soumises dans le flot de travaux. L'instructionProperty est définie et présentée dans le fichier TWS-Types.xsd et prend la forme suivante

value Tableau de valeurs pour dataType.

dataType
Chaîne identifiant la variable pour laquelle vous fournissez la ou les valeurs et qui peuvent être détectées dans la table de variable définie.

jobNumber
Entier (de 1 à 255) qui identifie le travail qui utilisera les variables spécifiées ci-dessous et qui ont dû être précédemment définies avec .setJobNumber(*numéro*)

Par exemple :

```
String [][] variablesMap1 = new String [3][2];
variablesMap1 [0][0] = "VAR1";
variablesMap1 [0][1] = "ValVar1ForExec1";
variablesMap1 [1][0] = "VAR2";
variablesMap1 [1][1] = "ValVar2ForExec1";
variablesMap1 [2][0] = "VAR3";
variablesMap1 [2][1] = "ValVar3ForExec1";
int jobNum = 5;

jobVariablesToBeSubstituted.put(jobNum, variablesMap1);

String [][] variablesMap2 = new String [2][2];
variablesMap2 [0][0] = "VAR2";
variablesMap2 [0][1] = "ValVar2ForExec2";
variablesMap2 [1][0] = "VAR4";
variablesMap2 [1][1] = "ValVar4ForExec2";
jobNum = 10;

jobVariablesToBeSubstituted.put(jobNum, variablesMap2);

...

String [][] variablesMap4 = new String [4][2];
variablesMap4 [0][0] = "VAR3";
variablesMap4 [0][1] = "ValVar3ForExec4";
variablesMap4 [1][0] = "VAR4";
variablesMap4 [1][1] = "ValVar4ForExec4";
variablesMap4 [2][0] = "VAR5";
variablesMap4 [2][1] = "ValVar5ForExec4";
variablesMap4 [3][0] = "VAR6";
variablesMap4 [3][1] = "ValVar6ForExec4";
jobNum = 20;

jobVariablesToBeSubstituted.put(jobNum, variablesMap4);
```

La séquence se répète pour tous les travaux qui utilisent leurs propres variables particulières ou des valeurs de variables plutôt que celles spécifiées pour la totalité du flot de travaux. Tous les travaux qui ne sont pas dans l'itération utilisent les variables ou les valeurs de variable spécifiées avec le paramètre variablesToBeSubstituted pour le flot de travaux ou aucune variable.

Résultat

Le service retourne un tableau des identificateurs des flots de travaux en

cours de soumission dans le plan. Notez que les identificateurs d'objet du plan retournés contiennent le caractère '\0'. Caractère non valide qui doit être remplacé par un blanc.

Exemple :
Voici un exemple de codage de editSubmitJobStreamWithJobVarSub :

```
String description = "";
 String group = null;
 String owner = null;
 String ownerDescription = null;
 String variableTable = "STREAM1";

 List<ZosJobInfo>jobsToDelete = new ArrayList<ZosJobInfo>();
     List<ZosJobInfo>jobsToAdd = new ArrayList<ZosJobInfo>();

     List<ZosJobInfo>jobsToModify = new ArrayList<ZosJobInfo>();
     ZosJobInfo job1 = new ZosJobInfo();
     job1.setJobNumber(5);
     job1.setJobName("EXEC1");
     jobsToModify.add(job1);
     ZosJobInfo job2 = new ZosJobInfo();
     job2.setJobNumber(10);
     job2.setJobName("EXEC2");
     jobsToModify.add(job2);
     ZosJobInfo job21 = new ZosJobInfo();
     job21.setJobNumber(15);
     job21.setJobName("EXEC3");
     jobsToModify.add(job21);
     ZosJobInfo job3 = new ZosJobInfo();
     job3.setJobNumber(20);
     job3.setJobName("EXEC4");
     jobsToModify.add(job3);

     HashMap<String, List<Integer>> dependencyToDelete = new HashMap<String, List<Integer>>();
     HashMap<String, List<Integer>> dependencyToAdd = new HashMap<String, List<Integer>>();

     // valeurs de variable JCL (pour tous les travaux) par défaut
  String[][] variablesToBeSubstituted = new String [1][2];
 variablesToBeSubstituted [0][0] = "VAR1";
 variablesToBeSubstituted [0][1] = "ValVar1ForExec2And3";

 String authorityGroup = null;
 boolean holdAll = true;
 DependenciesResolutionOption resolutionOption = DependenciesResolutionOption.RESOLUTION_ALL;

 // valeurs de variables JCL au niveau du travail
 HashMap<Integer, String[][]> jobVariablesToBeSubstituted = new HashMap<Integer, String[][]>();
  String [][] variablesMap1 = new String [3][2];
  variablesMap1 [0][0] = "VAR1";
  variablesMap1 [0][1] = "ValVar1ForExec1";
  variablesMap1 [1][0] = "VAR2";
  variablesMap1 [1][1] = "ValVar2ForExec1";
  variablesMap1 [2][0] = "VAR3";
  variablesMap1 [2][1] = "ValVar3ForExec1";
  int jobNum = 5;

  jobVariablesToBeSubstituted.put(jobNum, variablesMap1);

 String [][] variablesMap2 = new String [2][2];
 variablesMap2 [0][0] = "VAR2";
 variablesMap2 [0][1] = "ValVar2ForExec2";
 variablesMap2 [1][0] = "VAR4";
 variablesMap2 [1][1] = "ValVar4ForExec2";
 jobNum = 10;

 jobVariablesToBeSubstituted.put(jobNum, variablesMap2);

 String [][] variablesMap3 = new String [2][2];
 variablesMap3 [0][0] = "VAR2";
 variablesMap3 [0][1] = "ValVar2ForExec3";
 variablesMap3 [1][0] = "VAR4";
 variablesMap3 [1][1] = "ValVar4ForExec3";
 jobNum = 15;

 jobVariablesToBeSubstituted.put(jobNum, variablesMap3);

 String [][] variablesMap4 = new String [4][2];
```

```
variablesMap4 [0][0] = "VAR3";
variablesMap4 [0][1] = "ValVar3ForExec4";
variablesMap4 [1][0] = "VAR4";
variablesMap4 [1][1] = "ValVar4ForExec4";
variablesMap4 [2][0] = "VAR5";
variablesMap4 [2][1] = "ValVar5ForExec4";
variablesMap4 [3][0] = "VAR6";
variablesMap4 [3][1] = "ValVar6ForExec4";
jobNum = 20;

jobVariablesToBeSubstituted.put(jobNum, variablesMap4);

Context context = null;


@SuppressWarnings("unchecked")
     List<Identifier> list = plan.editAddJobStreamInstanceWithVariableSubstitution(
                             JSName, startTime, deadlineTime, priority, description,
                             group, owner, ownerDescription, variableTable, jobsToDelete,
                             jobsToAdd, jobsToModify, dependencyToDelete, dependencyToAdd,
                             variablesToBeSubstituted, authorityGroup, holdAll,
                             resolutionOption, jobVariablesToBeSubstituted, context);
```

Ajoutez, modifiez ou supprimez les travaux dans le flot de travaux que vous soumettez

Pour exécuter l'une de ces actions, vous devez tracer un tableau d'instructions `ZOSJob` contenant les spécifications de ces actions. Si le tableau `jobsList` est vide ou s'il contient des valeurs null (excepté si cela est autorisé), aucune action n'est entreprise.

Le tableau 7 décrit les paramètres que vous devez définir dans chaque instruction`ZOSJob` pour spécifier un travail que vous souhaitez ajouter, modifier ou supprimer dans le flot de travaux que vous soumettez dans le plan.

Tableau 7. Propriétés à définir pour les travaux ajoutés, modifiés et supprimés dans les éléments `ZOSJob`.

Elément	Action à exécuter sur le travail dans le flot de travaux		
	Ajouter	**Modifier**	**Supprimer**
action	Paramétrez sur `ADD`. Si la zone est vide ou contient une valeur non autorisée, un message d'erreur est retourné.	Paramétrez sur `MODIFY`. Si la zone est vide ou contient une valeur non autorisée, un message d'erreur est retourné.	Paramétrez sur `DELETE`. Si la zone est vide ou contient une valeur non autorisée, un message d'erreur est retourné.
jobNumber	Paramétrez sur une valeur comprise entre 1 et 255. Si la zone est vide, un message d'erreur est retourné.	Paramétrez sur une valeur comprise entre 1 et 255 correspondant au numéro de travail du plan. Si la zone est vide, un message d'erreur est retourné.	Paramétrez sur une valeur comprise entre 1 et 255 correspondant au numéro de travail du plan. Si la zone est vide, un message d'erreur est retourné.

Tableau 7. Propriétés à définir pour les travaux ajoutés, modifiés et supprimés dans les éléments `ZOSJob`. (suite)

Elément	Action à exécuter sur le travail dans le flot de travaux		
	Ajouter	**Modifier**	**Supprimer**
workstationName	Paramétrez sur l'ID du poste de travail tel qu'il est défini dans le plan en cours. Maximum 4 caractères. Si la zone est vide, un message d'erreur est retourné.	Paramétrez sur l'ID du poste de travail tel qu'il est défini dans le plan en cours. Maximum 4 caractères. Si la zone est vide, c'est la dernière valeur enregistrée qui est utilisée.	Saisissez une valeur null.
textDescription	Facultatif. Maximum 24 caractères.	Maximum 24 caractères. Si la zone est vide, c'est la dernière description enregistrée qui est utilisée.	Saisissez une valeur null.
jobName	Paramétrez sur le nom du travail. Maximum 8 caractères. Si la zone est vide, un message d'erreur est retourné.	Maximum 8 caractères. Si la zone est vide, c'est la dernière valeur enregistrée qui est utilisée.	Saisissez une valeur null.
parallelServer	Nombre de serveurs parallèles requis pour exécuter le travail. Si la zone est vide, sa valeur par défaut est 0. Suivant le poste de travail, un message d'erreur peut être retourné dans certains cas.	Paramétrez sur -1 pour laisser la dernière valeur enregistrée. Par défaut, la valeur est 0.	Saisissez une valeur null.
durée	Exprimée en millisecondes. Si la zone est vide, un message d'erreur est retourné.	Paramétrez sur -1 pour laisser la dernière valeur enregistrée. Par défaut, la valeur est 0.	Saisissez une valeur null.
autoSubmit	Si la soumission automatique est requise, paramétrez sur `true` ou sur`false`. Si la zone est vide, elle est paramétrée sur `false`.	Si la soumission automatique est requise, paramétrez sur `true` ou sur`false`. Si la zone est vide, c'est la dernière valeur enregistrée qui est utilisée.	Saisissez une valeur null.
timeDependent	Si le travail dépend de l'heure, paramétrez sur `true` ou sur `false`. Si la zone est vide, elle est paramétrée sur `false`.	Si le travail dépend de l'heure, paramétrez sur `true` ou sur `false`. Si la zone est vide, c'est la dernière valeur enregistrée qui est utilisée.	Saisissez une valeur null.

Tableau 7. Propriétés à définir pour les travaux ajoutés, modifiés et supprimés dans les éléments `ZOSJob`. (suite)

Elément	Action à exécuter sur le travail dans le flot de travaux		
	Ajouter	**Modifier**	**Supprimer**
centralizedScript	Si le travail possède un script centralisé, paramétrez sur `true` ou sur `false`. Si la zone est vide, elle est paramétrée sur `true`. Cette option est uniquement active sur les travaux planifiés pour être exécutés sur un agent tolérant aux pannes.	Saisissez une valeur null (impossible de modifier la valeur d'origine).	Saisissez une valeur null.
inputArrivalTime	Heure d'entrée des données au format HH.MM.	Heure d'entrée des données au format HH.MM. Si la zone est vide, c'est la dernière valeur enregistrée qui est utilisée.	Saisissez une valeur null.
R1	Le nombre d'instances de ressource de travail 1 est requis. Si la zone est vide, sa valeur est 0.	Le nombre d'instances de ressource de travail 1 est requis. Paramétrez sur -1 pour laisser la dernière valeur enregistrée. Par défaut, la valeur est 0.	Saisissez une valeur null.
R2	Le nombre d'instances de ressource de travail 2 est requis. Si la zone est vide, sa valeur est 0.	Le nombre d'instances de ressource de travail 2 est requis. Paramétrez sur -1 pour laisser la dernière valeur enregistrée. Par défaut, la valeur est 0.	Saisissez une valeur null.
internalStatus	Peut prendre l'une des valeurs suivantes : • COMPLETE • DELETED • ERROR • WAITING • STARTED • UNDECIDED • READY • PENDINGPRED • WAITINGFORINPUT • INTERRUPTED • NOREPORTINGPRED Si la zone est vide, elle est paramétrée sur `UNDECIDED`.	Si la zone est vide, c'est la dernière valeur enregistrée qui est utilisée.	Saisissez une valeur null.

N'oubliez pas de définir les dépendances (le cas échéant) des nouveaux travaux que vous ajoutez.

Ajout ou suppression des dépendances internes dans le plan

Pour exécuter l'une de ces actions, vous devez tracer un tableau d'instructions `Dependency` contenant les spécifications de ces actions. Si le tableau `dependecyLst` est vide ou s'il contient des valeurs null, aucune action n'est entreprise.

La section ci-dessous décrit les paramètres que vous devez définir dans chaque instruction `Dependency` pour spécifier une dépendance que vous souhaitez ajouter ou supprimer dans le flot de travaux du plan. Si tous les éléments ne sont pas définis, une erreur est retournée :

action Action à entreprendre sur la dépendance. Peut être `ADD` ou `DELETE`.

type Type de dépendance. Peut être `PREDECESSOR` ou `SUCCESSOR`.

jobNumber
Numéro du travail concerné.

dependencyNumber
Numéro du travail que vous définissez comme prédécesseur ou successeur de *jobNumber*.

queryJobStreams (z/OS)

Décrit le service Web queryJobStreams de Tivoli Workload Scheduler for z/OS.

Description
Utilisez ce service pour exécuter les requêtes sur les instances de flot de travaux Tivoli Workload Scheduler for z/OS.

Paramètres d'entrée

engineName
Nom du moteur Tivoli Workload Scheduler for z/OS.

filter Liste de critères de filtrage représentée par un tableau d'instructions `filterCriteria`. Les instructions `filterCriteria` sont définies et présentées dans le fichier `TWS-Types.xsd` et dans «Définition des critères de filtrage d'analyse des flots de travaux z/OS».

Résultat
La réponse `queryJobStreamsResponse` retourne un tableau `queryJobStreamsReturn` des identificateurs de flots de travaux correspondant à la requête. La matrice comporte les zones suivantes :

occurrenceToken
Identificateur d'occurrence.

owner ID du propriétaire défini pour l'application.

authorityGroup
Nom du groupe de droits d'accès auquel appartient l'application.

containingMonitoredJob
L'application comprend au moins une opération surveillée. Peut être `true` ou `false`.

Définition des critères de filtrage d'analyse des flots de travaux z/OS :
Chaque instruction `filterCriteria` présente la forme suivante :

details
Non utilisé ; paramétré sur null.

value Tableau de valeurs pour dataType.

minimum
Valeur unique représentant le minimum d'une plage.

maximum
Valeur unique représentant le maximum d'une plage.

dataType
Chaîne identifiant la zone pour laquelle vous avez fourni une valeur, plusieurs valeurs ou une plage. Les zones sont définies et présentées dans le fichier `SchedulingFactory.wsdl` et comme suit :

JOB_STREAM_NAME
Nom du flot de travaux.

WORKSTATION_NAME
Nom du poste de travail exécutant le flot de travaux.

STATUS_LIST
Peut prendre l'une des valeurs suivantes :
- BLOCKED
- CANCELLED
- COMPLETED
- ERROR
- HELD
- READY
- STARTED
- WAITING
- UNDECIDED

INTERNAL_STATUS_LIST
Peut prendre l'une des valeurs suivantes :
- COMPLETE
- DELETED
- ERROR
- WAITING
- STARTED
- UNDECIDED

PRIORITY
Priorité d'exécution avec laquelle le flot de travaux a été exécuté. Peut prendre l'une des valeurs suivantes :
- Nombre compris entre 0 et 99.
- hi
- go

PRIORITY_RANGE
Permet d'analyser les flots de travaux dans une plage de valeurs de priorité. Dans ce cas, le `minimum` et le `maximum` doivent également être fournis.

START_TIME
Heure de début définie pour le flot de travaux au format `AAAAMMJJ HHMM` ou en millisecondes.

START_TIME_RANGE
Permet d'analyser les flots de travaux dans une plage d'heures de début au format `AAAAMMJJ HHMM` ou en millisecondes. Dans ce cas, le `minimum` et le `maximum` doivent également être fournis.

DEADLINE_TIME
Heure d'échéance définie pour le travail au format AAAAMMJJ HHMM ou en millisecondes.

DEADLINE_TIME_RANGE
Permet d'analyser les flots de travaux dans une plage d'heures d'échéance au format AAAAMMJJ HHMM ou en millisecondes. Dans ce cas, le minimum et le maximum doivent également être fournis.

OCCURRENCE_TOKEN
Jeton d'occurrence de l'instance de flot de travaux du plan (en valeur hexadécimale).

OWNER
Propriétaire du flot de travaux.

AUTH_GROUP
Groupe de droits d'accès affecté au flot de travaux.

MONITORED_JOB
Détermine si seuls les travaux surveillés doivent être sélectionnés. Peut être true ou false.

Détails JobService

Décrit les services Web disponibles dans le fichier JobService.wsdl.

Cette section décrit les services Web que vous pouvez utiliser à partir du fichier JobService.wsdl.

Les services Web sont décrits séparément pour Tivoli Workload Scheduler et Tivoli Workload Scheduler for z/OS.

Services Web JobService de Tivoli Workload Scheduler

Décrit les services Web disponibles dans JobService.wsdl pour Tivoli Workload Scheduler.

Les services Web suivants peuvent être utilisés pour servir d'interface avec les instances de travail de Tivoli Workload Scheduler.

getProperties

Décrit le service Web getProperties des travaux Tivoli Workload Scheduler.

Description
Utilisez ce service pour afficher des informations concernant une instance de travail.

Vous devez avoir un accès list au travail du fichier de sécurité pour exécuter ce service.

Paramètres d'entrée

engineName
Non utilisé ; paramétré sur null.

jobId Identificateur de travail dans le plan : *workstationName#jobStreamName.jobName*.

Résultat
La réponse getPropertiesResponse retourne un tableau getPropertiesReturn contenant les zones d'informations suivantes :

jobId Identificateur du travail.

jobName
Nom du travail.

jobStreamName
Nom du flot de travaux comprenant le travail.

workstationName
Nom du poste de travail exécutant le travail.

jobStreamWorkstationName
Nom du poste de travail associé au flot de travaux.

jobNumber
Numéro attribué au travail lors de l'exécution.

priority
Priorité d'exécution du travail. Peut être l'une des valeurs (ou plage de valeurs) suivantes :
- Nombre compris entre 0 et 99.
- hi
- go

status Peut prendre l'une des valeurs suivantes :
- ERROR
- HELD
- READY
- RUNNING
- SUCCESSFULL
- UNDECIDED
- WAITING

internalStatus
Peut prendre l'une des valeurs suivantes :
- ABEND
- ABENDP
- BOUND
- CANCP
- DONE
- ERROR
- EXEC
- EXTRN
- FAIL
- FENCE
- HOLD
- INTRO
- PEND
- READY
- RJOB
- SCHED
- SUCC
- SUCCP
- SUSP
- USER
- WAIT
- WAITD

requiredConfirmation
Peut être `true` ou `false`.

aliased
Peut être `true` ou `false`.

canceled
Peut être `true` ou `false`.

every L'instance de travail a été exécutée avec l'option `every`. Peut être `true` ou `false`.

everyRerun
L'instance de travail est une réexécution d'un travail défini à l'aide de l'option `every`. Peut être `true` ou `false`.

external
Le travail est un prédécesseur d'un autre flot de travaux ou l'un de ses travaux. Peut être `true` ou `false`.

jobLate
Le travail a dépassé son échéance d'exécution. Peut être `true` ou `false`.

pendingCancellation
Le flot de travaux est en cours d'annulation. L'annulation est reportée jusqu'à ce que toutes les dépendances, y compris l'heure `at`, soient résolues. Peut être `true` ou `false`.

recoveryRerunJob
Le travail est défini par l'action de reprise RERUN qui permet au travail de reprise d'intervenir avant que le programme ne tente d'exécuter une nouvelle fois le travail parent. Peut être `true` ou `false`.

released
Le travail a été supprimé de ses dépendances. Peut être `true` ou `false`.

rerunJob
Le travail a été réexécuté. Peut être `true` ou `false`.

running
Le travail est toujours en cours. Peut être `true` ou `false`.

startTime
Heure de début définie pour le travail au format `AAAAMMJJ HHMM` ou en millisecondes.

lateststartTime
Valeur de la restriction `until` de l'instance de travail au format `AAAAMMJJ HHMM` ou en millisecondes.

lateststartAction
Action entreprise une fois la durée de restriction `until` écoulée. Peut être CANCEL, CONTINUE ou SUPPRESS.

deadlineTime
Valeur de la restriction d'échéance de l'instance de travail au format `AAAAMMJJ HHMM` ou en millisecondes.

repeatRange
Intervalle de temps entre `chaque` réexécution du travail en millisecondes.

Si le service n'a pas été exécuté avec succès, `getPropertiesResponse` retourne l'une des erreurs décrites dans «Gestion des erreurs», à la page 38.

setProperties

Décrit le service Web setProperties pour les travaux Tivoli Workload Scheduler.

Description

Utilisez ce service pour définir les propriétés supplémentaires d'un travail dans le plan.

Vous devez avoir un accès `submit` au travail du fichier de sécurité pour exécuter ce service.

Paramètres d'entrée

engineName
Non utilisé ; paramétré sur null.

jobId Identificateur de travail dans le plan : *workstationName#jobStreamName.jobName*.

properties
Tableau contenant les propriétés que vous souhaitez définir. Ces propriétés peuvent être :

priorityIsMonitored
La priorité de travail est surveillée s'il s'agit d'un travail clé. La valeur est `true` ou `false`.

requiresConfirmation
La confirmation de l'opérateur est obligatoire après l'exécution du travail pour marquer ce dernier comme ayant réussi ou échoué. La valeur est `true` ou `false`.

startTime
Heure à laquelle le travail doit démarrer au format `AAAAMMJJ HHMM` ou en millisecondes.

lateststartTime
Dernière heure à laquelle le travail doit démarrer au format `AAAAMMJJ HHMM` ou en millisecondes.

lateststartAction
Action qui sera entreprise si le travail dépasse sa dernière heure de début. Peut être CANCEL, CONTINUE ou SUPPRESS.

deadlineTime
Durée pendant laquelle le travail doit être exécuté au format `AAAAMMJJ HHMM` ou en millisecondes.

repeatRange
Intervalle de temps entre `chaque` réexécution du travail en millisecondes.

Résultat

La réponse `setPropertiesResponse` est vide si le service a été exécuté avec succès ; sinon, elle retourne l'une des erreurs décrites dans «Gestion des erreurs», à la page 38.

getOutput

Décrit le service Web getOutput des travaux Tivoli Workload Scheduler.

Description

Utilisez ce service pour obtenir le journal d'exécution d'un travail dans le plan.

Vous devez être connecté en tant que utilisateur_TWS pour disposer des droits adéquats pour exécuter ce service.

Paramètres d'entrée

engineName
Non utilisé ; paramétré sur null.

jobId Identificateur de travail dans le plan : *workstationName#jobStreamName.jobName*.

Résultat
La réponse `getOutputResponse` retourne une chaîne contenant le journal d'exécution du travail spécifié.

Si le service n'a pas été exécuté avec succès, `getOutputResponse` retourne l'une des erreurs décrites dans «Gestion des erreurs», à la page 38.

kill

Décrit le service Web kill des travaux Tivoli Workload Scheduler.

Description
Utilisez ce service pour arrêter un travail en cours d'exécution.

Vous devez avoir un accès `kill` au travail du fichier de sécurité pour exécuter ce service.

Paramètres d'entrée

engineName
Non utilisé ; paramétré sur null.

jobId Identificateur de travail dans le plan : *workstationName#jobStreamName.jobName*.

Résultat
La réponse `killResponse` est vide si le service a été exécuté avec succès ; sinon, elle retourne l'une des erreurs décrites dans «Gestion des erreurs», à la page 38.

cancel

Décrit le service Web d'annulation des travaux Tivoli Workload Scheduler.

Description
Utilisez ce service pour annuler un travail.

Si vous annulez le travail avant de l'avoir lancé, il ne démarre pas. Si vous l'annulez une fois qu'il a été lancé, il poursuit son exécution. Si vous annulez un travail en cours d'exécution et qu'il se termine sur un état ABEND, aucune tentative de reprise automatique du travail n'est effectuée.

Vous devez avoir un accès `cancel` au travail du fichier de sécurité pour exécuter ce service.

Paramètres d'entrée

engineName
Non utilisé ; paramétré sur null.

jobId Identificateur de travail dans le plan : *workstationName#jobStreamName.jobName*.

isPending
La valeur est `true` ou `false`. `True` annule le travail uniquement après la résolution de ses dépendances. `False` annule le travail

immédiatement (tous les travaux et flots de travaux dépendant du travail annulé sont immédiatement supprimés de la dépendance).

Résultat

La réponse `cancelResponse` est vide si le service a été exécuté avec succès ; sinon, elle retourne l'une des erreurs décrites dans «Gestion des erreurs», à la page 38.

releaseAllDependencies

Décrit le service Web releaseAllDependencies des travaux Tivoli Workload Scheduler.

Description

Utilisez ce service pour supprimer un travail de toutes les dépendances définies.

Vous devez avoir un accès `release` au travail du fichier de sécurité pour exécuter ce service.

Paramètres d'entrée

engineName
Non utilisé ; paramétré sur null.

jobId Identificateur de travail dans le plan : *workstationName#jobStreamName.jobName*.

Résultat

La réponse `releaseAllDependenciesResponse` est vide si le service a été exécuté avec succès ; sinon, elle retourne l'une des erreurs décrites dans «Gestion des erreurs», à la page 38.

Services Web JobService de Tivoli Workload Scheduler for z/OS

Décrit les services Web disponibles dans JobService.wsdl pour Tivoli Workload Scheduler for z/OS.

Les services Web suivants peuvent être utilisés pour servir d'interface avec les opérations de Tivoli Workload Scheduler for z/OS.

getProperties (z/OS)

Décrit le service Web getProperties des opérations Tivoli Workload Scheduler for z/OS.

Description

Utilisez ce service pour afficher des informations concernant une opération.

Paramètres d'entrée

engineName
Nom du contrôleur Tivoli Workload Scheduler for z/OS.

jobId Identificateur d'opération dans le plan en cours. Notez que les identificateurs d'objet du plan retournés contiennent le caractère '\0'. Caractère non valide qui doit être remplacé par un blanc.

Résultat

La réponse `getPropertiesResponse` retourne un tableau `getPropertiesReturn` contenant les zones d'informations suivantes :

occurrenceToken
Identificateur de l'occurrence contenant l'opération.

extendedStatus
Code de statut étendu affecté à l'opération.

errorCode
Code d'erreur terminant l'opération.

operationCommand
Peut être : BD = Bind shadow; job EX = Execute operation; KJ = Kill operation; KR = Kill recovery job; MH = Hold operation; MR = Release operation; NP = NOP operation; PN = Prompt reply no; PY = Prompt reply yes; UN = Un-NOP operation.

authorityGroup
Nom du groupe de droits d'accès auquel appartient l'opération.

cleanUpStatus
Peut être : Blank=none C=Completed E=Ended in error I=Initiated O=Avail opinfo R=Request opinfo S=Started W=Waiting opinfo

latestOutPassed
L'opération a dépassé sa dernière heure de début et est en retard. Peut être `true` ou `false`.

latestOut
Heure la plus tardive à laquelle, calculée à partir de l'heure de création de plan, à laquelle l'opération peut être lancée pour respecter l'échéance (HHMM).

actualArrival
Heure d'exécution réelle de l'opération (HHMM).

actualEnd
Heure à laquelle l'opération est signalée comme exécutée ou terminée par une erreur (HHMM).

Si le service n'a pas été exécuté avec succès, `getPropertiesResponse` retourne l'une des erreurs décrites dans «Gestion des erreurs», à la page 38.

setProperties (z/OS)

Décrit le service Web setProperties des opérations Tivoli Workload Scheduler for z/OS.

Description
Utilisez ce service pour définir des propriétés supplémentaires pour une opération.

Paramètres d'entrée

engineName
Nom du contrôleur Tivoli Workload Scheduler for z/OS.

jobId Identificateur d'opération dans le plan en cours. Notez que les identificateurs d'objet du plan retournés contiennent le caractère '\0'. Caractère non valide qui doit être remplacé par un blanc.

properties
Tableau contenant les propriétés que vous souhaitez définir. Ces propriétés peuvent être :

priorityIsMonitored
La priorité est surveillée s'il s'agit d'une opération clé. La valeur est `true` ou `false`.

startTime
Heure à laquelle l'opération doit démarrer au format `AAAAMMJJ HHMM` ou en millisecondes.

lateststartTime
Dernière heure à laquelle l'opération doit démarrer au format `AAAAMMJJ HHMM` ou en millisecondes.

lateststartAction
Action qui sera entreprise si l'opération dépasse sa dernière heure de début. Peut être CANCEL, CONTINUE ou SUPPRESS.

deadlineTime
Heure à laquelle l'opération doit être exécutée au format `AAAAMMJJ HHMM` ou en millisecondes.

Résultat

La réponse `setPropertiesResponse` est vide si le service a été exécuté avec succès ; sinon, elle retourne l'une des erreurs décrites dans «Gestion des erreurs», à la page 38.

getOutput (z/OS)

Décrit le service Web getOutput des opérations Tivoli Workload Scheduler for z/OS.

Description

Utilisez ce service pour obtenir le journal de travail d'une opération.

Paramètres d'entrée

engineName
Nom du contrôleur Tivoli Workload Scheduler for z/OS.

jobId Identificateur d'opération dans le plan en cours. Notez que les identificateurs d'objet du plan retournés contiennent le caractère '\0'. Caractère non valide qui doit être remplacé par un blanc.

Résultat

La réponse `getOutputResponse` retourne une chaîne contenant le journal de travail de l'opération spécifiée.

Si le service n'a pas été exécuté avec succès, `getOutputResponse` retourne l'une des erreurs décrites dans «Gestion des erreurs», à la page 38.

cancel (z/OS)

Décrit le service Web cancel des opérations Tivoli Workload Scheduler for z/OS.

Description

Utilisez ce service pour annuler une opération.

Notez que ce service fonctionne uniquement si l'opération n'a pas démarré. Si l'opération a déjà démarré, elle poursuit son exécution.

Paramètres d'entrée

engineName
Nom du contrôleur Tivoli Workload Scheduler for z/OS.

jobId Identificateur d'opération dans le plan en cours. Notez que les identificateurs d'objet du plan retournés contiennent le caractère '\0'. Caractère non valide qui doit être remplacé par un blanc.

isPending
L'opération a le statut en attente. Elle peut prendre la valeur `true` ou `false`.

Résultat
La réponse `cancelResponse` est vide si le service a été exécuté avec succès ; sinon, elle retourne l'une des erreurs décrites dans «Gestion des erreurs», à la page 38.

Détails JobStreamService

Décrit les services Web disponibles dans le fichier JobStreamService.wsdl.

Cette section décrit les services Web que vous pouvez utiliser à partir du fichier `JobStreamService.wsdl`.

Les services Web sont décrits séparément pour Tivoli Workload Scheduler et Tivoli Workload Scheduler for z/OS.

Services Web JobStreamService de Tivoli Workload Scheduler

Décrit les services Web disponibles dans JobStreamService.wsdl pour Tivoli Workload Scheduler.

Les services Web suivants peuvent être utilisés pour servir d'interface avec les instances de flot de travaux de Tivoli Workload Scheduler.

getProperties

Décrit le service Web setProperties des flots de travaux Tivoli Workload Scheduler.

Description
Utilisez ce service pour afficher des informations concernant une instance de flot de travaux.

Vous devez avoir un accès `list` au flot de travaux du fichier de sécurité pour exécuter ce service.

Paramètres d'entrée

engineName
Non utilisé ; paramétré sur null.

jobStreamId
Identificateur de flot de travaux dans le plan : *workstationName#jobStreamName(hhmm[date])* ou *workstation#jobstream_id*. Pour plus d'informations, voir *Tivoli Workload Scheduler - Guide d'utilisation et de référence*.

Résultat
La réponse `getPropertiesResponse` retourne un tableau `getPropertiesReturn` contenant les zones d'informations suivantes :

jobStreamId
Identificateur d'instance de flot de travaux.

jobStreamName
Nom du flot de travaux.

aliasJobStreamName
Nom d'alias de flot de travaux.

originalJobStreamName
Le nom du flot de travaux tel que défini dans la base de données.

workstationName
Nom du poste de travail associé au flot de travaux.

status Peut prendre l'une des valeurs suivantes :
- BLOCKED
- CANCELLED
- ERROR
- HELD
- READY
- RUNNING
- SUCCESSFUL
- UNDECIDED
- WAITING

internalStatus
Peut prendre l'une des valeurs suivantes :
- ABEND
- ABENDP
- CANCP
- DONE
- ERROR
- EXEC
- EXTRN
- FAIL
- FENCE
- HOLD
- INTRO
- PEND
- PRET
- RJOB
- SCHED
- SUCC
- SUCCP
- SUSP
- USER
- WAIT
- WAITD

limite Nombre maximal de travaux du flot de travaux.

priority
Priorité d'exécution définie pour le flot de travaux. Peut être l'une des valeurs (ou plage de valeurs) suivantes :
- Nombre compris entre 0 et 99.
- hi
- go

numberOfJobs
Nombre de travaux contenus dans le flot de travaux.

canceled
L'instance de flot de travaux présente le statut `cancelled`. Peut être `true` ou `false`.

carriedForward
L'instance de flot de travaux a été exécutée. Peut être `true` ou `false`.

carryForward
Le flot de travaux a été planifié avec l'option `carryforward`. Peut être `true` ou `false`.

external
Le flot de travaux est un prédécesseur d'un autre flot de travaux ou l'un de ses travaux situé dans un autre réseau Tivoli Workload Scheduler. Peut être `true` ou `false`.

lateJobStream
L'instance de flot de travaux a dépassé son échéance d'exécution. Peut être `true` ou `false`.

pendingCancellation
Le flot de travaux est en cours d'annulation. L'annulation est reportée jusqu'à ce que toutes les dépendances, y compris l'heure `at`, soient résolues. Peut être `true` ou `false`.

released
Le flot de travaux a été supprimé de ses dépendances. Peut être `true` ou `false`.

startTime
Heure de début définie pour le flot de travaux au format `AAAAMMJJ HHMM` ou en millisecondes.

lateststartTime
Valeur de la restriction `until` du flot de travaux au format `AAAAMMJJ HHMM` ou en millisecondes.

lateststartAction
Action entreprise une fois la durée de restriction `until` écoulée. Peut être CANCEL, CONTINUE ou SUPPRESS.

deadlineTime
Valeur de la restriction d'échéance de l'instance de flot de travaux au format `AAAAMMJJ HHMM` ou en millisecondes.

Si le service n'a pas été exécuté avec succès, `getPropertiesResponse` retourne l'une des erreurs décrites dans «Gestion des erreurs», à la page 38.

setProperties

Décrit le service Web setProperties des flots de travaux Tivoli Workload Scheduler.

Description
Utilisez ce service pour définir des propriétés supplémentaires pour un flot de travaux du plan.

Vous devez avoir un accès `submit` au flot de travaux du fichier de sécurité pour exécuter ce service.

Paramètres d'entrée

engineName
Non utilisé ; paramétré sur null.

jobStreamId
Identificateur de flot de travaux dans le plan : *workstationName#jobStreamName(hhmm[date])* ou *workstation#jobstream_id*. Pour plus d'informations, voir *Tivoli Workload Scheduler - Guide d'utilisation et de référence*.

properties

Tableau contenant les propriétés que vous souhaitez définir. Ces propriétés peuvent être :

limite Nombre de travaux du flot de travaux pour être exécutés simultanément sur le même poste de travail.

priority

Priorité d'exécution. Peut être l'une des valeurs (ou plage de valeurs) suivantes :

- Nombre compris entre 0 et 99.
- hi
- go

isCarryForward

La valeur est `true` ou `false`.

isMonitored

Le flot de travaux est surveillé. La valeur est `true` ou `false`.

startTime

Heure à laquelle le flot de travaux doit démarrer au format `AAAAMMJJ HHMM` ou en millisecondes.

lateststartTime

Dernière heure à laquelle le flot de travaux doit démarrer au format `AAAAMMJJ HHMM` ou en millisecondes.

lateststartAction

Action qui sera entreprise si le flot de travaux dépasse sa dernière heure de début. Peut être CANCEL, CONTINUE ou SUPPRESS.

deadlineTime

Durée pendant laquelle le flot de travaux doit être exécutée au format `AAAAMMJJ HHMM` ou en millisecondes.

Résultat

Si le service a été exécuté avec succès, la réponse `setPropertiesResponse` retourne l'identificateur du flot de travaux modifié et il retourne l'une des erreurs décrites dans «Gestion des erreurs», à la page 38.

getJobsList

Décrit le service Web getJobsList des flots de travaux Tivoli Workload Scheduler.

Description

Utilisez ce service pour obtenir une liste des travaux - et leurs propriétés - d'un flot de travaux du plan.

Vous devez avoir un accès `list` au flot de travaux du fichier de sécurité pour exécuter ce service.

Paramètres d'entrée

engineName

Non utilisé ; paramétré sur null.

jobStreamId

Identificateur de flot de travaux dans le plan : *workstationName#jobStreamName(hhmm[date])* ou *workstation#jobstream_id*. Pour plus d'informations, voir *Tivoli Workload Scheduler - Guide d'utilisation et de référence*.

Résultat

La réponse `getJobsListResponse` retourne un tableau `getJobsListReturn` contenant la liste des travaux contenus dans le flot de travaux. Pour chaque instance de travail, les détails suivants sont également répertoriés :

jobId Identificateur du travail.

jobName
Nom du travail.

jobStreamName
Nom du flot de travaux comprenant le travail.

workstationName
Nom du poste de travail exécutant le travail.

jobStreamWorkstationName
Nom du poste de travail associé au flot de travaux.

jobNumber
Numéro attribué au travail lors de l'exécution.

priority
Priorité d'exécution du travail. Peut être l'une des valeurs (ou plage de valeurs) suivantes :
- Nombre compris entre 0 et 99.
- hi
- go

status Peut prendre l'une des valeurs suivantes :
- ERROR
- HELD
- READY
- RUNNING
- SUCCESSFULL
- UNDECIDED
- WAITING

internalStatus
Peut prendre l'une des valeurs suivantes :
- ABEND
- ABENDP
- BOUND
- CANCP
- DONE
- ERROR
- EXEC
- EXTRN
- FAIL
- FENCE
- HOLD
- INTRO
- PEND
- PRET
- RJOB
- SCHED
- SUCC
- SUCCP
- SUSP
- USER

- WAIT
- WAITD

requiredConfirmation
Peut être `true` ou `false`.

aliased
Peut être `true` ou `false`.

canceled
Peut être `true` ou `false`.

every L'instance de travail a été exécutée avec l'option `every`. Peut être `true` ou `false`.

everyRerun
L'instance de travail est une réexécution d'un travail défini à l'aide de l'option `every`. Peut être `true` ou `false`.

external
Le travail est un prédécesseur d'un autre flot de travaux ou l'un de ses travaux. Peut être `true` ou `false`.

jobLate
Le travail a dépassé son échéance d'exécution. Peut être `true` ou `false`.

pendingCancellation
Le flot de travaux est en cours d'annulation. L'annulation est reportée jusqu'à ce que toutes les dépendances, y compris l'heure `at`, soient résolues. Peut être `true` ou `false`.

recoveryRerunJob
Le travail est défini par l'action de reprise RERUN qui permet au travail de reprise d'intervenir avant que le programme ne tente d'exécuter une nouvelle fois le travail parent. Peut être `true` ou `false`.

released
Le travail a été supprimé de ses dépendances. Peut être `true` ou `false`.

rerunJob
Le travail a été réexécuté. Peut être `true` ou `false`.

exécution
Le travail est toujours en cours. Peut être `true` ou `false`.

startTime
Heure de début définie pour le travail au format `AAAAMMJJ HHMM` ou en millisecondes.

lateststartTime
Valeur de la restriction `until` de l'instance de travail au format `AAAAMMJJ HHMM` ou en millisecondes.

lateststartAction
Action entreprise une fois la durée de restriction `until` écoulée. Peut être CANCEL, CONTINUE ou SUPPRESS.

deadlineTime
Valeur de la restriction d'échéance de l'instance de travail au format `AAAAMMJJ HHMM` ou en millisecondes.

repeatRange
Intervalle de temps entre `chaque` réexécution du travail en millisecondes.

Si le service n'a pas été exécuté avec succès, `getJobsListResponse` retourne l'une des erreurs décrites dans «Gestion des erreurs», à la page 38.

cancel

Décrit le service Web cancel des flots de travaux Tivoli Workload Scheduler.

Description

Utilisez ce service pour annuler un flot de travaux.

Si vous annulez le flot de travaux avant de l'avoir lancé, il ne démarre pas. Si vous l'annulez une fois qu'il est lancé, les travaux en cours sont exécutés jusqu'à la fin mais aucun autre travail n'est lancé.

Vous devez avoir un accès `cancel` au flot de travaux du fichier de sécurité pour exécuter ce service.

Paramètres d'entrée

engineName
Non utilisé ; paramétré sur null.

jobStreamId
Identificateur de flot de travaux dans le plan : *workstationName#jobStreamName(hhmm[date])* ou *workstation#jobstream_id*. Pour plus d'informations, voir *Tivoli Workload Scheduler - Guide d'utilisation et de référence*.

isPending
Il peut prendre la valeur `true` ou `false`. `True` annule le flot de travaux uniquement après la résolution de ses dépendances. `False` annule le flot de travaux immédiatement (tous les travaux et flots de travaux dépendant du travail annulé sont immédiatement supprimés de la dépendance).

Résultat

La réponse `cancelResponse` est vide si le service a été exécuté avec succès ; sinon, elle retourne l'une des erreurs décrites dans «Gestion des erreurs», à la page 38.

releaseAllDependencies

Décrit le service Web releaseAllDependencies des flots de travaux Tivoli Workload Scheduler.

Description

Utilisez ce service pour supprimer un flot de travaux de toutes les dépendances définies.

Vous devez avoir un accès `release` au flot de travaux du fichier de sécurité pour exécuter ce service.

Paramètres d'entrée

engineName
Non utilisé ; paramétré sur null.

jobStreamId
Identificateur de flot de travaux dans le plan : *workstationName#jobStreamName(hhmm[date])* ou

workstation#jobstream_id. Pour plus d'informations, voir *Tivoli Workload Scheduler - Guide d'utilisation et de référence*.

Résultat

La réponse `releaseAllDependenciesResponse` est vide si le service a été exécuté avec succès ; sinon, elle retourne l'une des erreurs décrites dans «Gestion des erreurs», à la page 38.

Services Web JobStreamService de Tivoli Workload Scheduler for z/OS

Décrit les services Web disponibles dans JobStreamService.wsdl pour Tivoli Workload Scheduler for z/OS.

Les services Web suivants peuvent être utilisés pour servir d'interface avec les occurrences d'application de Tivoli Workload Scheduler for z/OS.

getProperties (z/OS)

Décrit le service Web getProperties des occurrences d'application Tivoli Workload Scheduler for z/OS.

Description

Utilisez ce service pour afficher des informations concernant une occurrence d'application.

Paramètres d'entrée

engineName
Nom du contrôleur Tivoli Workload Scheduler for z/OS.

jobStreamId
Identificateur d'occurrence dans le plan en cours. Notez que les identificateurs d'objet du plan retournés contiennent le caractère '\0'. Caractère non valide qui doit être remplacé par un blanc.

Résultat

La réponse `getPropertiesResponse` retourne un tableau `getPropertiesReturn` contenant les zones d'informations suivantes :

occurrenceToken
Identificateur d'occurrence.

owner ID du propriétaire défini pour l'application.

authorityGroup
Nom du groupe de droits d'accès auquel appartient l'application.

containingMonitoredJob
L'application comprend au moins une opération surveillée. Peut être `true` ou `false`.

Si le service n'a pas été exécuté avec succès, `getPropertiesResponse` retourne l'une des erreurs décrites dans «Gestion des erreurs», à la page 38.

setProperties (z/OS)

Décrit le service Web setProperties des occurrences d'application Tivoli Workload Scheduler for z/OS.

Description

Utilisez ce service pour définir des propriétés supplémentaires pour une occurrence d'application.

Paramètres d'entrée

engineName
Nom du contrôleur Tivoli Workload Scheduler for z/OS.

jobStreamId
Identificateur d'occurrence dans le plan en cours. Notez que les identificateurs d'objet du plan retournés contiennent le caractère '\0'. Caractère non valide qui doit être remplacé par un blanc.

properties
Tableau contenant les propriétés que vous souhaitez définir. Ces propriétés peuvent être :

priority
Priorité d'exécution. La valeur est compris entre 1 (la plus faible) et 9 (la plus urgente).

isMonitored
L'occurrence comprend les opérations surveillées. La valeur est `true` ou `false`.

startTime
Heure à laquelle l'occurrence doit démarrer au format `AAAAMMJJ HHMM` ou en millisecondes.

lateststartTime
Dernière heure à laquelle l'occurrence doit démarrer au format `AAAAMMJJ HHMM` ou en millisecondes.

lateststartAction
Action qui sera entreprise si l'occurrence dépasse sa dernière heure de début. Peut être CANCEL, CONTINUE ou SUPPRESS.

deadlineTime
Heure à laquelle l'occurrence doit être exécutée au format `AAAAMMJJ HHMM` ou en millisecondes.

Résultat
Si le service a été exécuté avec succès, la réponse `setPropertiesResponse` retourne l'identificateur du flot de travaux modifié et il retourne l'une des erreurs décrites dans «Gestion des erreurs», à la page 38.

getJobsList (z/OS)

Décrit le service Web getJobsList des occurrences d'application Tivoli Workload Scheduler for z/OS.

Description
Utilisez ce service pour obtenir une liste des opérations - et leurs propriétés - constituant une occurrence d'application.

Paramètres d'entrée

engineName
Nom du contrôleur Tivoli Workload Scheduler for z/OS.

jobStreamId
Identificateur d'occurrence dans le plan en cours. Notez que les identificateurs d'objet du plan retournés contiennent le caractère '\0'. Caractère non valide qui doit être remplacé par un blanc.

Résultat
La réponse `getJobsListResponse` retourne un tableau `getJobsListReturn`

contenant la liste des opérations contenues dans l'occurence. Pour chaque opération, les détails suivants sont également répertoriés :

occurrenceToken
Identificateur de l'occurrence contenant l'opération.

extendedStatus
Code de statut étendu affecté à l'opération.

errorCode
Code d'erreur terminant l'opération.

operationCommand
Peut être : BD = Bind shadow; job EX = Execute operation; KJ = Kill operation; KR = Kill recovery job; MH = Hold operation; MR = Release operation; NP = NOP operation; PN = Prompt reply no; PY = Prompt reply yes; UN = Un-NOP operation.

authorityGroup
Nom du groupe de droits d'accès auquel appartient l'opération.

cleanUpStatus
Peut être : Blank=none C=Completed E=Ended in error I=Initiated O=Avail opinfo R=Request opinfo S=Started W=Waiting opinfo

latestOutPassed
L'opération a dépassé sa dernière heure de début et est en retard. Peut être `true` ou `false`.

latestOut
Heure la plus tardive à laquelle, calculée à partir de l'heure de création de plan, à laquelle l'opération peut être lancée pour respecter l'échéance (HHMM).

actualArrival
Heure d'exécution réelle de l'opération (HHMM).

actualEnd
Heure à laquelle l'opération est signalée comme exécutée ou terminée par une erreur (HHMM).

Si le service n'a pas été exécuté avec succès, `getJobsListResponse` retourne l'une des erreurs décrites dans «Gestion des erreurs», à la page 38.

cancel (z/OS)

Décrit le service Web cancel des occurrences d'application Tivoli Workload Scheduler for z/OS.

Description
Utilisez ce service pour supprimer une occurrence d'application à partir du plan en cours.

Paramètres d'entrée

engineName
Nom du contrôleur Tivoli Workload Scheduler for z/OS.

jobStreamId
Identificateur d'occurrence dans le plan en cours. Notez que les identificateurs d'objet du plan retournés contiennent le caractère '\0'. Caractère non valide qui doit être remplacé par un blanc.

isPending
L'occurrence a le statut en attente. Elle peut prendre la valeur `true` ou `false`.

Résultat
La réponse `cancelResponse` est vide si le service a été exécuté avec succès ; sinon, elle retourne l'une des erreurs décrites dans «Gestion des erreurs», à la page 38.

Informations complémentaires

Explique comment obtenir des informations supplémentaires concernant l'utilisation des services Web.

Pour plus d'informations sur la gestion des fichiers `WSDL` afin de créer votre propre interface utilisateur basée sur des services Web, reportez-vous au Redbook IBM *IBM Redbooks: WebSphere Version 5.1 Application Developer 5.1.1 Web Services Handbook*, *SG24-6891* :

- Ce Redbook IBM décrit le nouveau concept de services Web de différents points de vue. Il présente les blocs de constitution principaux sur lesquels reposent les services Web. Les normes et nouveaux concepts y sont présentés et examinés.
- Bien que ces concepts soient décrits indépendamment du fournisseur, ce manuel présente également le point de vue d'IBM et illustre, à l'aide d'applications de démonstration adaptées, comment les services Web peuvent être mis en oeuvre avec la gamme de produits IBM, en particulier WebSphere Application Server Version 5.1 et WebSphere Studio Application Developer Version 5.1.1.
- Ce manuel est la mise à jour principale du Redbook IBM Web Services Wizardry with WebSphere Studio Application Developer, SG24-6292 et de WebSphere Version 5 Web Services Handbook, SG24-6891-00.

Pour accéder à cette publication, suivez ce lien : http://www.redbooks.ibm.com/abstracts/sg246891.html.

Remarques

Le présent document peut contenir des informations ou des références concernant certains produits, logiciels ou services IBM non annoncés dans ce pays. Pour plus de détails, référez-vous aux documents d'annonce disponibles dans votre pays, ou adressez-vous à votre partenaire commercial IBM. Toute référence à un produit, logiciel ou service IBM n'implique pas que seul ce produit, logiciel ou service puisse être utilisé. Tout autre élément fonctionnellement équivalent peut être utilisé, s'il n'enfreint aucun droit d'IBM. Il est de la responsabilité de l'utilisateur d'évaluer et de vérifier lui-même les installations et applications réalisées avec des produits, logiciels ou services non expressément référencés par IBM.

IBM peut détenir des brevets ou des demandes de brevet couvrant les produits mentionnés dans le présent document. La remise de ce document ne vous donne aucun droit de licence sur ces brevets ou demandes de brevet. Si vous désirez recevoir des informations concernant l'acquisition de licences, veuillez en faire la demande par écrit à l'adresse suivante :

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 U.S.A.

Pour le Canada, veuillez adresser votre courrier à :

IBM Director of Commercial Relations
IBM Canada Ltd
3600 Steeles Avenue East
Markham, Ontario
L3R 9Z7 Canada

Les informations sur les licences concernant les produits utilisant le jeu de caractères codé sur deux octets peuvent être obtenues en contactant le département de propriété intellectuelle IBM de votre pays ou en envoyant toute demande par écrit à l'adresse suivante :

Intellectual Property Licensing
Législation sur la propriété intellectuelle et légale
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japon

Le paragraphe suivant ne s'applique ni au Royaume-Uni, ni dans aucun pays dans lequel il serait contraire aux lois locales:

LES INFORMATIONS SONT LIVREES EN L'ETAT SANS AUCUNE GARANTIE EXPLICITE OU IMPLICITE. IBM DECLINE NOTAMMENT TOUTE RESPONSABILITE RELATIVE A CES INFORMATIONS EN CAS DE CONTREFAÇON, AINSI QU'EN CAS DE DEFAUT D'APTITUDE A L'EXECUTION D'UN TRAVAIL DONNE.

Certaines juridictions n'autorisent pas l'exclusion des garanties implicites ou explicites, auquel cas l'exclusion ci-dessus ne vous sera pas applicable.

Le présent document peut contenir des inexactitudes ou des coquilles. Il est mis à jour périodiquement. Chaque nouvelle édition inclut les mises à jour. IBM peut, à tout moment et sans préavis, modifier les produits et logiciels décrits dans ce document.

Les références à des sites Web non-IBM sont fournies à titre d'information uniquement et n'impliquent en aucun cas une adhésion aux contenus de ces sites Web. Les éléments figurant sur ces sites Web ne font pas partie des éléments du présent produit IBM et l'utilisation de ces sites relève de votre seule responsabilité.

IBM pourra utiliser ou diffuser, de toute manière qu'elle jugera appropriée et sans aucune obligation de sa part, tout ou partie des informations qui lui seront fournies.

Les licenciés souhaitant obtenir des informations permettant : (i) l'échange des données entre des logiciels créés de façon indépendante et d'autres logiciels (dont celui-ci), et (ii) l'utilisation mutuelle des données ainsi échangées, doivent adresser leur demande à :

IBM Corporation
2Z4A/101
11400 Burnet Road
Austin, TX 78758 U.S.A.

Ces informations peuvent être soumises à des conditions particulières, prévoyant notamment le paiement d'une redevance.

Le logiciel sous licence décrit dans ce document et tous les éléments sous licence disponibles s'y rapportant sont fournis par IBM conformément aux dispositions de l'ICA , des Conditions internationales d'utilisation des logiciels IBM ou de tout autre accord équivalent.

Le présent document peut contenir des exemples de données et de rapports utilisés couramment dans l'environnement professionnel. Ces exemples mentionnent des noms fictifs de personnes, de sociétés, de marques ou de produits à des fins illustratives ou explicatives uniquement. Toute ressemblance avec des noms de personnes, de sociétés ou des données réelles serait purement fortuite.

Marques

IBM, le logo IBM et ibm.com sont des marques d'International Business Machines Corporation aux Etats-Unis et/ou dans certains autres pays. Si ces marques et d'autres marques d'IBM sont accompagnées d'un symbole de marque (® ou ™), ces symboles signalent des marques d'IBM aux Etats-Unis à la date de publication de ce document. Ces marques peuvent aussi être des marques enregistrées ou de droit commun dans d'autres pays. Une liste des marques d'IBM est disponible sur le Web sur "Informations sur le copyright et les marques" à l'adresse suivante : http://www.ibm.com/legal/copytrade.shtml

Java ainsi que tous les logos et toutes les marques incluant Java sont des marques et des marques déposées de Oracle et/ou de ses filiales.

Microsoft et Windows sont des marques de Microsoft Corporation aux Etats-Unis et/ou dans certains autres pays.

UNIX est une marque enregistrée aux Etats-Unis et dans certains autres pays, dont seule la société The Open Group peut concéder la licence.

Index

O

P

Q

R

S

T

U

Numéro de programme : 5698-T08, 5698-WSH et 5698-WSE

SC11-6398-03