IBM Tivoli Workload Scheduler

Guide d'utilisation et de référence

Version 9.2

SC11-2009-10

IBM Tivoli Workload Scheduler

Guide d'utilisation et de référence

Version 9.2

SC11-2009-10

Important
Avant d'utiliser le présent document et le produit associé, prenez connaissance des informations générales figurant à la section «Remarques», à la page 857.

Réf. US : SC32-1274-15

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

Cette édition s'applique à la version 9.2.0 de Tivoli Workload Scheduler (numéro de programme 5698-WSH) et 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 cette publication

IBM® Tivoli Workload Scheduler simplifie la gestion de systèmes dans des environnements distribués en intégrant des fonctions de gestion de systèmes. Tivoli Workload Scheduler planifie, automatise et contrôle le traitement de la charge de travail de votre entreprise dans son intégralité. Le document *Tivoli Workload Scheduler - Guide d'utilisation et de référence* fournit des informations détaillées sur l'interface de ligne de commande, le langage de planification et les commandes d'utilitaire de Tivoli Workload Scheduler.

Nouveautés

Prenez connaissance des nouveautés de cette édition.

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

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

Nouveautés de cette publication

Prenez connaissance des nouveautés de cette publication.

Les informations suivantes ont été ajoutées ou modifiées dans la présente publication :

•

A qui s'adresse cette publication

Prenez connaissance du public concerné par cette publication.

Le présent manuel s'adresse aux personnes chargées de la planification, de l'ordonnancement, de la surveillance ou de la gestion d'un environnement de planification de la charge de travail. Il s'agit généralement d'opérateurs et d'administrateurs Tivoli Workload Scheduler.

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 plus d'informations 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 sur le support

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

Si vous rencontrez un problème 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 de problèmes 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 problème, 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*.

Conventions du présent document

Prenez connaissance des conventions utilisées dans cette publication.

Le présent document utilise plusieurs conventions pour identifier les actions et les termes spécifiques, les chemins et les commandes propres au système d'exploitation, la syntaxe des commandes et les graphiques des marges.

Conventions typographiques

Ce document utilise les conventions typographiques suivantes :

Gras

- Commandes en minuscules et commandes en minuscules et majuscules difficiles à distinguer du reste du texte
- Commandes d'interface (cases à cocher, boutons de commande, boutons d'option, sélecteurs rotatifs, zones, dossiers, icônes, zones de liste, éléments des zones de liste, listes à plusieurs colonnes, conteneurs, options de menu, noms de menu, onglets, fiches techniques), libellés (par exemple **Conseil :** et **Remarques relatives au système d'exploitation :**)
- Mot clés et paramètres dans le texte

Italique

- Mot définis dans le texte
- Insistance sur les mots (mots en tant que tels)
- Nouveaux termes dans le texte (sauf dans une liste de définitions)
- Variables et valeurs que vous devez fournir

`Espacement fixe`

- Exemples et exemples de code
- Noms de fichiers, mots clés de programmation et autres éléments difficiles à distinguer du reste du texte
- Invites et texte des messages destinés à l'utilisateur
- Texte à entrer par l'utilisateur
- Valeurs des arguments ou options de commande

Variables et chemins spécifiques au système d'exploitation

Pour les variables d'environnement et la notation des répertoires, ce document utilise la convention UNIX, excepté lorsque le contexte spécifique de l'exemple de chemin d'accès est Windows.

Lorsque vous utilisez une ligne de commande Windows, remplacez **$***variable* par **%***variable***%** pour les variables d'environnement, et remplacez toutes les barres obliques (*/*) par des barres obliques inversées (\) dans les chemins d'accès aux répertoires. Les noms des variables d'environnement diffèrent parfois dans les environnements Windows et UNIX. Par exemple, %TEMP% sous Windows correspond à $tmp sous UNIX.

Remarque : Si vous utilisez le shell bash sur un système Windows, vous pouvez utiliser les conventions UNIX.

Syntaxe des commandes

Lorsqu'il décrit des commandes, le présent document utilise la syntaxe suivante :

Tableau 1. Syntaxe des commandes

Convention de syntaxe	Description	Exemple
Nom de la commande	Premier terme ou série de caractères consécutifs.	**conman**

Tableau 1. Syntaxe des commandes (suite)

Convention de syntaxe	**Description**	**Exemple**
Crochets ([ ])	Les informations entre crochets ([ ]) sont facultatives. Toutes les autres informations doivent être fournies.	[**-file** *fichier_définition*]
Accolades ({ })	Les accolades ({ }) désignent un ensemble d'options s'excluant mutuellement, lorsqu'une seule option est requise.	{**-prompts** | **-prompt** *nom_invite* }
Trait de soulignement (_)	Les traits de soulignement (_) relient plusieurs mots dans une variable.	*nom_invite*
Barre verticale (|)	Les options s'excluant mutuellement sont séparées par une barre verticale (|). Vous pouvez entrer une seule des options séparées par la barre verticale au cours de la même utilisation de la commande.	{**-prompts** | **-prompt** *nom_invite* }
Gras	Le texte en **gras** désigne les informations qui doivent être saisies sur la ligne de commande exactement telles quelles. Cela s'applique aux noms de commande et aux options invariables.	**composer add** *file_name*
Italique	Le texte en italique est variable et doit être remplacé par les informations qu'il représente. Dans l'exemple de droite, l'utilisateur remplacera la variable nom_fichier par le nom du fichier approprié.	*file_name*
Points de suspension (...)	Les points de suspension (...) placés après une option indiquent que l'option peut être répétée plusieurs fois avec des valeurs différentes. Ils peuvent être employés à l'intérieur ou à l'extérieur des crochets.	[**–x** *file_name*]... Les points de suspension à l'extérieur des crochets indiquent que la variable **–x** *file_name* est facultative et peut être répétée comme suit : ***–x*** *nom_fichier1* ***–x*** *nom_fichier2****–x*** *nom_fichier3* [**–x** *file_name*...] Les points de suspension placés entre crochets signifient que la variable **–x** *file_name* est facultative et qu'elle peut être répétée comme suit : ***–x*** *nom_fichier1 nom_fichier2 nom_fichier3* **–x** *file_name* [**–x** *file_name*]... Les points de suspension utilisés sous cette syntaxe indiquent que vous devez définir au moins une fois la variable **–x** *file_name*.

Chapitre 1. Présentation du produit Tivoli Workload Scheduler

Grâce à IBM Tivoli Workload Scheduler, vous pouvez gérer l'environnement de production et automatiser plusieurs activités de surveillance. Tivoli Workload Scheduler gère le traitement des travaux, résout les interdépendances, lance et suit le déroulement des travaux. Dans la mesure où les travaux sont exécutés immédiatement après résolution de leurs dépendances, le délai d'inactivité est réduit au minimum et la productivité est largement optimisée. Si un travail échoue, Tivoli Workload Scheduler assure le processus de reprise en sollicitant peu (voire pas du tout) l'aide de l'opérateur.

Le présent chapitre comprend les sections suivantes :

- «Explication des concepts de base»
- «Interfaces utilisateur Tivoli Workload Scheduler», à la page 27
- «Démarrage de la production», à la page 29

Explication des concepts de base

La présente section décrit les concepts de base de Tivoli Workload Scheduler et comprend les sections suivantes :

- «Objets de base de données Tivoli Workload Scheduler»
- «Réseau Tivoli Workload Scheduler», à la page 20
- «Configuration de l'environnement d'exécution Tivoli Workload Scheduler», à la page 21
- «Définition d'activités de planification à l'aide de Tivoli Workload Scheduler», à la page 22
- «Gestion des activités de planification de production avec Tivoli Workload Scheduler», à la page 26

Objets de base de données Tivoli Workload Scheduler

Cette section présente les objets de base de données Tivoli Workload Scheduler que vous utilisez. Les objets de base de données suivants sont décrits :

- Travail, voir «Travail», à la page 2
- Flot de travaux, voir «Flot de travaux», à la page 2
- Application de charge de travail, voir «Application de charge de travail», à la page 3
- Cycle d'exécution, voir «Cycle d'exécution», à la page 4
- Groupe de cycle d'exécution, voir «Groupe de cycle d'exécution», à la page 5
- Agenda, voir «Agenda», à la page 10
- Invite, voir «Invite», à la page 10
- Poste de travail, voir «Poste de travail», à la page 11
- Classe de poste de travail, voir «Classe de postes de travail», à la page 15
- Domaine, voir «Domaine», à la page 16
- Règle d'événement, voir «Règle d'événement», à la page 19
- Ressource, voir «Ressource», à la page 19
- Paramètre, voir «Paramètre», à la page 19
- Table de variables, voir «Table de variables», à la page 20

Travail

Un *travail* est une unité spécifiant une action, par exemple une sauvegarde des données hebdomadaire, à effectuer sur des postes de travail spécifiques du réseau Tivoli Workload Scheduler. Dans un environnement réparti Tivoli Workload Scheduler en revanche, ils peuvent être définis de manière indépendante à partir de flots de travaux ou dans une définition de flot de travaux.

Les types de travail peuvent être divisés entre des travaux Tivoli Workload Scheduler et des types de travaux avec options avancées. Les types de travail existants sont des scripts génériques ou des commandes que vous personnalisez en fonction de vos besoins. Les types de travaux avec options avancées sont des travaux conçus pour effectuer des opérations spécifiques, telles que des opérations de base de données, de transfert de fichier, Java et de service Web. Vous planifiez ces types de travaux uniquement sur les agents, les pools et les pools dynamiques.

Si vous souhaitez optimiser la fonction dynamique lors de la planification des types de travaux avec options avancées, vous les planifiez sur des pools et des pools dynamiques, ce qui affecte dynamiquement le travail aux meilleures ressources disponibles. Si n'avez besoin de définir que des types de travaux avec options avancées, sans utiliser la fonction de planification dynamique, vous planifiez ces travaux sur un agent spécifique, sur lequel le travail s'exécute de façon statique.

Que le moteur Tivoli Workload Scheduler soit distribué ou basé sur z/OS, vous pouvez définir localement un *travail reflet* pour mapper l'instance de travail distant qui s'exécute sur un moteur Tivoli Workload Scheduler différent.

Pour plus d'informations sur la manière de définir des travaux, voir «Définition de travail», à la page 169.

Pour plus d'informations sur la définition des postes de travail, voir «Définition de poste de travail», à la page 149.

Flot de travaux

Un *flot de travaux* est une séquence de travaux à exécuter, ainsi que des heures, des priorités et d'autres dépendances déterminant l'ordre du traitement. A chaque flot de travaux est affectée une heure d'exécution symbolisée par un cycle d'exécution avec un agenda, un ensemble de dates ou des fréquences de répétition.

Dépendances dans un environnement réparti :
Vous pouvez avoir des dépendances entre des travaux et des flots de travaux. Il peut s'agir de :

Dépendances internes
Certaines dépendances sont établies entre des travaux appartenant au même flot de travaux.

Dépendances externes
Il s'agit de dépendances entre des flots de travaux, entre des flots de travaux et des travaux appartenant à d'autres flots, ou encore entre des travaux appartenant à divers flots de travaux.

Dépendances interréseaux
Dépendances par rapport à des travaux ou des flots de travaux s'exécutant sur un autre réseau Tivoli Workload Scheduler. Les dépendances interréseaux nécessitent un poste de travail agent réseau afin de communiquer avec le réseau Tivoli Workload Scheduler externe.

Les dépendances sur des ressources sont prises en charge par Tivoli Workload Scheduler tant dans les environnements répartis que z/OS.

Pour plus d'informations sur la définition des flots de travaux, voir «Définition de flot de travaux», à la page 236.

Application de charge de travail

Une application de charge de travail représente un ou plusieurs flots de travaux ainsi que tous les travaux référencés qui peuvent être partagés avec d'autres environnements Tivoli Workload Scheduler par un processus de déploiement simple.

Une application de charge de travail est un objet de base de données Tivoli Workload Scheduler qui agit en tant que conteneur pour un ou plusieurs flots de travaux. Vous pouvez utiliser une applications de charge de travail pour normaliser une solution d'automatisation de charge de travail de sorte que la solution puisse être réutilisée dans un ou plusieurs environnements Tivoli Workload Scheduler automatisant de ce fait des processus métier.

Vous pouvez préparer un modèle de application de charge de travail dans un environnement Tivoli Workload Scheduler source, puis l'exporter de sorte qu'il puisse être déployé dans un environnement cible. Le processus d'exportation extrait de l'environnement source tous les éléments nécessaires pour faire une copie la solution dans un autre environnement. Il produit un fichier compressé contenant un certain nombre de fichiers nécessaires pour importer application de charge de travail dans l'environnement cible. Ces fichiers contiennent une définition des objets dans l'environnement source extrait de la base de données Tivoli Workload Scheduler. Pour les éléments qui dépendent de la topologie de l'environnement cible, une configuration manuelle est nécessaire. Par exemple, les définitions extraites de l'environnement source contiennent des références aux postes de travail qui n'existent pas dans l'environnement cible. Pour cette raison, avant de procéder à l'importation, un mappage de certains des éléments doit être fait en associant le nom de l'objet dans l'environnement cible.

Le modèle exporté de l'application de charge de travail contient des définitions ou des références pour tous les objets suivants :

- Flots de travaux
- Travaux
- Postes de travail, classes de postes de travail
- Agendas
- Invites
- Cycles d'exécution
- Groupes de cycle d'exécution
- Ressources
- Dépendances interréseaux
- Dépendances externes

Pour plus d'informations sur la manière de définir les modèles application de charge de travail, voir «Définition des applications de charge de travail», à la page 297.

Cycle d'exécution

Un ***cycle d'exécution*** indique pour quels jours l'exécution d'un flot de travaux est planifiée. Un cycle est défini pour un flot de travaux spécifique et ne peut pas être utilisé par plusieurs flots de travaux. Vous pouvez définir les types suivants de cycle d'exécution :

Simple
: Jours définis par l'utilisateur pendant lesquels un flot de travaux sera exécuté. Un cycle d'exécution simple est défini pour un flot de travaux spécifique et il ne peut pas être utilisé par d'autres flots de travaux.

Quotidien
: Cycle d'exécution indiquant que l'exécution du flot de travaux s'effectue selon une certaine fréquence et un type de jour que vous avez définis. Par exemple, il peut s'exécuter quotidiennement, tous les trois jours ou uniquement les jours ouvrés.

Hebdomadaire
: Cycle d'exécution spécifiant les jours de la semaine au cours desquels un flot de travaux est exécuté. Par exemple, en faisant appel à ce cycle, vous pouvez demander que l'exécution ait lieu chaque lundi, mercredi et vendredi.

Mensuel
: Cycle d'exécution indiquant que l'exécution du flot de travaux s'effectue selon une certaine date ou un jour que vous avez défini mensuellement. Par exemple, il peut s'exécuter tous les premiers et deuxièmes jours du mois, tous les deux mois ou tous les premiers lundis et deuxièmes mardis du mois, tous les trois mois.

Annuel
: Cycle d'exécution indiquant que l'exécution du flot de travaux s'effectue, par exemple, tous les ans ou tous les trois ans.

Basé sur des décalages
: Cycle d'exécution qui utilise une combinaison de périodes et décalages définis par l'utilisateur. Par exemple, un décalage de 3 sur une période de 15 jours est le troisième jour à partir du début de la période. Pour les périodes cycliques, il est préférable d'utiliser des cycles d'exécution basés sur des décalages. Ce terme est utilisé comme tel uniquement dans Tivoli Workload Scheduler for z/OS, mais le concept s'applique également au produit distribué.

Basé sur des règles
: Cycle d'exécution qui utilise des règles fondées sur des listes de nombres ordinaux, des types de jour, des intervalles d'agenda communs (ou noms de période dans Tivoli Workload Scheduler for z/OS). Par exemple, le dernier jeudi de chaque mois. Les cycles d'exécution basés sur des règles sont fondés sur des périodes conventionnelles, telles que les mois, les semaines et les jours de la semaine. Dans Tivoli Workload Scheduler for z/OS, les cycles d'exécution peuvent également reposer sur des périodes que vous définissez, telles que des semestres. Ce terme est utilisé comme tel uniquement dans Tivoli Workload Scheduler for z/OS, mais le concept s'applique également au produit distribué. Vous pouvez également indiquer une règle afin d'établir quand le flot de travail doit s'exécuter si l'exécution définie tombe un jour chômé.

Ces types de cycle d'exécution peuvent être inclusifs ou exclusifs, c'est-à-dire :

Inclusif
Cycle d'exécution indiquant les jours et heures auxquels l'exécution d'un flot de travaux a été planifiée. Les cycles d'exécution inclusifs ne sont pas prioritaires par rapport aux cycles d'exécution exclusifs.

Exclusif
Cycle d'exécution indiquant les jours et heures auxquels un flot de travaux ne peut pas être exécuté. Les cycles d'exécution exclusifs sont prioritaires par rapport aux cycles d'exécution inclusifs.

Groupe de cycle d'exécution

Vous pouvez éventuellement définir un groupe de cycle d'exécution pour votre flot de travaux au lieu, ou en plus, d'un certain nombre de cycles d'exécution uniques.

Un groupe de cycle d'exécution est une liste de cycles d'exécution qui sont combinés pour produire un ensemble de dates d'exécution.

À l'aide des groupes de cycle d'exécution, vous pouvez bénéficier des avantages suivants :

Un groupe de cycle d'exécution est un objet de base de données distinct
Il est défini par lui-même et peut être mis en correspondance avec un ou plusieurs flots de travaux. Il n'est pas défini en tant qu'élément d'un flot de travaux spécifique comme les cycles d'exécution uniques.

Le même groupe de cycle d'exécution peut être utilisé sur différents flots de travaux
Ceci améliore la convivialité globale des cycles d'exécution, parce que vous pouvez indiquer le même groupe de cycle d'exécution dans des flots de travaux multiples, ce qui évite de recourir à des définitions de cycle d'exécution multiples pour les mêmes règles de planification.

Les groupes de cycle d'exécution étendent l'utilisation des cycles d'exécution exclusifs
Des cycles d'exécution exclusifs (ou négatifs) sont utilisés pour générer des occurrences négatives, qui identifient les jours où un flot de travaux serait normalement planifié mais où il n'est pas nécessaire. La somme des cycles d'exécution exclusifs est soustraites de celle des cycles d'exécution inclusifs. Une occurrence négative annule toujours toutes les occurrences positives correspondantes et vous pouvez indiquer une occurrence négative seulement si l'equivalent positif existe déjà. Une correspondance exacte des jours, ainsi que les restrictions de temps, est nécessaire entre les cycles d'exécution exclusifs et inclusifs pour que l'annulation puisse se produire. Les groupes de cycle d'exécution ajoutent beaucoup de flexibilité en permettant aux utilisateurs d'appliquer des cycles d'exécution exclusifs à un sous-ensemble de cycles positifs plutôt qu'à tous les cycles. Regroupez vos cycles d'exécution dans des *sous-ensembles* de sorte que les cycles d'exécution exclusifs puissent être appliqués seulement aux occurrences positives générées par les cycles d'exécution appartenant au même ensemble.

Les cycles d'exécution doivent être organisés dans des *sous-ensembles* au sein d'un groupe de cycle d'exécution. Les sous-ensembles sont toujours dans une relation logique **OU** les uns avec les autres. Le résultat du groupe de cycle d'exécution est toujours une date ou un ensemble de dates ; il ne peut pas être négatif.

Par exemple, vous voudrez peut-être que votre flot de travaux s'exécute chaque jour du mois excepté le dernier jour du mois. Mais, vous voulez

également qu'il soit planifié le dernier jour de l'année (le dernier jour de décembre). Vous pouvez définir un groupe de cycle d'exécution à l'aide de sous-ensembles, comme suit :

Sous-ensemble 1

- **Cycle d'exécution 1** - cycle d'exécution inclusif chaque jour du mois
- **Cycle d'exécution 2** - cycle d'exécution exclusif le plus dernier jour du mois

Sous-ensemble 2

- **Cycle d'exécution 3** - cycle d'exécution inclusif le 31 décembre

où, le cycle d'exécution 2 annule le dernier jour de chaque mois dans le sous-ensemble 1, alors que le cycle d'exécution 3 génère le 31 décembre en tant que date distincte ; par conséquent, vous pouvez planifier le flot de travaux le 31 décembre.

Les groupes de cycle d'exécution permettent d'utiliser des cycles d'exécution logiques ET entre des cycles d'exécution individuels dans le sous-ensemble

Par défaut, les cycles d'exécution dans un sous-ensemble sont dans une relation logique **OU** mais vous pouvez la changer en relation **ET** logique si le résultat du groupe de cycle d'exécution est une date ou un ensemble positif de dates (inclusif). Pour chaque cycle d'exécution, vous pouvez indiquer l'un ou l'autre des opérateurs (**ET**, **OU**), et obtenir le comportement suivant :

1. Tous les cycles d'exécution du groupe qui sont dans la relation *ET* sont calculés en premier. Le résultat de ce calcul est une date ou un ensemble de dates.
2. Puis, tous les cycles d'exécution dans la relation *OU* sont ajoutés au résultat de l'étape précédente.

Un comportement semblable est appliqué aux cycles d'exécution inclusifs et exclusifs pour déterminer la date limite ou l'ensemble de dates d'un groupe.

Inclusif (A)
Cycle d'exécution basé sur des règles. Sélectionnez les jours où le flot de travaux doit être exécuté s'ils appartiennent à tous les types A de l'ensemble de cycles d'exécution.

Exclusif (D)
Cycle d'exécution basé sur des règles d'exclusion. Sélectionnez les jours où le flot de travaux NE doit PAS être exécuté s'ils appartiennent à tous les types D de l'ensemble de cycles d'exécution.

Par exemple, vous pouvez ajouter deux conditions ensemble :

```
Exécuter le mercredi "ET" le 8ème jour ouvrable du mois.
```

De cette façon, les seules dates planifiées sont n'importe quel 8ème jour ouvrable du mois qui tombe un mercredi.

Compatibilité complète avec des cycles d'exécution *traditionnels*

Les cycles d'exécution *traditionnels* indiqués dans la définition du flot de travaux peuvent référencer des groupes de cycle d'exécution, avec la possibilité d'indiquer le ou les décalages (tout comme avec les périodes pour z/OS ou avec les agendas pour les systèmes répartis).

Un ensemble de dates (débuts d'intervalle) est créé automatiquement au niveau de cycle d'exécution directement (inclusivement ou exclusivement avec des décalages, ou dans la règle. Il s'agit d'un processus en deux étapes avec les cycles d'exécution suivants :

1. Définir l'"événement métier" clé, tel que la Fin du mois, à l'aide des cycles d'exécution et des règles de jours libres
2. Définir les règles qui utilisent les dates de l'"événement métier" comme intervalles par rapport auxquels l'autre exécution par lots peut être planifiée.

Par exemple, vous disposez d'un *processus de fin du mois* qui s'exécute le dernier vendredi d'un mois, mais qui est repoussé au jour ouvrable suivant, sauf en décembre où il s'exécute le 3ème vendredi du mois. Cette règle de planification peut être définie avec quelques règles, cycles d'exécution et règles de plage horaire libres.

Deux jours ouvrables avant fin du mois, vous devez exécuter un processus de pré-validation pour régler les problèmes métier avant l'exécution. Vous ne pouvez pas sélectionner le dernier mercredi du mois, parce qu'avec certains mois, celui-ci peut se trouver après le dernier vendredi. De même, si le dernier vendredi était un jour chômé, le dernier mercredi ne se situera pas 2 jours ouvrables avant celui-ci, car la règle de jour chômé s'applique UNIQUEMENT au jour où la règle entre en vigueur.

Il se peut que beaucoup d'autres exécutions par lot doivent également être exécutées un certain nombre de jours avant ou après la fin du mois, mais les mêmes restrictions s'appliquent.

Vous pouvez maintenant définir le travail pour qu'il s'exécute relativement à un élément défini par une combinaison de cycles d'exécution et pour libérer les règles quotidiennes.

Utilisation des agendas avec des cycles d'exécution au sein d'un groupe de cycle d'exécution

Vous pouvez éventuellement indiquer plusieurs agendas pour calculer la définition de jours ouvrables et chômés pour un cycle d'exécution. L'agenda principal est utilisé pour calculer qui les jours ouvrables qui sont valides, et un agenda secondaire est utilisé pour calculer les dates chômées spécifiques. Si les dates calculées selon l'agenda secondaire correspondent aux jours ouvrables dans l'agenda principal, le travail est planifié ; s'ils ne correspondent pas, le travail n'est pas planifié.

Par exemple, une société globale qui exécute la charge de travail aux Etats-Unis pour beaucoup d'autres pays nécessite un grand nombre de combinaisons d'agenda pour s'assurer que les travaux par lots fonctionnent seulement un jour qui est un jour ouvrable aux Etats-Unis et dans l'autre pays. L'agenda peut être défini au niveau du flot de travaux et, s'il n'est pas spécifié, un agenda par défaut est utilisé. Cependant, lorsqu'il est défini, l'agenda au niveau du cycle d'exécution peut être utilisé comme agenda secondaire et l'agenda de flot de travaux (ou agenda par défaut) peut être utilisé comme agenda principal.

Par exemple, le calendrier principal peut être *JOURS OUVRABLES*, défini du LUNDI au VENDREDI à l'exclusion des dates de congés des Etats-Unis. Vous pourriez également devoir calculer les exécutions de travail basées sur l'agenda *HKWORK*, qui est défini du lundi au vendredi, à l'exclusion des dates de congés de Hong Kong. Le travail pourrait avoir plusieurs planifications :

- Exécution les jours ouvrables, mais pas le dernier jour ouvrable et pas le lundi
- Exécution le lundi, mais pas le dernier jour ouvrable
- Exécution le dernier jour ouvrable

Puisque chaque planification est calculée en fonction de l'agenda *WORKHK*, elle est également contrôlée par rapport à l'agenda *WORKDAYS* pour s'assurer qu'elle est planifiée un jour ouvrable des Etats-Unis.

Utilisation des restrictions de temps avec les groupes de cycle d'exécution

Vous pouvez indiquer des contraintes de temps pour définir le moment où le traitement doit commencer ou la durée au bout de laquelle le traitement ne doit plus commencer. Pour ce faire, vous pouvez associer les *restrictions de temps* au travail, aux flots de travaux, aux cycles d'exécution et aux groupes de cycle d'exécution. Quand vous définissez une restriction de temps, vous obtenez fondamentalement un *moment*. Puisque vous pouvez associer des restrictions de temps à plusieurs objets, la hiérarchie suivante indique l'ordre dans lequel les différentes restrictions de temps sont prises en compte pour définir réellement quand commencer le traitement :

1. Restriction de temps définie dans le cycle d'exécution du flot de travaux
2. Restriction de temps définie dans le flot de travaux
3. Restriction de temps définie dans le cycle d'exécution contenu dans le groupe de cycle d'exécution associé au flot de travaux.
4. Restriction de temps définie dans le groupe de cycle d'exécution associé au flot de travaux.
5. Début de la journée

Cela implique les points suivants :

Les restrictions de temps du flot de travaux

Substituent et sont prioritaires *sur toutes les autres restrictions de temps* définies dans les cycles d'exécution ou dans les groupes de cycle d'exécution associés au flot de travaux.

Aucune restriction de temps dans le flot de travaux ni dans le groupe de cycle d'exécution

Le groupe génère seulement une date qui est le *Début de journée*. Si des décalages et des règles de jours chômés doivent être calculés, le calcul commence toujours à partir du *Début de journée*.

Restrictions de temps dans le groupe de cycle d'exécution (pas dans le flot de travaux)

Les restrictions de temps (et le décalage possible) sont calculés à partir du *Début de journée* et la date et l'heure qui en résulte indique le début du traitement.

Exemples

Tableau 2. Scénario 1. Aucune restriction de temps dans le groupe de cycle d'exécution

Groupe de cycle d'exécution	Date planifiée	Premier démarrage
Groupe de cycle d'exécution	10/24	10/24
Groupe de cycle d'exécution avec décalage (+ 3 jours)	10/27 (samedi)	10/27/ (samedi)
Groupe de cycle d'exécution avec règle de jours chômés	10/29/ (lundi)	0/29/ (lundi)

Tableau 2. Scénario 1. Aucune restriction de temps dans le groupe de cycle d'exécution (suite)

Groupe de cycle d'exécution	Date planifiée	Premier démarrage
Cycle d'exécution dans le flot de travaux avec restriction de temps		
Cycle d'exécution dans le flot de travaux avec décalage de + 4 jours ouvrables	11/02 (vendredi)	11/02 (vendredi)
Cycle d'exécution dans le flot de travaux avec règle de jours chômés	11/02 (vendredi)	11/02 (vendredi)
Cycle d'exécution dans le flot de travaux avec le début au plus tôt +1 13h	11/02 (vendredi)	11/03 (samedi) 13h
Cycle d'exécution dans le flot de travaux sans restrictions de temps		
Cycle d'exécution dans le flot de travaux avec décalage de + 4 jours ouvrables	11/02 (vendredi)	11/02 (vendredi) Début de journée
Cycle d'exécution dans le flot de travaux avec règle de jours chômés	11/02 (vendredi)	11/02 (vendredi) Début de journée

Tableau 3. Scénario 2. Restriction de temps dans le groupe de cycle d'exécution sans décalage

Groupe de cycle d'exécution	Date planifiée	Premier démarrage
Groupe de cycle d'exécution	10/24	10/24
Groupe de cycle d'exécution avec décalage d'agenda (+ 3 jours)	10/27/ (samedi)	10/27/ (samedi)
Groupe de cycle d'exécution avec règle de jours chômés	10/29/ (lundi)	0/29/ (lundi)
Cycle d'exécution dans le flot de travaux avec restriction de temps		
Cycle d'exécution dans le flot de travaux avec décalage de + 4 jours ouvrables	11/02 (vendredi)	11/02 (vendredi)
Cycle d'exécution dans le flot de travaux avec règle de jours chômés	11/02 (vendredi)	11/02 (vendredi)
Cycle d'exécution dans le flot de travaux avec le début au plus tôt +1 13h	11/02 (vendredi)	11/03 (samedi) 13h
Cycle d'exécution dans le flot de travaux sans restrictions de temps		
Cycle d'exécution dans le flot de travaux avec décalage de + 4 jours ouvrables	11/02 (vendredi)	11/02 (vendredi) Début de journée
Cycle d'exécution dans le flot de travaux avec règle de jours chômés	11/02 (vendredi)	11/02 (vendredi) Début de journée

Tableau 4. Scénario 3. Restriction de temps dans le groupe de cycle d'exécution avec décalage (+1 12h00)

Groupe de cycle d'exécution	**Date planifiée**	**Premier démarrage**
Groupe de cycle d'exécution	10/24	10/24
Groupe de cycle d'exécution avec décalage d'agenda (+ 3 jours)	10/27/ (samedi)	10/27/ (samedi)
Groupe de cycle d'exécution avec règle de jours chômés	10/29/ (lundi)	10/29/ (lundi)
Groupe de cycle d'exécution avec décalage +1 12h00	10/29/ (lundi)	10/30 12h00 (mardi)
Cycle d'exécution dans le flot de travaux avec restriction de temps		
Cycle d'exécution dans le flot de travaux avec décalage de + 4 jours ouvrables	11/02 (vendredi)	11/02 (vendredi)
Cycle d'exécution dans le flot de travaux avec règle de jours chômés	11/02 (vendredi)	11/02 (vendredi)
Cycle d'exécution dans le flot de travaux avec le début au plus tôt +1 13h	11/02 (vendredi)	11/03 (samedi) 13h
Cycle d'exécution dans le flot de travaux sans restrictions de temps		
Cycle d'exécution dans le flot de travaux avec décalage de + 4 jours ouvrables	11/02 (vendredi)	11/03 12:00 (samedi)
Cycle d'exécution dans le flot de travaux avec règle de jours chômés	11/02 (vendredi)	11/03 12:00 (samedi)

Disponibilité de la commande GENDAYS au niveau du groupe de cycle d'exécution

En utilisant GENDAYS, vous pouvez vérifier le résultat de la combinaison de tous les cycles d'exécution du groupe.

Agenda

Un *agenda* est une liste de dates indiquant si un flot de travaux s'exécute et quand.

Un agenda peut également être sélectionné pour être utilisé comme agenda des *jours chômés* dans un flot de travaux. Un agenda des jours chômés est un agenda affecté à un flot de travaux afin de représenter les jours où le flot de travaux et les travaux associés ne s'exécutent pas. Il peut également être configuré pour considérer les samedis et/ou les dimanches comme jours ouvrables. Par convention, de nombreux utilisateurs définissent un agenda des jours chômés nommé *holidays*, dans lequel le samedi et le dimanche sont spécifiés comme jours chômés.

Pour plus d'informations sur la manière de définir des agendas, voir «Définition d'agenda», à la page 216.

Invite

Une *invite* est un texte s'affichant sous forme de message et interrompant le traitement d'un travail ou flot de travaux jusqu'à ce qu'une réponse affirmative soit reçue. (La réponse peut être indiquée manuellement par l'opérateur ou

automatiquement par une action de règle d'événement.) Une fois la réponse à l'invite obtenue, le traitement reprend. Les invites peuvent servir de dépendances dans les travaux ou les flots de travaux. Vous pouvez également utiliser des invites pour informer un opérateur de l'exécution d'une tâche spécifique. Dans ce cas, aucune réponse n'est nécessaire de la part de l'opérateur.

Il existe trois types d'invites :

Invite globale ou nommée
Invite définie dans la base de données en tant qu'objet de planification. Elle est identifiée par un nom unique et peut être utilisée par n'importe quel travail ou flot de travaux.

Invite locale ou ad-hoc
Invite définie au sein d'une définition de travail ou de flot de travaux. Elle ne possède pas de nom et elle n'est pas définie en tant qu'objet de planification dans la base de données ; elle ne peut donc pas être utilisée par d'autres travaux ou flots de travaux.

Invite de reprise ou fin anormale
Type spécifique d'invite dont l'utilisation est réservée aux fins de travail anormales. La réponse à cette invite détermine le résultat du travail ou du flot de travaux auquel le travail appartient. Une invite de reprise peut également être associée à une action et à un type spécial de travail appelé *travail de reprise*.

Pour plus d'informations sur la manière de définir des invites, voir «Définition d'invite», à la page 224.

Poste de travail

Cette section fournit des informations concernant l'utilisation des postes de travail pour la planification des travaux et des flots de travaux. Si vous souhaitez en savoir plus sur les postes de travail dans le cadre de la planification de votre réseau, toutes les informations dont vous avez besoin se trouvent dans le manuel *Tivoli Workload Scheduler - Guide d'installation et de planification*.

Le système informatique sur lequel vous exécutez vos travaux et flots de travaux est appelé *poste de travail*. Lorsque vous définissez un travail ou un flot de travaux dans la base de données Tivoli Workload Scheduler, vous identifiez les définitions de poste de travail pour les systèmes informatiques physiques ou virtuels sur lesquels l'exécution de votre travail est planifiée. Les postes de travail peuvent être regroupés de manière logique selon des *classes de postes de travail* et organisés dans une hiérarchie de *domaines*, laquelle est gérée par des *gestionnaires de domaine*.

Pour plus d'informations relatives aux classes de poste de travail, voir «Classe de postes de travail», à la page 15. Pour les domaines, voir «Domaine», à la page 16.

Lorsque vous créez une définition de poste de travail pour un système appartenant à votre réseau, vous déterminez un ensemble de caractéristiques qui identifie de manière unique ce système et affecte son mode d'exécution des travaux. Voici quelques exemples de ces caractéristiques : l'adresse IP du poste de travail, s'il est ou non derrière un pare-feu, si ses communications sont sécurisées ou non, le fuseau horaire du poste de travail et l'identité de son gestionnaire de domaine.

Les postes de travail du réseau de planification Tivoli Workload Scheduler peuvent non seulement traiter les travaux et les flots de travaux, mais aussi remplir d'autres

rôles. Lors de la conception du réseau, des rôles ont été affectés aux postes de travail en fonction des besoins de votre activité. Vous trouverez ci-dessous une liste complète des rôles de poste de travail :

Gestionnaire de domaine maître
Poste de travail agissant comme concentrateur de gestion pour le réseau. Il gère l'ensemble de vos objets de planification. Ce poste de travail est enregistré dans la base de données Tivoli Workload Scheduler en tant que poste **maître**.

Gestionnaire de domaine maître de sauvegarde
Poste de travail qui peut servir de sauvegarde pour le gestionnaire de domaine maître, en cas d'incident. Il s'agit véritablement d'un gestionnaire de domaine maître, en attente d'activation. Son utilisation est facultative. Pour en savoir plus sur le basculement vers un gestionnaire de domaine maître de sauvegarde, consultez le manuel *Tivoli Workload Scheduler - Guide d'administration*. Ce poste de travail doit être installé comme gestionnaire de domaine maître configuré en tant que copie de sauvegarde. Ce poste de travail est enregistré dans la base de données Tivoli Workload Scheduler en tant qu'**agent tolérant aux pannes**.

Gestionnaire de domaine
Poste de travail qui contrôle un domaine et qui partage des responsabilités de gestion pour une partie du réseau Tivoli Workload Scheduler. Il est installé comme un agent, puis il est configuré comme un poste de travail de domaine maître lorsque vous définissez le poste de travail dans la base de données. Ce poste de travail est enregistré dans la base de données Tivoli Workload Scheduler en tant que poste **manager** (gestionnaire).

Gestionnaire de domaine dynamique
Composant installé d'un réseau Tivoli Workload Scheduler réparti jouant le rôle de concentrateur de gestion dans un domaine. Toutes les communications à destination et en provenance des agents d'un domaine passent par le gestionnaire de domaine dynamique. Lorsque vous installez un gestionnaire de domaine dynamique, les types de postes de travail suivants sont créés dans la base de données :

fta Composant agent tolérant aux pannes configuré manuellement en tant que gestionnaire de domaine

broker
Composant serveur courtier

agent Composant agent dynamique

gestionnaire de domaine dynamique de secours
Poste de travail qui peut servir de sauvegarde pour le gestionnaire de domaine dynamique lorsque des problèmes surviennent. Il s'agit véritablement d'un gestionnaire de domaine dynamique, en attente d'activation. Son utilisation est facultative. Pour en savoir plus sur le basculement vers un gestionnaire de domaine dynamique de sauvegarde, consultez le manuel *Tivoli Workload Scheduler - Guide d'administration*. Lorsque vous installez un gestionnaire de domaine dynamique, les types de postes de travail suivants sont créés dans la base de données :

fta Composant agent tolérant aux pannes.

broker
Composant serveur courtier

agent Composant agent dynamique

Agent tolérant aux pannes
Poste de travail qui reçoit et traite des travaux. En cas de problème de communication avec son gestionnaire de domaine, il peut exécuter les travaux en local. Il est installé comme un agent, puis il est configuré comme un poste de travail de l'agent tolérant aux pannes lorsque vous définissez le poste de travail dans la base de données. Ce poste de travail est enregistré dans la base de données Tivoli Workload Scheduler en tant qu'**agent tolérant aux pannes**.

Agent standard
Poste de travail qui reçoit et exécute uniquement les travaux sous le contrôle de son gestionnaire de domaine. Il est installé comme un agent, puis il est configuré comme un poste de travail d'agent standard lorsque vous définissez le poste de travail dans la base de données. Ce poste de travail est enregistré dans la base de données Tivoli Workload Scheduler en tant que **s-agent** (agent standard).

Agent étendu
Poste de travail sur lequel une méthode d'accès Tivoli Workload Scheduler for Applications a été installée en tant que pont, de manière à pouvoir planifier des travaux sur des applications SAP R/3, Oracle E-Business Suite , PeopleSoft et z/OS ou des applications personnalisées. Il doit être physiquement hébergé par un gestionnaire de domaine maître, un gestionnaire de domaine, un agent standard ou un agent tolérant aux pannes puis défini en tant qu'agent étendu dans la base de données. Pour plus d'informations, voir *Tivoli Workload Scheduler for Applications - Guide d'utilisation*. Ce poste de travail est enregistré dans la base de données Tivoli Workload Scheduler en tant que poste **x-agent** (agent étendu).

Courtier de charge de travail
Poste de travail où s'exécutent à la fois des types de travaux existants et des types de travaux avec options avancées. Il s'agit du serveur courtier installé avec le gestionnaire de domaine maître et le gestionnaire de domaine dynamique. Il peut héberger l'un ou plusieurs des postes de travail suivants :

- Agent étendu
- Moteur distant
- Pool
- Pool dynamique
- Agent. Cette définition comprend les agents suivants :
  - Agent
  - agent Tivoli Workload Scheduler for z/OS
  - Agent pour z/OS

  Pour plus d'informations sur l'agent et l'agent Tivoli Workload Scheduler for z/OS, voir *Scheduling Workload Dynamically*. Pour plus d'informations sur le agent pour z/OS, voir *Planification avec l'agent pour z/OS*.

Ce poste de travail est enregistré dans la base de données Tivoli Workload Scheduler en tant que **broker** (courtier).

Agent dynamique
Poste de travail qui gère une grande variété de types de travaux, comme par exemple les travaux d'une base de données ou d'un protocole FTP spécifiques, en plus des types de travaux existants. Ce poste de travail est automatiquement créé et enregistré dans la base de données Tivoli Workload Scheduler lorsque vous installez l'agent. L'agent est hébergé par

le poste de travail Workload Broker. Etant donné que les processus d'installation et d'enregistrement sont effectués automatiquement, lorsque vous affichez l'agent dans Dynamic Workload Console, il apparaît mis à jour par le Resource Advisor Agent. Vous pouvez regrouper les agents dans des pools et des pools dynamiques. Ce poste de travail est enregistré dans la base de données Tivoli Workload Scheduler en tant qu'**agent**.

Dans une configuration simple, les agents dynamiques se connectent directement à un gestionnaire de domaine maître ou à un gestionnaire de domaine dynamique. Toutefois, dans des topologies de réseau plus complexes, si la configuration de réseau empêche le gestionnaire de domaine maître ou le gestionnaire de domaine dynamique de communiquer directement avec le agent dynamique, vous pouvez configurer vos agents dynamiques pour utiliser une passerelle locale ou distante.

Remarque : Si l'option globale `enAddWorkstation` vaut `"yes"`, la définition de poste de travail d'agent dynamique est automatiquement ajoutée au plan après que le processus d'installation a créé le poste de travail d'agent dynamique dans la base de données.

Pool Un regroupement de postes de travail logiques est un ensemble d'agents avec des caractéristiques matérielles ou logicielles similaires, sur lequel sont soumis les travaux. Tivoli Workload Scheduler équilibre les travaux entre les agents du pool et réaffecte automatiquement les travaux aux agents disponibles lorsqu'un agent n'est plus disponible. Pour créer un pool d'agents dans votre environnement Tivoli Workload Scheduler, définissez un poste de travail de type **pool** hébergé par le poste de travail Workload Broker, puis sélectionnez les agents que vous souhaitez ajouter au pool. Vous pouvez définir le pool à l'aide de Dynamic Workload Console ou de la commande **composer**. Ce poste de travail est enregistré dans la base de données Tivoli Workload Scheduler en tant que **pool**. Lorsque vous créez un pool dans votre environnement Tivoli Workload Scheduler, une ressource logique portant le même nom est automatiquement créée dans Dynamic Workload Broker. La ressource logique est utilisée pour lier et regrouper les agents appartenant au même pool, et en tant que condition requise pour les travaux planifiés dans le pool Tivoli Workload Scheduler. Supposons que ces objets de base de données sont deux objets différents. Si vous renommez le pool Tivoli Workload Scheduler, cette modification n'est pas apportée à la ressource logique Dynamic Workload Broker.

Pool dynamique
Un regroupement de postes de travail logiques est un ensemble d'agents, défini dynamiquement en fonction des exigences de ressources que vous indiquez et qui est hébergé par le poste de travail Workload Broker. Par exemple, si vous avez besoin d'un poste de travail sollicitant peu l'unité centrale et que Windows est installé pour exécuter votre travail, vous spécifiez les exigences suivantes à l'aide de Dynamic Workload Console ou de la commande **composer**. Lorsque vous sauvegardez l'ensemble d'exigences, un nouveau poste de travail est automatiquement créé dans la base de données Tivoli Workload Scheduler. Ce poste de travail mappe tous les agents de votre environnement qui répondent aux exigences spécifiées. Le pool généré est mis à jour dynamiquement dès qu'un nouvel agent approprié devient disponible. Les travaux planifiés sur ce poste de travail héritent automatiquement des exigences définies pour le poste de

travail. Ce poste de travail est hébergé par le poste de travail Workload Broker et est enregistré dans la base de données Tivoli Workload Scheduler en tant que **d-pool** (pool dynamique).

Moteur distant
Poste de travail qui gère l'échange des informations sur la résolution des dépendances croisées entre votre environnement et un moteur Tivoli Workload Scheduler for z/OS distant (contrôleur) ou un moteur Tivoli Workload Scheduler (gestionnaire de domaine maître ou gestionnaire de domaine maître de sauvegarde). Ce poste de travail est hébergé par le poste de travail Workload Broker et est enregistré dans la base de données Tivoli Workload Scheduler en tant que **rem-eng** (moteur distant).

Remarque : Si vous planifiez la modification des types de postes de travail, prenez en compte les règles suivantes :

- Vous pouvez modifier les postes de travail d'agent tolérant aux pannes, d'agent standard, d'agent étendu, de gestionnaire de domaine et de courtier de charge de travail dynamiques par tout type de poste de travail, à l'exception de l'agent dynamique, du pool, du pool dynamique et du moteur distant.
- Vous ne pouvez pas modifier le type de l'agent dynamique, du pool, du pool dynamique et du moteur distant.

Pour plus d'informations sur la définition des postes de travail, voir «Définition de poste de travail», à la page 149.

Classe de postes de travail

Les postes de travail peuvent être regroupés par classe. Une *classe de postes de travail* est un groupe de postes de travail ayant des caractéristiques de planification de travail similaires. Une classe peut contenir un nombre illimité de postes de travail et un poste de travail peut figurer dans plusieurs classes. Les travaux et flots de travaux peuvent être définis pour s'exécuter dans une classe de postes de travail spécifique. Cela permet de faciliter l'exécution de travaux et flots de travaux sur plusieurs postes de travail.

Par exemple, vous pouvez définir les types suivants de classes de postes de travail :

- Classes de postes de travail regroupant les postes selon votre structure de services interne qui vous permet de définir un travail qui s'exécutera sur tous les postes d'un même service
- Classes de postes de travail regroupant les postes selon les logiciels qui y sont installés et qui vous permettent de définir un travail qui s'exécutera sur tous les postes ayant une même application.
- Classes de postes de travail regroupant les postes selon le rôle de l'utilisateur associé, de sorte que vous puissiez définir un travail qui s'exécutera sur tous les postes appartenant, par exemple, aux responsables.

Dans cet exemple, un poste donné peut se trouver dans une classe définie pour son service, dans une autre spécifique à son utilisateur et dans diverses classes selon les logiciels qui y sont installés.

Les postes de travail peuvent être regroupés par domaine. Cela est fait lorsque votre réseau est configuré. Le nom de domaine n'est pas un critère de sélection pour le choix de l'emplacement d'exécution d'un travail, il est donc possible que vous deviez reproduire votre structure de domaines avec des classes de postes de travail si vous souhaitez planifier l'exécution d'un travail sur tous les postes d'un domaine.

Pour plus d'informations sur les domaines, voir «Domaine»

Pour plus d'informations relatives à la définition des classes de poste de travail, voir «Définition de classe de postes de travail», à la page 166.

Domaine

Tous les postes de travail d'un réseau Tivoli Workload Scheduler répartis sont organisés en un ou plusieurs *domaines*, chacun d'eux étant composé d'un ou de plusieurs agents et d'un gestionnaire de domaine faisant office de concentrateur de gestion. La plupart des communications à destination et en provenance des agents d'un domaine passe par le gestionnaire de domaine.

Tous les réseaux ont un domaine maître dans lequel le gestionnaire de domaine est le gestionnaire de domaine maître. Il gère la base de données de tous les objets de planification du domaine et les fichiers de configuration centraux. Le gestionnaire de domaine maître génère le plan et crée et distribue le fichier Symphony. En outre, il gère les journaux et les rapports du réseau.

Vous pouvez organiser tous les agents de votre réseau dans un seul domaine ou dans plusieurs domaines.

Réseaux à domaine unique

Un réseau à domaine unique comprend un gestionnaire de domaine maître et un certain nombre d'agents. Un réseau à domaine unique est présenté ci-après. Ce type de réseau convient parfaitement aux sociétés ayant un nombre limité de sites et de fonctions métier. Toutes les communications du réseau sont acheminées via le gestionnaire de domaine maître. Lorsque le réseau couvre un seul site, vous ne vous préoccupez que de la fiabilité de votre réseau local et de la quantité de trafic qu'il peut gérer.

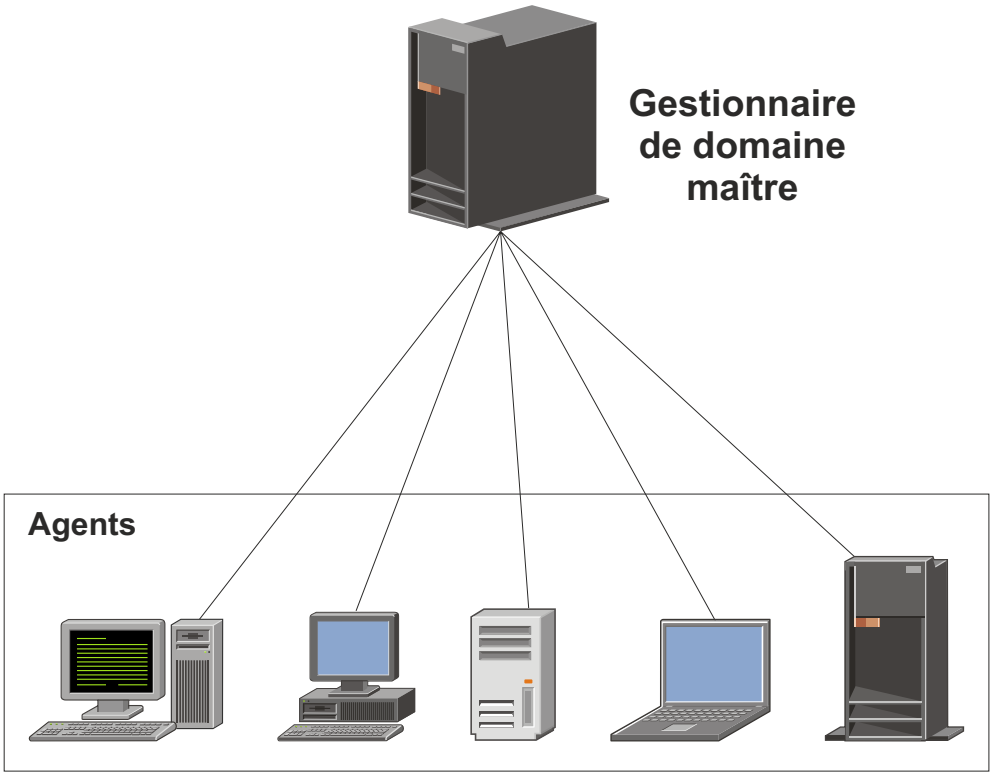


Figure 1. Réseau à domaine unique

Réseaux à domaines multiples

Les réseaux à domaines multiples conviennent particulièrement aux sociétés s'étendant sur plusieurs sites, régions ou fonctions métier. Un réseau à domaines multiples comprend un gestionnaire de domaine maître, un certain nombre de gestionnaires de domaine de niveau inférieur et divers agents dans chacun des domaines. Les agents communiquent uniquement avec leurs gestionnaires de domaine et les gestionnaires de domaine communiquent, quant à eux, avec leurs gestionnaires de domaine parent. La hiérarchie de domaine peut comporter un nombre indéfini de niveaux.

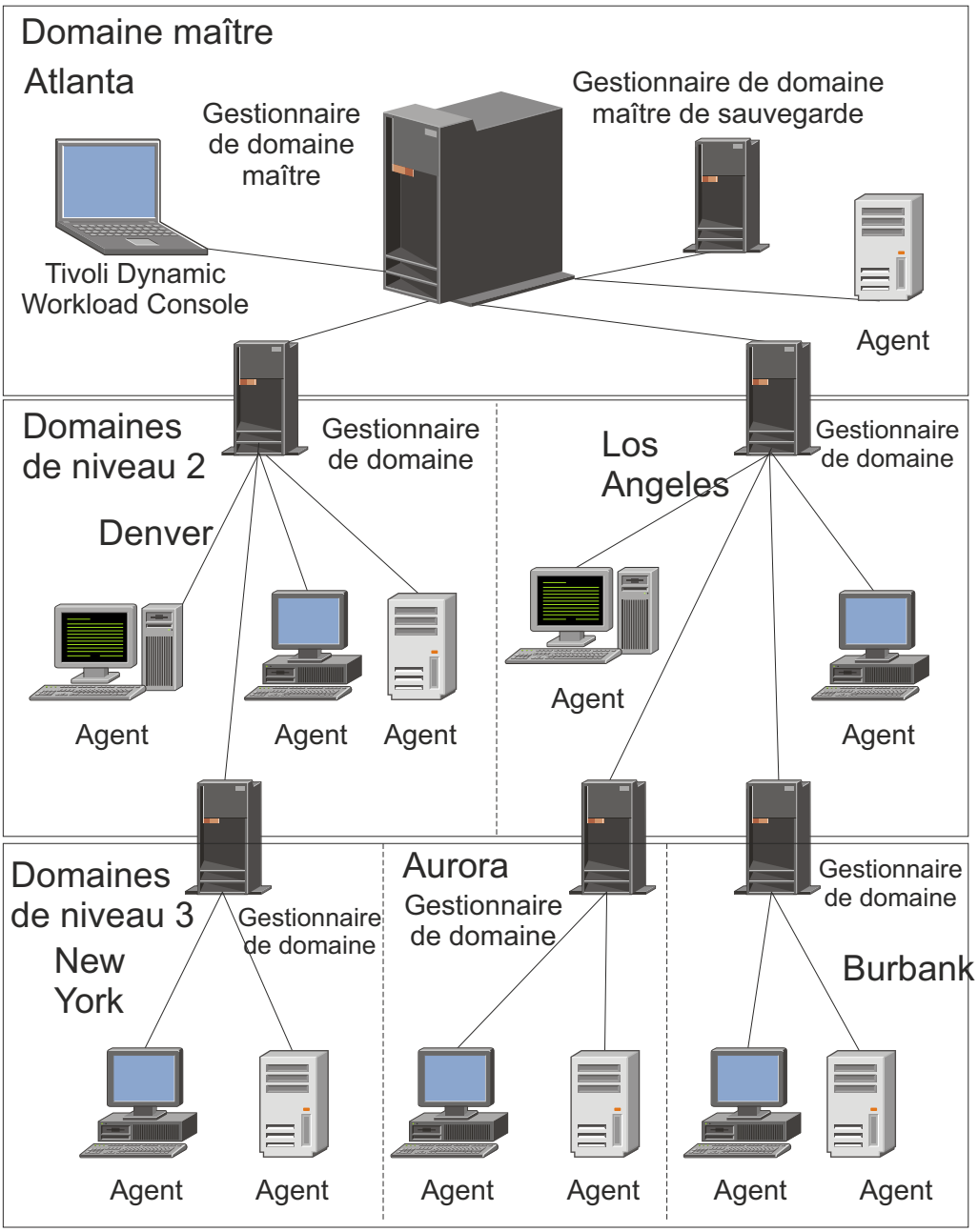


Figure 2. Réseau à domaines multiples

Dans cet exemple, le gestionnaire de domaine maître est situé à Atlanta. Il contient les fichiers de base de données utilisés pour documenter les objets de planification et assurer la distribution du fichier Symphony auprès de ses agents et gestionnaires de domaine à Denver et Los Angeles. Les gestionnaires de domaine de Denver et Los Angeles distribuent ensuite à leur tour le fichier Symphony auprès de leurs agents et gestionnaires de domaine subordonnés à New York, Aurora et Burbank. Le gestionnaire de domaine d'Atlanta est chargé de la diffusion des informations inter-domaines sur l'ensemble du réseau.

Toute communication à destination et en provenance du gestionnaire de domaine de Boulder est acheminée via son gestionnaire de domaine parent situé à Denver. Si des calendriers ou des travaux du domaine de Boulder

dépendent de calendriers ou de travaux du domaine d'Aurora, ces dépendances sont résolues par le gestionnaire de domaine de Denver. La plupart des dépendances interagents sont gérées en local par les gestionnaires de domaine de niveau inférieur, ce qui permet de réduire le trafic sur le réseau de manière notable.

Il est possible de modifier dynamiquement l'infrastructure de domaine à mesure que le réseau se développe. Le déplacement d'un poste de travail vers un autre domaine implique simplement de modifier le nom de domaine dans sa définition de base de données.

Tableau 5. Astuce

Conseil :	Il n'est pas possible de planifier l'exécution de travaux ou de flots de travaux sur tous les postes de travail dans un domaine en identifiant ce dernier dans la définition de travail ou de flot de travaux. Pour ce faire, créez une *classe de postes de travail* contenant tous les postes de travail dans le domaine.

Pour plus d'informations relatives aux classes de poste de travail, voir «Classe de postes de travail», à la page 15

Pour plus d'informations sur la manière de définir des domaines, voir «Définition de domaine», à la page 167.

Règle d'événement

Une *règle d'événement* définit un ensemble d'actions qui s'exécutent lorsqu'il existe des conditions d'événement spécifiques. La définition d'une règle d'événement fait référence à des événements et des actions du déclencheur.

Pour plus d'informations sur la définition des règles d'événement, voir «Définition de règles d'événement», à la page 136.

Ressource

Une *ressource* est une ressource système physique ou logique que vous utilisez comme dépendance pour les travaux et flots de travaux. Un travail ou un flot de travaux avec une dépendance de ressource ne peut démarrer tant que la quantité de ressource spécifiée n'est pas disponible.

Pour plus d'informations sur la manière de définir des ressources, voir «Définition de ressource», à la page 225.

Paramètre

Un *paramètre* est un objet auquel vous affectez différentes valeurs qui seront remplacées dans les travaux ou flots de travaux, soit à partir de valeurs de la base de données soit au moment de l'exécution. Les paramètres sont utiles lorsque vous avez des valeurs qui changent en fonction de votre travail ou flot de travaux. Les définitions de travail ou de flot de travaux utilisant des paramètres sont automatiquement mises à jour avec la valeur appropriée au début du cycle de production. Utilisez les paramètres en remplacement des valeurs répétitives lors de la définition de travaux ou flots de travaux. Par exemple, si vous utilisez des paramètres pour les ouvertures de session utilisateur et les noms de fichiers script dans les définitions de travaux, ainsi que pour les dépendances de fichier et d'invite, vous utilisez des valeurs permettant une gestion centralisée dans la base de données du gestionnaire maître.

Pour plus d'informations relatives à la définition des paramètres, voir «Définition des variables et des paramètres», à la page 217.

Utilisateur

Un *Utilisateur* est le nom d'utilisateur utilisé comme valeur de connexion pour plusieurs définitions de travail du système d'exploitation. Les utilisateurs doivent être définis dans la base de données.

Si vous planifiez un travail sur un agent, un pool ou un pool dynamique, le travail s'exécute avec l'utilisateur défini sur le pool ou pool dynamique. Cependant, l'utilisateur doit exister sur tous les postes de travail du pool ou du pool dynamique où vous planifiez d'exécuter le travail.

Remarque : Si l'option globale `enAddUser` vaut `"yes"`, la définition d'utilisateur est automatiquement ajoutée au plan après la création ou la modification de la définition d'utilisateur dans la base de données.

Table de variables

Une *table de variables* est une table comportant plusieurs variables et leurs valeurs. Tous les paramètres globaux, désormais appelés *variables*, sont inclus dans au moins une table de variables.

Il n'est pas nécessaire de créer des tables de variables pour utiliser des variables. En effet, le planificateur fournit une table de variables par défaut.

Toutefois, il est possible que vous souhaitiez définir une variable portant le même nom mais dont les valeurs diffèrent, selon le moment et le contexte d'utilisation. Pour ce faire, vous affectez différentes valeurs à la même variable dans différentes tables de variables. Vous pouvez alors utiliser le même nom de variable dans différentes définitions de travail ou lorsque vous définissez des dépendances de fichier et d'invite. Les tables de variables peuvent être affectées au niveau des cycles d'exécution, des flot de travaux et des postes de travail.

Les tables de variables peuvent s'avérer particulièrement utiles pour les définitions de travail, notamment lorsque vous utilisez une définition de travail en tant que modèle pour un travail appartenant à plusieurs flots de travaux. Par exemple, vous pouvez affecter différentes valeurs à la même variable et réutiliser la même définition de travail dans plusieurs flots de travaux.

Pour plus d'informations sur la définition des tables de variables, voir «Définition de table de variables», à la page 222.

Réseau Tivoli Workload Scheduler

Un réseau Tivoli Workload Scheduler est composé d'un ensemble de ***postes de travail*** reliés entre eux sur lesquels vous exécutez un traitement des travaux par lots à l'aide des fonctionnalités de gestion Tivoli Workload Scheduler.

Les postes de travail communiquent via TCP/IP et une technologie de stockage avant transfert qui permet de préserver la cohérence et la tolérance aux pannes à travers le réseau. En d'autres termes, si un poste de travail n'est pas connecté, toutes les informations sont stockées dans le fichier de messages et ne sont envoyées qu'une fois la liaison rétablie.

Le réseau Tivoli Workload Scheduler comprend un ou plusieurs domaines, chacun d'eux comportant un poste de travail ***gestionnaire de domaine*** jouant le rôle de concentrateur de gestion et un ou plusieurs postes de travail ***agents***.

Il existe quatre types d'agent : ***standard***, ***tolérant aux pannes***, ***étendu*** et ***Workload Broker***. Les agents standard et tolérants aux pannes peuvent être définis sur des ordinateurs UNIX et Windows. Les agents étendus sont des définitions logiques hébergées chacune par un poste de travail physique et permettent de traiter les travaux lorsqu'aucun agent n'est installé. Par exemple, ils sont utilisables avec Peoplesoft, SAP R/3, z/OS, CA-7, JES, OPC, Oracle EBS et VMS mais vous pouvez également les installer sur des systèmes UNIX et Windows. Les agents Workload Broker sont des postes de travail qui gèrent le cycle de vie des travaux de type Tivoli Workload Scheduler Workload Broker dans Tivoli Dynamic Workload Broker.

Un ***poste de travail de moteur distant*** est un autre type de poste de travail que vous pouvez définir dans votre réseau. Ce type de poste de travail permet de gérer la communication avec un moteur Tivoli Workload Scheduler distant, distribué ou basé sur z/OS, afin de gérer les dépendances des travaux locaux à partir des travaux définis sur le moteur distant. Pour plus d'informations, voir Chapitre 19, «Définition et gestion des dépendances croisées», à la page 681.

Pour plus d'informations sur les postes de travail, voir «Définition de poste de travail», à la page 149.

Dans la topologie hiérarchique de Tivoli Workload Scheduler, le ***gestionnaire de domaine maître*** est le gestionnaire de domaine du domaine supérieur. Toutes les tâches de configuration de la production et la génération du ***plan de production*** sont effectuées sur le gestionnaire de domaine maître. Un plan de production contient toutes les activités de gestion des travaux devant être effectuées à travers le réseau Tivoli Workload Scheduler au cours d'une période déterminée. Une copie du plan de production est distribuée à partir du gestionnaire de domaine maître aux autres postes de travail. Sur chaque poste de travail, Tivoli Workload Scheduler lance ses propres travaux, en assure le suivi et envoie le statut du traitement des travaux au gestionnaire de domaine maître.

Pour plus d'informations sur les fonctionnalités Tivoli Workload Scheduler de gestion du plan, voir Chapitre 4, «Gestion du cycle de production», à la page 61

Configuration de l'environnement d'exécution Tivoli Workload Scheduler

La présente section vous offre un aperçu évolué de la configuration possible de l'environnement d'exécution Tivoli Workload Scheduler.

Définition des propriétés de configuration

Vous pouvez définir deux types de propriété pour configurer votre environnement d'exécution Tivoli Workload Scheduler : les propriétés définies sur le gestionnaire de domaine maître qui concernent le traitement sur tous les postes de travail du réseau Tivoli Workload Scheduler et les propriétés définies localement sur un poste de travail qui concernent uniquement le traitement sur ce poste de travail. Les premières sont gérées à l'aide du programme de ligne de commande Tivoli Workload Scheduler intitulé **optman**, tandis que les deuxièmes sont définies en local sur le poste de travail par personnalisation des fichiers de configuration **useropts**, **localopts** et **jobmanrc**.

Pour plus d'informations sur l'utilisation de la ligne de commande **optman** pour gérer les options globales et sur les options locales définies dans le fichier `localopts`, voir *IBM Tivoli Workload Scheduler - Guide d'administration*.

Pour plus d'informations sur les options locales définies dans le fichier `useropts`, voir «Configuration des options pour l'utilisation des interfaces utilisateur», à la page 59.

Configuration de la sécurité

Chaque fois que vous exécutez un programme Tivoli Workload Scheduler ou que vous appelez une commande Tivoli Workload Scheduler, les informations de sécurité sont lues à partir d'un fichier spécial appelé ***fichier de sécurité*** pour déterminer vos droits d'utilisateur. Ce fichier contient une ou plusieurs *définitions d'utilisateur*. Une définition d'utilisateur est un groupe d'un ou plusieurs utilisateurs autorisés ou non à exécuter des actions spécifiques sur des types d'objet de planification spécifiques.

L'utilisateur Tivoli Workload Scheduler principal, ***utilisateur_TWS***, est défini au moment de l'installation dans le fichier de sécurité. Il permet d'exécuter la procédure de configuration, de définir des propriétés et de gérer les définitions d'utilisateur dans le fichier de sécurité. Vous pouvez modifier le fichier de sécurité à tout moment de façon à répondre aux besoins de votre configuration système.

Pour plus d'informations sur la gestion des autorisations utilisateur, voir *Tivoli Workload Scheduler - Guide d'administration*.

Définition d'activités de planification à l'aide de Tivoli Workload Scheduler

Pour exécuter des activités de planification à l'aide de Tivoli Workload Scheduler, vous devez d'abord définir l'environnement que vous souhaitez gérer en termes d'*objets de planification* et de règles à appliquer lors de la planification d'opérations à exécuter sur ces objets. Ces informations sont stockées par Tivoli Workload Scheduler dans une base de données relationnelle DB2 ou Oracle, que nous appellerons ultérieurement ***base de données***.

Outre les définitions des objets de planification (les travaux, les flots de travaux, les ressources, les postes de travail, par exemple), la base de données contient également des statistiques relatives aux travaux et flots de travaux traités, ainsi que des informations relatives à l'utilisateur qui a créé un objet et à la dernière modification d'un objet. Vous pouvez gérer les définitions d'objets de planification dans la base de données soit à l'aide du programme de ligne de commande Tivoli Workload Scheduler nommé **composer**, soit à l'aide de l'interface graphique **Dynamic Workload Console**. Vous pouvez extraire de la base de données des statistiques ou des informations historiques concernant les travaux et les flots de travaux traités à l'aide :

- Des **utilitaires de rapport** de Tivoli Workload Scheduler à partir de la ligne de commande.
- De Dynamic Workload Console.
- Des vues de la base de données.

Pour plus d'informations relatives à la définition des objets de planification, voir Chapitre 8, «Définition d'objets dans la base de données», à la page 147.

Pour plus d'informations sur les commandes de l'utilitaire de génération d'états, voir Chapitre 15, «Extraction des états et des statistiques», à la page 609.

Pour de plus amples informations sur Dynamic Workload Console, reportez-vous à la documentation correspondante.

Pour plus d'informations sur les vues de la base de données, voir *IBM Tivoli Workload Scheduler - Vues de la base de données*.

Contrôle du traitement des travaux et des flots de travaux

Vous pouvez contrôler le traitement des travaux et des flots de travaux en définissant une ou plusieurs règles à partir de ce qui suit :

Définition de dépendances

Une ***dépendance*** est un prérequis devant être respecté pour que le traitement se poursuive. Vous pouvez définir des dépendances pour les travaux et les flots de travaux afin de garantir leur traitement dans l'ordre approprié. Dans l'environnement de planification distribué Tivoli Workload Scheduler, vous pouvez choisir parmi quatre différents types de dépendances :

- ***A l'achèvement des travaux et flots de travaux*** : un travail ou un flot de travaux, appelé *successeur*, ne doit pas commencer son traitement avant que d'autres travaux et flots de travaux, appelés *prédécesseurs*, ne se soient correctement exécutés. Pour plus d'informations, voir «follows», à la page 255.
- ***Ressource*** : un travail ou un flot de travaux a besoin d'une ou plusieurs ressources avant de pouvoir démarrer. Pour plus d'informations, voir «needs», à la page 266.
- ***Fichier*** : un travail ou un flot de travaux a besoin d'avoir accès à un ou plusieurs fichiers avant de pouvoir démarrer. Pour plus d'informations, voir «opens», à la page 273.
- ***Invite*** : un travail ou un flot de travaux doit attendre une réponse affirmative à une invite avant de pouvoir démarrer. Pour plus d'informations, voir «Définition d'invite», à la page 224 et «prompt», à la page 276.

Vous pouvez définir jusqu'à 40 dépendances pour un travail ou un flot de travaux.

Dans un réseau Tivoli Workload Scheduler, les dépendances peuvent se répercuter sur plusieurs postes de travail. Par exemple, vous pouvez rendre un travail `job1`, exécuté sur votre environnement local Tivoli Workload Scheduler `site1`, dépendant de l'aboutissement du travail `job2`, exécuté sur un environnement Tivoli Workload Scheduler distant `site2`. L'environnement de planification distant peut être soit des moteurs Tivoli Workload Scheduler for z/OS (contrôleur), soit d'autres moteurs Tivoli Workload Scheduler (gestionnaire de domaine maître). Deux types de dépendances implémentent cette exigence :

Dépendance interréseau

Il s'agit d'une implémentation simple et distribuée. Utilisez ce type de dépendance lorsque :

- L'environnement local Tivoli Workload Scheduler est distribué.
- Vous voulez rechercher une instance de travail prédécesseur distant uniquement dans le plan en cours d'exécution (plan de production) sur l'environnement distant.
- Vous avez besoin de faire correspondre une instance prédécesseur dans le plan distant et non *cette* instance de prédécesseur spécifique.
- Vous pouvez attendre que l'intervalle d'interrogation expire avant d'être mis à jour quant à la transition du statut du travail distant.
- Vous pouvez utiliser différentes syntaxes et configurations reposant sur l'environnement Tivoli Workload Scheduler distant distribué plutôt que sur z/OS.
- Vous acceptez d'utiliser un protocole de connexion propriétaire pour la communication avec le moteur distant.

Pour plus d'informations, voir Chapitre 18, «Gestion des dépendances interréseaux», à la page 671.

Dépendance croisée
Il s'agit d'un implémentation plus complète. Utilisez ce type de dépendance lorsque :

- Votre environnement local Tivoli Workload Scheduler peut être soit distribué, soit z/OS.
- Vous voulez également rechercher l'instance de prédécesseur distante parmi les instances planifiées qui ne sont pas encore incluses dans le plan en cours d'exécution sur le moteur distant.
- Vous recherchez la correspondance d'une instance de prédécesseur distante précise dans le plan du moteur distant. Pour ce faire, vous pouvez utiliser différents critères de recherche prêts à l'emploi.
- Vous souhaitez que votre dépendance soit mise à jour dès que l'instance de travail distant change de statut. Pour ce faire, le produit utilise des notifications asynchrones depuis le moteur distant sur le moteur local.
- Vous souhaitez utiliser les mêmes syntaxe et configuration, que l'environnement Tivoli Workload Scheduler local soit distribué ou z/OS.
- Vous voulez utiliser des connexions HTTP ou HTTPS pour la communication avec le moteur distant.

Pour plus d'informations, voir Chapitre 19, «Définition et gestion des dépendances croisées», à la page 681.

Définition des contraintes temporelles

Des ***contraintes temporelles*** peuvent être indiquées tant pour des travaux que pour des flots de travaux. Pour un cycle d'exécution spécifique, vous pouvez indiquer l'heure de début du traitement à l'aide du mot clé **at**, ou l'heure après laquelle le traitement ne démarre plus avec le mot clé **until**. En indiquant ces deux heures, vous définissez une période d'exécution d'un travail ou d'un flot de travaux. Les mots clés **at** et **until** correspondent à des dépendances temporelles.

Il est également possible de spécifier le paramètre de temps **schedtime** pour désigner l'heure de référence lors du calcul des dépendances de travaux et de flots de travaux. Vous pouvez également spécifier une ***fréquence de répétition*** pour que Tivoli Workload Scheduler lance par exemple le même travail toutes les 30 minutes entre 8:30 et 13:30.

Vous pouvez également indiquer une **durée maximale** ou une **durée minimale** pour un travail défini dans un flot de travaux. Si un travail est en cours d'exécution et que la durée maximale a été dépassée, le travail peut être abandonné ou peut continuer à s'exécuter. Si un travail ne s'exécute pas assez longtemps pour atteindre la durée minimale spécifiée, le travail peut être défini sur l'état Fin anormale, sur l'état Confirmer en attendant la confirmation de l'utilisateur ou il peut continuer à s'exécuter.

Remarque : La spécification de la durée minimale et maximale d'un travail défini au sein d'un flot de travaux génère un fichier Symphony qui occupe 512 octets de plus qu'un travail sans ces spécifications.

Pour plus d'informations, voir «at», à la page 242, «deadline», à la page 245, «every», à la page 247, «schedtime», à la page 277, «until», à la page 280, «maxdur», à la page 263 et «mindur», à la page 265.

Définition de la priorité d'un travail et de la priorité minimale d'un poste de travail

Tivoli Workload Scheduler possède son propre système de mise en file d'attente qui utilise des niveaux de ***priorité***. L'attribution d'une priorité à des travaux ou des flots de travaux vous offre un contrôle accru sur leur priorité et leur ordre d'exécution.

La ***priorité minimale de travail*** permet d'exercer un autre type de contrôle sur le traitement de travaux sur un poste de travail. Lorsqu'elle est définie sur un niveau de priorité, seuls les travaux et les flots de travaux dont la priorité dépasse la priorité minimale de travail peuvent à s'exécuter sur ce poste de travail. Par exemple, si la priorité minimale est de 40, les travaux dont la priorité est inférieure ou égale à 40 ne pourront pas être lancés.

Pour plus d'informations, voir «fence», à la page 418 et «priority», à la page 275.

Définition de limites

La ***limite*** offre un moyen de définir le nombre maximal de travaux que Tivoli Workload Scheduler est autorisé à lancer. Vous pouvez fixer une limite :

- dans la définition du flot de travaux avec l'argument ***job limit***,
- Dans la définition d'un poste de travail à l'aide de la commande ***limit cpu***.

La définition d'une limite de 25 sur un poste de travail, par exemple, autorise Tivoli Workload Scheduler à limiter à 25 le nombre de travaux s'exécutant simultanément sur ce poste de travail.

Pour plus d'informations, voir «limit cpu», à la page 422 et «limit sched», à la page 423.

Définition de ressources

Vous pouvez définir des ***ressources*** afin de représenter des actifs physiques ou logiques sur votre système. Chaque ressource est représentée par un nom et un nombre d'unités disponibles. Si vous disposez de trois unités de bande par exemple, vous pouvez définir une ressource `bandes` avec trois unités disponibles. Un travail utilisant deux unités de la ressource `bandes` empêche alors le lancement d'autres travaux demandant plus d'une unité. Toutefois, étant donné qu'une ressource n'est pas liée à un actif de façon stricte, vous pouvez utiliser une ressource fictive en tant que dépendance pour contrôler le traitement des travaux.

Pour plus d'informations, voir «Définition de ressource», à la page 225.

Demande de confirmation d'un travail

Dans certains scénarios, l'état d'achèvement d'un travail ne peut pas être déterminé tant que vous n'avez pas effectué certaines tâches. Par exemple, vous pouvez décider de vérifier les résultats imprimés dans un rapport. Dans ce cas, vous pouvez déterminer dans la définition du travail que ce dernier nécessite une *confirmation* pour que Tivoli Workload Scheduler attende votre réponse avant de marquer le travail comme ayant abouti ou échoué.

Pour plus d'informations, voir «confirm», à la page 409.

Définition des actions de reprise d'un travail

Lorsque vous planifiez un travail, vous pouvez spécifier le type de reprise que Tivoli Workload Scheduler doit effectuer en cas d'échec du travail. Les options de reprise prédéfinies sont :

- Le système passe au travail suivant.

- Arrêter et ne pas lancer le travail suivant.
- Exécuter à nouveau le travail ayant échoué.

Par ailleurs, vous pouvez préciser d'autres actions à entreprendre en termes de travaux et d'invites de reprise. Par exemple, si un travail échoue, vous pouvez indiquer à Tivoli Workload Scheduler d'exécuter automatiquement un travail de reprise, d'émettre une invite de reprise nécessitant une réponse affirmative, puis d'exécuter à nouveau le travail ayant échoué.

Pour plus d'informations, voir «Travail», à la page 772.

Gestion des activités de planification de production avec Tivoli Workload Scheduler

Chaque fois qu'un nouveau plan de production est généré, Tivoli Workload Scheduler sélectionne les flots de travaux qui s'exécutent pendant la période spécifiée pour le plan et reporte les flots de travaux non terminés à partir du plan de production précédent. Toutes les informations requises sont écrites dans un fichier nommé ***Symphony*** qui est continuellement mis à jour au cours du traitement pour indiquer le travail terminé, le travail en cours et le travail restant à effectuer. Le programme de ligne de commande **conman** (Console Manager) de Tivoli Workload Scheduler permet de gérer les informations du fichier `Symphony`. Le programme de ligne de commande **conman** peut être utiliser pour :

- démarrer et arrêter les processus de contrôle de Tivoli Workload Scheduler,
- afficher le statut des travaux et des flots de travaux,
- modifier les priorités et les dépendances,
- modifier la priorité minimale de travail et le nombre maximal de travaux,
- réexécuter des travaux,
- annuler des travaux et des flots de travaux,
- soumettre de nouveaux travaux et flots de travaux,
- répondre à des invites,
- connecter et déconnecter des postes de travail dans le réseau Tivoli Workload Scheduler,
- modifier le nombre de ressources disponibles.

A partir de la version 9.1, toutes les informations du plan écrites dans le fichier `Symphony` sont ensuite répliquées dans la base de données. Plusieurs opérations de surveillance demandées par Dynamic Workload Console accèdent à la base de données, plutôt qu'au fichier Symphony, ce qui entraîne des temps de réponse et des performances globales plus rapides. Les opérations suivantes demandées par Dynamic Workload Console accèdent aux informations à partir de la base de données :

- Surveillance des travaux et des flots de travaux
- Actualisation des vues de surveillance des travaux et flots de travaux
- Surveillance des postes de travail
- Surveillance des ressources, fichiers et invites
- Exécution des rapports de version de référence
- Affichage du plan dans la vue graphique
- Affichage d'une vue Incidence

Automatisation de la charge de travail grâce aux règles d'événement

Avant de procéder à la planification de travaux, vous pouvez automatiser la charge de travail sur la base de la demande à l'aide des règles d'événement. L'objet des règles d'événement est de réaliser un ensemble d'actions prédéfinies en réponse à des événements spécifiques affectant les objets Tivoli Workload Scheduler et non liés à Tivoli Workload Scheduler.

Du point de vue des objets Tivoli Workload Scheduler, le produit est doté d'un module d'extension capable de détecter les événements suivants :

- Pour un travail ou flot de travaux spécifique :
  - Modification de l'état
  - Dépassement de l'heure du dernier démarrage
  - Soumission
  - Annulation
  - Redémarrage
  - Retard
- Pour un poste de travail donné :
  - Modification de l'état
  - Modification de l'état du lien à partir du poste de travail parent
  - Modification de l'état du lien à partir du poste de travail enfant
- Une invite spécifique est affichée ou reçoit une réponse
- Le serveur d'applications sur un poste de travail donné est démarré ou arrêté

Lorsque l'un de ces événements se produit, l'une des actions suivantes peut être déclenchée :

- Soumission d'un flot de travaux, d'un travail ou d'une tâche
- Réponse à une invite
- Exécution de commandes non liées à Tivoli Workload Scheduler
- Connexion à un message opérateur
- Notification des utilisateurs par courriel
- Envoi de messages vers Tivoli Enterprise Console

Vous pouvez également définir et exécuter des règles d'événement chargées de détecter, selon les cas, un ou plusieurs de ces événements, ou bien une séquence ou un ensemble de ces événements dont l'exécution ne s'achève pas dans le délai imparti.

Pour plus d'informations, voir Chapitre 7, «Exécution de l'automatisation de la charge de travail gérée par événements», à la page 127.

Interfaces utilisateur Tivoli Workload Scheduler

Pour utiliser Tivoli Workload Scheduler, vous disposez d'une combinaison de programmes d'interface de ligne de commande, d'interface graphique et d'API. En particulier, l'interface de ligne de commande est proposée par certaines fonctions avancées indisponibles dans l'interface graphique. Les programmes d'interface utilisateur Tivoli Workload Scheduler disponibles sont les suivants :

Dynamic Workload Console
: Une interface utilisateur Web disponible pour visualiser et contrôler les activités de planification dans la production sur les environnements Tivoli Workload Scheduler répartis et z/OS. Avec le Dynamic Workload Console

vous pouvez utiliser tout navigateur supporté pour accéder à l'environnement Tivoli Workload Scheduler à partir de tout emplacement de votre réseau.

Vous pouvez utiliser la console Dynamic Workload pour :

- Définir des objets de planification dans la base de données Tivoli Workload Scheduler
- Visualiser et gérer les objets de planification impliqués dans les activités de planification en cours
- Créer et contrôler les connexions aux environnements Tivoli Workload Scheduler
- Soumettre des travaux et des flots de travaux en production
- Définir les préférences utilisateurs
- Créer et gérer des règles d'événement
- Définir et gérer des travaux critiques

Dynamic Workload Console doit être installé sur un serveur pouvant atteindre les noeuds Tivoli Workload Scheduler en utilisant les connexions réseaux. Consultez le manuel Tivoli Workload Scheduler - *Guide de planification et d'installation* pour plus d'informations.

composer
Ce programme de ligne de commande permet de définir et de gérer des objets de planification dans la base de données. Pour plus d'informations sur ce programme d'interface et son utilisation, voir Chapitre 8, «Définition d'objets dans la base de données», à la page 147 et Chapitre 9, «Gestion des objets dans la base de données - composer», à la page 301.

conman
Ce programme de ligne de commande permet de surveiller et de contrôler le traitement du plan de production Tivoli Workload Scheduler. Pour plus d'informations sur ce programme d'interface, voir Chapitre 11, «Gestion des objets dans le plan - conman», à la page 373.

API et plug-ins Java™
Ensemble de classes et de méthodes disponibles s'exécutant dans un environnement JAVA que vous utilisez pour créer votre interface personnalisée afin de gérer les objets de planification dans la base de données et dans le plan. Cette API ne peut pas être utilisée pour créer votre interface personnalisée afin de définir des options globales. De plus, vous pouvez utiliser et modifier un ensemble de plug-in qui réalisent des tâches spécifiques, ou créer vos propres plug-in. Cette API est disponible via un kit de développement de logiciels qui fait partie du produit. Pour plus d'informations et pour apprendre comment accéder à la documentation de l'API et des plug-in, voir *IBM Tivoli Workload Scheduler Developer's Guide: Software Development Kit (Integration Workbench)*.

optman
Programme de ligne de commande permettant de gérer les paramètres qui affectent la globalité de l'environnement Tivoli Workload Scheduler. Ces paramètres, également appelés options globales, sont stockés dans la base de données. Pour plus d'informations sur ce programme d'interface, voir *Tivoli Workload Scheduler - Guide d'administration*.

planman
Ce programme de ligne de commande permet de gérer la fonctionnalité de planification de Tivoli Workload Scheduler. Pour plus d'informations sur ce programme d'interface, voir «Ligne de commande Planman», à la page 87.

Interface des services Web

Cette interface offre un mécanisme d'accès basé sur des services Web à un sous-ensemble de fonctionnalités servant à gérer les travaux et les flots de travaux dans le plan. Elle ne vous permet pas de gérer le plan, de définir des options globales ni de gérer des objets dans la base de données. Pour plus d'informations, voir *IBM Tivoli Workload Scheduler - Guide du développeur : Services Web*.

Vous devez installer la fonction Tivoli Workload Scheduler Command Line Client sur les agents et systèmes tolérants en panne à l'extérieur du réseau Tivoli Workload Scheduler pour utiliser les programmes de ligne de commande**composer** et **et optman** et pour exécuter les commandes **planman showinfo** et **planman unlock**.

Pour plus d'informations sur la définition des options permettant d'autoriser un utilisateur à accéder aux interfaces de ligne de commande, voir «Configuration des options pour l'utilisation des interfaces utilisateur», à la page 59.

Démarrage de la production

La présente section contient les procédures de base qui vous permettront d'implémenter rapidement Tivoli Workload Scheduler dans votre environnement à l'aide de l'interface de ligne de commande. Ces informations supposent que :

- Ces procédures sont effectuées sur le gestionnaire de domaine maître immédiatement après l'installation du produit sur les systèmes où vous souhaitez effectuer vos activités de planification.
- L'ID utilisateur utilisé pour effectuer les opérations est le même que celui qui a servi lors de l'installation du produit.

Si vous ne connaissez pas bien Tivoli Workload Scheduler, vous pouvez suivre les procédures de définition d'un nombre limité d'objets de planification et en ajouter d'autres à mesure que vous vous familiarisez avec le produit. Vous pouvez commencer, par exemple, par deux ou trois de vos applications les plus fréquemment utilisées et ne définir des objets de planification que pour répondre à leurs besoins.

Sinon, vous pouvez utiliser Dynamic Workload Console pour procéder à la modélisation et réaliser les tâches de fonctionnement. Pour plus d'informations, reportez-vous à la documentation du produit correspondant.

La première activité à effectuer consiste à accéder à la base de données Tivoli Workload Scheduler et à définir l'environnement dans lequel vous souhaitez effectuer vos activités de planification à l'aide des types d'objet de planification Tivoli Workload Scheduler. Pour ce faire, procédez comme suit :

1. **Configuration des variables d'environnement Tivoli Workload Scheduler**

   Exécutez l'un des scripts suivants :

   . ./*racine_TWS*/**tws_env.sh** pour les shells Bourne et Korn sous UNIX

   . ./*racine_TWS*/**tws_env.csh** pour les shells C sous UNIX

   *racine_TWS***tws_env.cmd** sous Windows

   dans un interpréteur de commandes pour définir les variables *PATH* et *TWS_TISDIR*.
2. **Connexion à la base de données Tivoli Workload Scheduler**

Vous pouvez utiliser la syntaxe suivante pour vous connecter au gestionnaire de domaine maître avec l'ID *utilisateur_TWS* :

```
composer -user <utilisateur_TWS> -password <mot_de_passe_utilisateur_TWS>
```

où *utilisateur_TWS* est l'ID utilisateur que vous avez spécifié au moment de l'installation.

Remarque : Si vous souhaitez exécuter cette tâche et les suivantes à partir d'un système autre que le gestionnaire de domaine maître, vous devez indiquer les paramètres de connexion lors du lancement de **composer** (voir «Configuration des options pour l'utilisation des interfaces utilisateur», à la page 59).

3. **Ajout éventuel à la base de données des définitions permettant de décrire la topologie de votre environnement de planification en termes de :**
   - **Domaines**

     Exécutez cette tâche si vous souhaitez créer une arborescence hiérarchique du chemin dans votre environnement. L'utilisation de plusieurs domaines diminue le trafic réseau en réduisant les communications entre le gestionnaire de domaine maître et les autres postes de travail. Pour plus d'informations, voir «Définition de domaine», à la page 167.
   - **Poste de travail**

     Définissez un poste de travail sur chaque machine appartenant à votre environnement de planification, à l'exception du gestionnaire de domaine maître, qui est automatiquement défini pendant l'installation du produit Tivoli Workload Scheduler. Pour plus d'informations, voir «Définition de poste de travail», à la page 149. Le gestionnaire de domaine maître est défini automatiquement dans la base de données au moment de l'installation.
4. **Définition éventuelle des utilisateurs autorisés à exécuter des travaux sur des postes de travail Windows**

   Définissez les utilisateurs autorisés à exécuter des travaux à l'aide de Tivoli Workload Scheduler en spécifiant un nom d'utilisateur et un mot de passe. Pour plus d'informations, voir «Définition d'utilisateur», à la page 211.
5. **Définition éventuelle d'agendas**

   Les agendas vous permettent de déterminer si un flot de travaux ou un travail doit être exécuté et à quel moment. Vous pouvez les utiliser pour inclure ou exclure des jours et des heures du traitement. Les agendas ne sont pas requis pour définir les jours de planification pour les flots de travaux (le cycle d'exécution `simple` ou `rule` peut être utilisé) ; l'objectif principal est de définir des ensembles de dates *global* pouvant être réutilisés dans plusieurs flots de travaux. Pour plus d'informations, voir «Définition d'agenda», à la page 216.
6. **Définition éventuelle de paramètres, d'invites et de ressources**

   Pour plus d'informations, voir «Définition des variables et des paramètres», à la page 217, «Définition d'invite», à la page 224 et «Définition de ressource», à la page 225.
7. **Définition de travaux et de flots de travaux**

   Pour plus d'informations, voir «Travail», à la page 772 et «Définition de flot de travaux», à la page 236.
8. **Définition éventuelle de restrictions et de paramètres pour contrôler l'heure d'exécution des travaux et des flots de travaux**

Vous pouvez définir des dépendances pour les travaux et les flots de travaux. Le programme accepte jusqu'à 40 dépendances pour un flot de travaux. Il peut s'agir de :

- Dépendances de ressource
- Dépendances de fichier
- Dépendances de prédécesseur/successeur de travail et de flot de travaux
- Dépendances d'invite

Vous pouvez définir des paramètres temporels pour que les travaux et les flots de travaux s'exécutent en termes de :

- Cycles d'exécution
- Contraintes temporelles

Vous pouvez adapter l'exécution simultanée des travaux sur un poste de travail ou dans un flot de travaux en définissant les deux paramètres suivants :

- Limite
- Priorité

9. **Extension automatique du plan à la fin du terme de production courant**

   Ajoutez le flot de travaux `final` à la base de données pour étendre automatiquement le plan de production à l'issue de chaque terme de production courant en exécutant la commande suivante :

   ```
   add Sfinal
   ```

   Pour plus d'informations, voir «Automatisation du traitement du plan de production», à la page 106.

10. **Génération du plan**

    Exécutez la commande **JnextPlan** pour générer le plan de production. La commande **JnextPlan** permet de lancer le traitement des informations de planification stockées dans la base de données et de créer le plan de production pour la période spécifiée. La période par défaut est 24 heures. Si vous avez automatisé la génération du plan (voir étape précédente), vous devez seulement exécuter la commande **JnextPlan** la première fois.

Après cette procédure, votre environnement de planification est prêt à fonctionner. Il traite par lots une séquence ordonnée de travaux et de flots de travaux par rapport à des ressources définies sur un groupe de postes de travail, le cas échéant. Par défaut, le nombre de travaux pouvant être exécutés simultanément sur un poste de travail est nul la première fois que vous exécutez la commande **JnextPlan**. Par conséquent, vous devez augmenter cette valeur en modifiant la valeur du mot clé `limit cpu` de façon à autoriser l'exécution des travaux sur ce poste de travail (voir «limit cpu», à la page 422).

Si vous souhaitez apporter une modification quelconque alors que le plan de production est déjà en cours de traitement, utilisez le programme **conman**. Pendant le traitement du plan de production sur le réseau, vous pouvez continuer à définir ou à modifier des travaux et des flots de travaux dans la base de données. Cependant, ces modifications ne seront utilisées que si vous soumettez les travaux (commande **sbj**) ou les flots de travaux (commande **sbs**) sur un poste de travail qui a déjà reçu le plan, ou après la génération d'un nouveau plan de production via **JnextPlan**. Pour plus d'informations sur le programme **conman** et sur les opérations réalisables sur le plan de production en cours, voir Chapitre 11, «Gestion des objets dans le plan - conman», à la page 373.

Chapitre 2. Description des processus et des commandes de base

Dans un réseau multiniveau Tivoli Workload Scheduler, un groupe de processus de planification spécialisés assure en local sur chaque poste de travail la gestion des travaux et renvoie les informations sur le traitement des travaux dans l'arborescence hiérarchique jusqu'au gestionnaire de domaine maître. A l'aide des informations reçues des postes de travail, le gestionnaire de domaine maître met alors à jour sa copie du fichier `Symphony` et le plan répliqué dans la base de données qui contiennent les enregistrements décrivant les activités de traitement des travaux à effectuer sur le réseau Tivoli Workload Scheduler pendant le plan de production courant. Il renvoie ensuite aux postes de travail concernés les mises à jour portant sur les activités à effectuer.

Exécution de commandes sous Windows

Sous Windows, assurez-vous d'exécuter les commandes Tivoli Workload Scheduler à partir d'une invite de commande dotée du niveau de privilège **Exécuter en tant qu'administrateur**.

Processus de poste de travail Tivoli Workload Scheduler

La gestion des communications entre les postes de travail, le traitement local des travaux et la notification des mises à jour d'état sont réalisés sur chaque poste de travail Tivoli Workload Scheduler par une série de processus de gestion qui sont actifs pendant que le moteur s'exécute. Sur les agent tolérant aux pannes et les gestionnaires de domaine, ces processus s'appuient sur l'infrastructure WebSphere Application Server. Cette infrastructure est automatiquement installée avec le poste de travail et permet à Tivoli Workload Scheduler de :

- Communiquer sur le réseau Tivoli Workload Scheduler.
- Gérer les mécanismes d'authentification pour les clients distants, comme les programmes de ligne de commande, qui se connectent au gestionnaire de domaine maître à l'aide des protocoles HTTP ou HTTPS.

Pour plus d'informations sur le démarrage et l'arrêt de l'infrastructure WebSphere Application Server et des processus Tivoli Workload Scheduler sur un poste de travail, voir «Démarrage et arrêt de processus sur un poste de travail», à la page 39. En dehors du démarrage et de l'arrêt de WebSphere Application Server et de la gestion des paramètres de connexion lors de la communication sur le réseau Tivoli Workload Scheduler, l'infrastructure de WebSphere Application Server est transparente lors de l'utilisation de Tivoli Workload Scheduler.

Dans ce guide, les *processus Tivoli Workload Scheduler* ou les *processus de poste de travail* permettent d'identifier les processus suivants :

netman
monman
writer
mailman
batchman
jobman

A l'exception des agents standard, ces processus sont lancés sur les postes de travail Tivoli Workload Scheduler selon l'ordre suivant :

netman
Netman est le processus de gestion du réseau. Il est lancé par la commande **Startup** et se comporte comme un programme d'écoute du réseau, dont il reçoit des requêtes start, stop, link ou unlink. **Netman** examine chaque demande entrante et génère un processus Tivoli Workload Scheduler local.

monman
Le processus **Monman** est démarré par **netman** et est utilisé dans le cadre de la gestion d'événements. Il permet de démarrer la surveillance et les services de **ssmagent** ayant pour rôle de détecter les services qui sont définis dans les règles déployées et activées sur le poste de travail concerné. Lorsque ces services interceptent les événements, ils les envoient, après une action de filtrage préliminaire, vers le serveur de traitement d'événements généralement exécuté sur le gestionnaire de domaine maître. Si aucune configuration de règle d'événement n'est téléchargée sur le poste de travail, les services de surveillance demeurent à l'état de veille.

Le processus de communication entre les agents de surveillance et le serveur de traitement d'événements est indépendant de la topologie du réseau Tivoli Workload Scheduler. Il repose directement sur le numéro de port EIF du processeur d'événements et les informations sur les événements transitent directement à partir des agents de surveillance, sans nécessiter de passage par des gestionnaires de domaine intermédiaires. A certain de degré de tolérance aux pannes est garanti par des mémoires locales qui stockent temporairement les occurrences d'événements sur les agents en cas de rupture des communications avec le processeur d'événements.

writer **Writer** est un processus lancé par **netman** pour transférer les messages entrants au processus **mailman** local. Les processus **writer** (il peut y en avoir plusieurs sur un poste de travail gestionnaire de domaine) sont démarrés par les demande de connexion (voir «link», à la page 424) et sont arrêtés par les demandes de connexion (voir «unlink», à la page 510) ou lorsque le processus **mailman** en cours de communication se termine.

mailman
Mailman est le processus de gestion du courrier (Mail Management). Il achemine les messages vers les postes de travail locaux ou distants. Sur un gestionnaire de domaine, des processus **mailman** supplémentaires peuvent être créés pour diviser la charge de travail de **mailman** due à l'initialisation des agents et pour améliorer le caractère opportun des messages. Lorsque le gestionnaire de domaine démarre, il crée une instance distincte du processus **mailman** pour chaque *ID serveur* spécifié dans les définitions de poste de travail de l'agent tolérant aux pannes et des agents standard qu'il gère. Chaque poste de travail est contacté avec son propre *ID serveur* sur le gestionnaire de domaine. Pour plus d'informations, voir «Définition de poste de travail», à la page 149.

batchman
Batchman est le processus de contrôle de production. Il interagit directement avec la copie du fichier `Symphony` distribuée aux postes de travail au début de la période de production et la met à jour. **Batchman** assure plusieurs fonctions :

- Gestion locale du traitement et de la mise à jour du plan.

- Résolution des dépendances des travaux et des flots de travaux.
- Sélection des travaux à exécuter.
- Mise à jour du plan avec les résultats du traitement des travaux.

Batchman est le seul processus habilité à mettre à jour le fichier Symphony.

jobman

Jobman est le processus de gestion des travaux. Il lance les travaux sous le contrôle de **batchman** et transmet le statut des travaux au processus **mailman**. Il est chargé de suivre l'état des travaux et de définir l'environnement conformément aux scripts jobmanrc et .jobmanrc lors des demandes de lancement de travaux. Pour plus d'informations sur ces scripts, voir Chapitre 3, «Configuration de l'environnement de travail», à la page 47. Lorsque le processus **jobman** reçoit un message de lancement de travail de **batchman**, il génère un processus de surveillance des travaux. Le nombre maximal de processus de surveillance des travaux pouvant être générés sur un poste de travail est déterminé à l'aide de la commande **limit cpu** à partir de l'invite de ligne de commande **conman** (voir «limit cpu», à la page 422).

moniteur de travaux (jobman sous UNIX, JOBMON.exe et joblnch.exe sous Windows)

Le processus de surveillance des travaux exécute d'abord un ensemble d'actions qui définissent l'environnement avant que le travail ne soit lancé, puis il lance le travail en exécutant la commande ou le fichier script spécifié dans la définition du travail. Pour savoir comment spécifier la commande ou le fichier script lancé avec le travail, voir «Travail», à la page 772.

Les activités de configuration se composent du lancement du fichier de configuration standard (*TWS_home*/jobmanrc sous UNIX et *TWS_home*/jobmanrc.cmd sous Windows) qui contient les paramètres applicables à tous les travaux s'exécutant sur le poste de travail. De plus, sur les postes de travail UNIX un script de configuration local *utilisateur_TWS*/.jobmanrc est lancé, s'il existe dans le répertoire d'accueil de l'utilisateur lançant de travail. Ce fichier de configuration local contient des paramètres qui s'appliquent uniquement aux travaux lancés par cet utilisateur spécifique. En cas d'échec de l'une de ces étapes, le travail se termine à l'état FAIL.

Avertissement : Si, sous Windows, une variable système nommé TEMP existe, l'utilisateur *utilisateur_TWS* doit être autorisé à créer des fichiers dans le répertoire pour lequel la variable est définie. Si cette exigence n'est pas remplie, le fichier binaire **JOBMON.exe** ne réussit pas à démarrer correctement.

Tous les processus, à l'exception de **jobman**, s'exécutent en tant qu'*utilisateur_TWS*. **Jobman** s'exécute en tant que root.

Sur les postes de travail d'agent standard, le processus **batchman** n'est pas lancé car ce type de poste de travail ne gère pas la planification des travaux. Ces postes de travail lancent uniquement des travaux sous la direction de leur gestionnaire de domaine. Localement, sur le poste de travail, les processus de gestion attendent de recevoir une demande de lancement d'un travail émise par le gestionnaire de domaine en mode écoute. Une fois la demande reçue, le travail est lancé localement et le résultat est renvoyé au gestionnaire de domaine. Pour plus

d'informations relatives aux postes de travail de l'agent standard, voir *IBM Tivoli Workload Scheduler - Guide de planification et d'installation*.

La figure 3, à la page 37 affiche l'arborescence du processus sur les postes de travail Tivoli Workload Scheduler autres que les agents standard et installés sous UNIX :

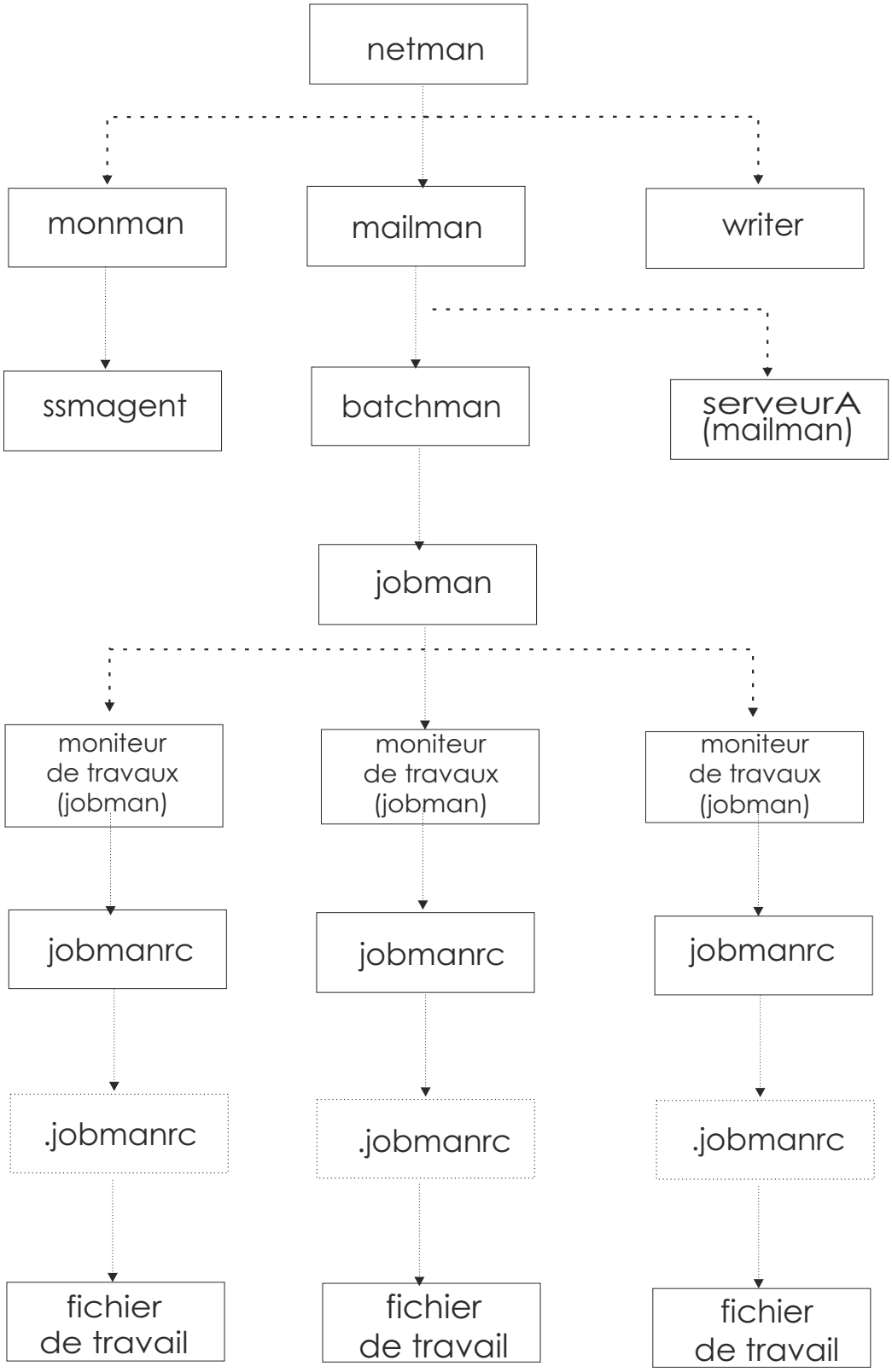


Figure 3. Arborescence des processus sous UNIX

La figure 4, à la page 38 affiche l'arborescence du processus sur les postes de travail Tivoli Workload Scheduler autres que les agents standard et installés sous

Windows :

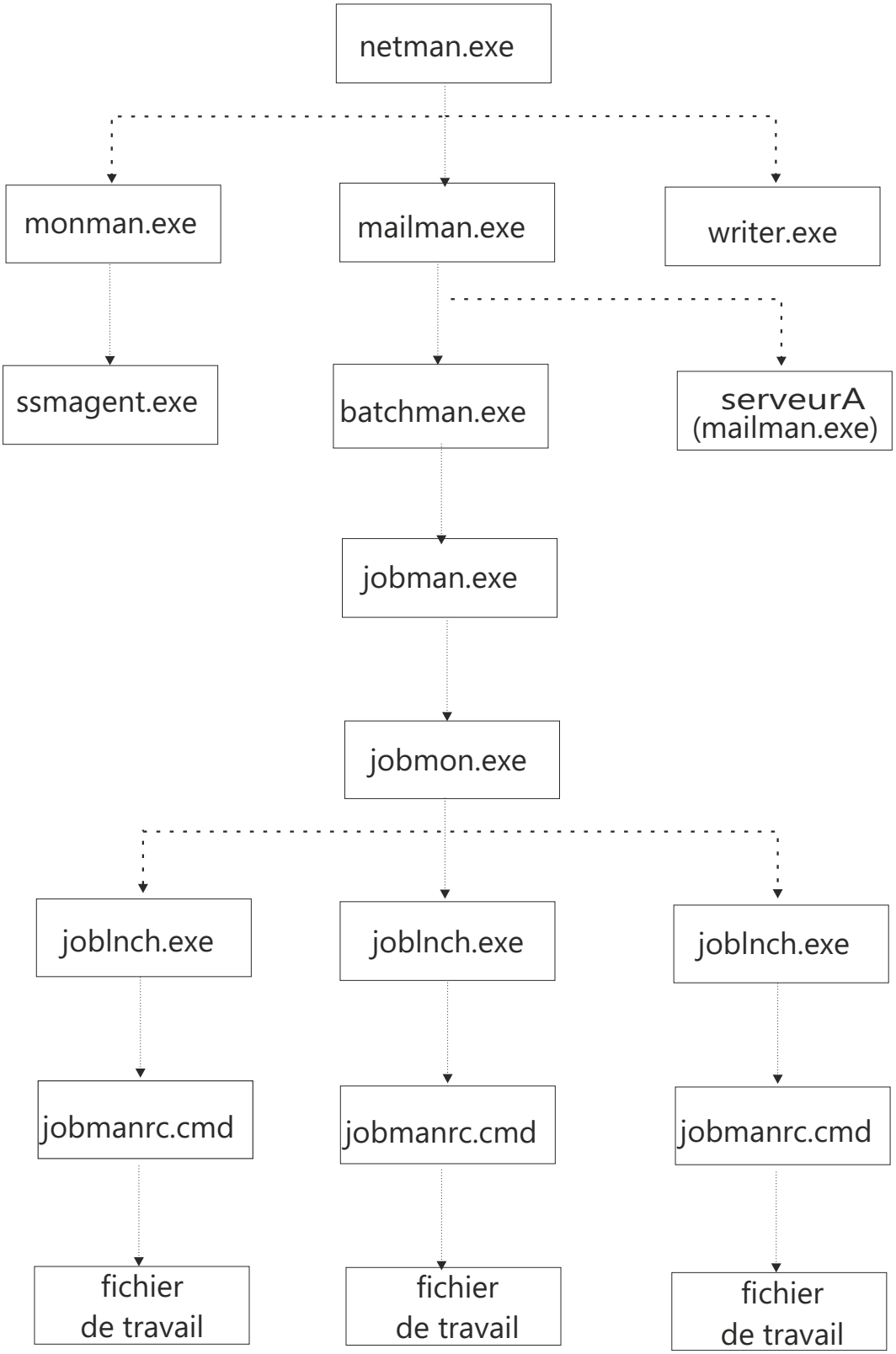


Figure 4. Arborescence des processus sous Windows

Sur les plateformes Windows, il existe un service supplémentaire, `Tivoli Token Service`, qui permet aux processus Tivoli Workload Scheduler d'être lancés comme s'ils étaient émis par l'utilisateur Tivoli Workload Scheduler.

Démarrage et arrêt de processus sur un poste de travail

Le type de système d'exploitation installé sur le poste de travail détermine la manière dont les processus Tivoli Workload Scheduler peuvent être lancés depuis la ligne de commande. Le tableau 6 explique comment démarrer et arrêter l'infrastructure WebSphere Application Server et les processus Tivoli Workload Scheduler sur un poste de travail en fonction du système d'exploitation.

Tableau 6. Démarrage et arrêt de Tivoli Workload Scheduler sur un poste de travail

Action	**Commandes utilisées sur la plateforme UNIX**	**Commandes utilisées sur la plateforme Windows**
Démarre tous les processus Tivoli Workload Scheduler dont WebSphere Application Server et le moteur de surveillance d'événements.	conman start conman startappserver conman startmon	conman start conman startappserver conman startmon
Démarre netman et WebSphere Application Server. Sous Windows démarre également `Tivoli Token Service`	./StartUp.sh	StartUp
Arrête tous les processus Tivoli Workload Scheduler et WebSphere Application Server.	conman shutdown ./stopWas.sh	conman shutdown -appsrv shutdown -appsrv
Arrête tous les processus Tivoli Workload Scheduler, à l'exception de WebSphere Application Server.	conman shutdown	conman shutdown shutdown
Démarre tous les processus Tivoli Workload Scheduler, à l'exception de WebSphere Application Server et du moteur de surveillance d'événements.	conman start	conman start
Arrête tous les processus Tivoli Workload Scheduler, à l'exception de **netman**, **monman**, **writer** et **appservman**.	conman stop	conman stop
Arrête tous les processus Tivoli Workload Scheduler (y compris **netman**).	conman shutdown	conman shutdown shutdown
Démarre WebSphere Application Server	./startWas.sh ou conman startappserver	startWas ou conman startappserver
Arrête WebSphere Application Server	./stopWas.sh ou conman stopappserver	stopWas ou conman stopappserver

Tableau 6. Démarrage et arrêt de Tivoli Workload Scheduler sur un poste de travail (suite)

Action	Commandes utilisées sur la plateforme UNIX	Commandes utilisées sur la plateforme Windows
Démarrer le moteur de surveillance d'événements.	conman startmon	conman startmon
Arrêter le moteur de surveillance d'événements	conman stopmon	conman stopmon
Démarrer l'agent localement	./StartUpLwa.sh **Remarque :** peut être exécuté par l'*utilisateur_TWS* ou par le superutilisateur uniquement.	startuplwa **Remarque :** Sous Windows 2008, doit être exécuté par un administrateur.
Arrêter l'agent localement	./ShutDownLwa.sh **Remarque :** peut être exécuté par l'*utilisateur_TWS* ou par le superutilisateur uniquement.	shutdownlwa **Remarque :** Sous Windows 2008, doit être exécuté par un administrateur.

Remarque : Sous Windows, les systèmes s'abstiennent d'utiliser les services Windows pour arrêter WebSphere Application Server. Utilisez à la place l'une des commandes figurant dans ce tableau. Si vous utilisez les services Windows pour arrêter WebSphere Application Server, le processus *appserverman*, dont l'exécution se poursuit, le redémarre.

Pour plus d'informations sur la commande d'utilitaire **StartUp**, voir «StartUp», à la page 576.

Pour plus d'informations sur la commande d'utilitaire **shutdown**, voir «shutdown», à la page 575.

Voir *IBM Tivoli Workload Scheduler - Guide d'administration* pour plus d'informations sur les commandes **startWas** et **stopWas**.

Pour plus d'informations sur la commande **conman start**, voir «start», à la page 479.

Pour plus d'informations sur la commande **conman stop**, voir «stop», à la page 485.

Pour plus d'informations sur la commande **conman shutdown**, voir «shutdown», à la page 478.

Voir «startappserver», à la page 481 pour plus d'informations sur la commande **conman startappserver**.

Voir «stopappserver», à la page 488 pour plus d'informations sur la commande **conman stopappserver**.

Voir «startmon», à la page 483 pour plus d'informations sur la commande **conman startmon**.

Voir «stopmon», à la page 491 pour plus d'informations sur la commande **conman stopmon**.

Si l'agent est installé sur un système Windows, WebSphere Application Server et les processus **netman** sont automatiquement lancés au moment du démarrage en tant que services avec `Tivoli Token Service`. Si l'agent est installé sur un système UNIX, WebSphere Application Server et les processus **netman** peuvent être automatiquement lancés au démarrage en ajoutant une instruction appelant **Startup** dans le fichier `/etc/inittab`.

Démarrage et arrêt de la agent

Le type de système d'exploitation installé sur le poste de travail détermine comment les agents peuvent être démarrés à partir de la ligne de commande.

Tableau 7. Démarrage et arrêt de la agent

Action	Commandes utilisées sur la plateforme UNIX	Commandes utilisées sur la plateforme Windows
Démarrer l'agent localement	./StartUpLwa.sh **Remarque :** peut être exécuté par l'*utilisateur_TWS* ou par le superutilisateur uniquement.	startuplwa **Remarque :** Sous Windows 2008, doit être exécuté par un administrateur.
Arrêter l'agent localement	./ShutDownLwa.sh **Remarque :** peut être exécuté par l'*utilisateur_TWS* ou par le superutilisateur uniquement.	shutdownlwa **Remarque :** Sous Windows 2008, doit être exécuté par un administrateur.

Pour plus d'informations relatives à l'arrêt et au démarrage d'un agent, voir ShutDownLwa et StartUpLwa.

Communication interprocessus du poste de travail

Tivoli Workload Scheduler utilise les files d'attente de messages pour les communications interprocessus locales. Il existe quatre fichiers de messages ; ils résident dans le répertoire *racine_TWS* :

NetReq.msg
Ce fichier de messages est lu par le processus **netman** pour les commandes locales. Il reçoit les messages de type START, STOP, LINK et UNLINK (demandes de démarrage, d'arrêt, de connexion et de déconnexion).

Mailbox.msg
Ce fichier de messages est lu par le processus **mailman**. Au moyen de l'interface utilisateur graphique (Dynamic Workload Console) ou du gestionnaire de console (**conman**), il reçoit les messages entrants des processus **batchman** et **jobman** locaux ainsi que d'autres postes de travail Tivoli Workload Scheduler du réseau.

Intercom.msg
Ce fichier de messages est lu par le processus **batchman** et il contient les instructions envoyées par le processus **mailman** local.

Courier.msg
Ce fichier de messages est écrit par le processus **batchman** et lu par le processus **jobman**.

PlanBox.msg
Ce fichier de messages est écrit par le processus **batchman** et est lu par le moteur.

Server.msg
Ce fichier de messages est écrit par le processus **batchman** et est lu par le moteur.

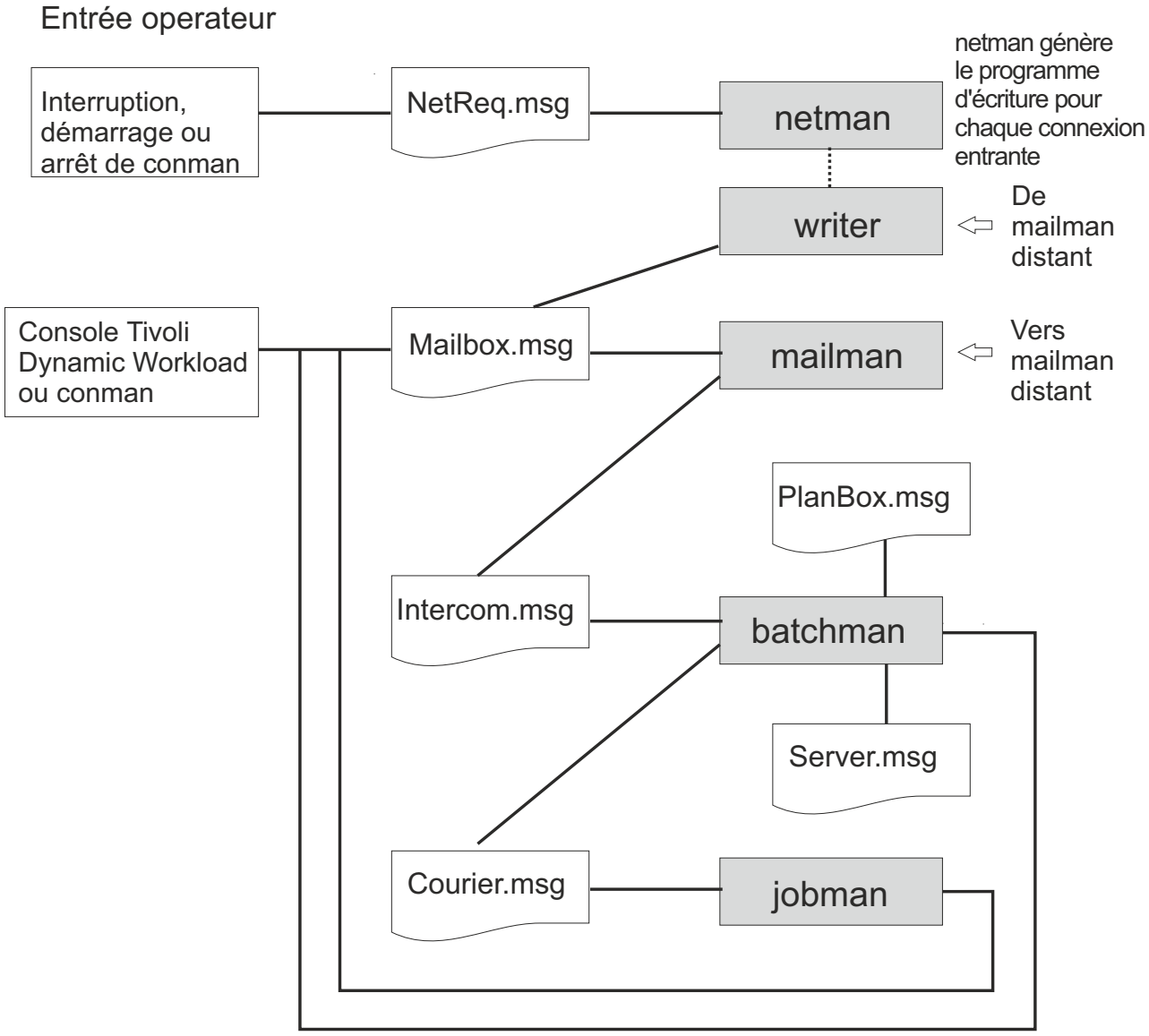


Figure 5. Communication interprocessus

Ces fichiers ont une taille maximale de 10 Mo par défaut. Cette taille maximale peut être modifiée à l'aide de l'utilitaire **evtsize** (voir «evtsize», à la page 557).

Communication réseau de Tivoli Workload Scheduler

Tivoli Workload Scheduler utilise le protocole TCP/IP pour les communications réseau. Le nom de noeud et le numéro de port utilisés pour établir la connexion TCP/IP sont définis pour chaque poste de travail dans la définition qui leur est associée. Pour plus de détails, voir «Définition de poste de travail», à la page 149.

Tivoli Workload Scheduler recourt à une technologie de *stockage et réacheminement* pour maintenir la cohérence et la tolérance aux pannes à tout moment sur l'ensemble du réseau en mettant les messages en attente dans des fichiers de messages lorsque la connexion n'est pas active. Lorsque la communication TCP/IP est établie entre les systèmes, Tivoli Workload Scheduler fournit des communications bidirectionnelles entre les postes de travail à l'aide de liaisons. Ces liaisons sont contrôlées par l'indicateur **autolink** défini (voir «Définition de poste de travail», à la page 149) et par des commandes émises par le programme de ligne de commande **conman** (voir «link», à la page 424 et «unlink», à la page 510).

Lorsqu'une liaison est ouverte, les messages sont transmis entre les postes de travail. Lorsqu'une liaison est fermée, le poste de travail expéditeur stocke les messages dans un fichier local de messages et les envoie au poste de travail de destination dès que la liaison est de nouveau ouverte.

Il existe essentiellement deux types de communication dans l'environnement Tivoli Workload Scheduler : l'initialisation des connections et la livraison des événements de planification sous la forme des messages de changement d'état pendant la période de traitement. Ces deux types de communication sont décrits ici de façon détaillée.

Initialisation de la connexion et configuration de la communication bidirectionnelle

Voici les étapes permettant l'établissement d'une liaison Tivoli Workload Scheduler bidirectionnelle entre un gestionnaire de domaine et un agent tolérant aux pannes distant :

1. Sur le gestionnaire de domaine, le processus **mailman** lit le nom d'hôte, l'adresse TCP/IP et le numéro de port de l'agent tolérant aux pannes à partir du fichier `Symphony`.
2. Le processus **mailman** s'exécutant sur le gestionnaire de domaine établit une connexion TCP/IP avec le processus **netman** de l'agent tolérant aux pannes à l'aide des informations obtenues à partir du fichier `Symphony`.
3. Le processus **netman** s'exécutant sur l'agent tolérant aux pannes détermine que la demande provient du processus **mailman** qui s'exécute sur le gestionnaire de domaine et il génère un nouveau processus **writer** pour traiter la connexion entrante.
4. Le processus **mailman** s'exécutant sur le gestionnaire de domaine est désormais connecté au processus **writer** de l'agent tolérant aux pannes. Le processus **writer** de l'agent tolérant aux pannes communique le numéro d'exécution courant de sa copie du fichier `Symphony` au processus **mailman** du gestionnaire de domaine. Ce numéro d'exécution est l'identificateur utilisé par Tivoli Workload Scheduler pour reconnaître chaque fichier `Symphony` généré par **JnextPlan**. Cette étape est indispensable pour permettre au gestionnaire de domaine de vérifier si le plan courant a déjà été envoyé à l'agent tolérant aux pannes.

5. Le processus **mailman** du gestionnaire de domaine compare le numéro d'exécution de son fichier Symphony au numéro d'exécution du fichier Symphony de l'agent tolérant aux pannes. Si ces numéros diffèrent, le processus **mailman** du gestionnaire de domaine envoie au processus **writer** de l'agent tolérant aux pannes la dernière copie du fichier Symphony.
6. Une fois que le fichier Symphony courant est en place sur l'agent tolérant aux pannes, le processus **mailman** du gestionnaire de domaine envoie une commande start à l'agent tolérant aux pannes.
7. Le processus **netman** de l'agent tolérant aux pannes démarre le processus **mailman** local. A ce stade, une liaison de communication unidirectionnelle est établie depuis le gestionnaire de domaine vers l'agent tolérant aux pannes.
8. Le processus **mailman** de l'agent tolérant aux pannes lit le nom d'hôte, l'adresse TCP/IP et le numéro de port du gestionnaire de domaine à partir du fichier Symphony et il utilise ces informations pour établir une liaison en retour vers le processus **netman** du gestionnaire de domaine.
9. Le processus **netman** du gestionnaire de domaine détermine que la demande provient du processus **mailman** de l'agent tolérant aux pannes et il génère un nouveau processus **writer** pour traiter la connexion entrante. Le processus **mailman** de l'agent tolérant aux pannes est désormais connecté au processus **writer** du gestionnaire de domaine et une liaison de communication bidirectionnelle est établie. En conséquence, le processus **writer** du gestionnaire de domaine enregistre les messages reçus depuis l'agent tolérant aux pannes dans le fichier Mailbox.msg sur le gestionnaire de domaine et le processus **writer** de l'agent tolérant aux pannes enregistre les messages reçus depuis le gestionnaire de domaine dans le fichier Mailbox.msg sur l'agent tolérant aux pannes.

Le traitement des travaux et la transmission des événements de planification sous la forme de messages de changement d'état au cours du jour de traitement effectués localement par l'agent tolérant aux pannes

Au cours de la période de production, le fichier Symphony présent sur l'agent tolérant aux pannes est lu et mis à jour avec les informations de changement d'état relatives aux travaux exécutés localement par les processus de poste de travail Tivoli Workload Scheduler. Voici les opérations effectuées localement sur l'agent tolérant aux pannes pour lire et mettre à jour le fichier Symphony et pour traiter les travaux :

1. Le processus **batchman** lit un enregistrement dans le fichier Symphony qui indique que travail1 doit être lancé sur le poste de travail.
2. Le processus **batchman** écrit dans le fichier Courier.msg que travail1 doit être lancé.
3. Le processus **jobman** lit ces informations dans le fichier Courier.msg, lance travail1 et écrit dans le fichier Mailbox.msg que travail1 a démarré avec son *id_processus* et son *horodatage*.
4. Le processus **mailman** lit ces informations dans son fichier Mailbox.msg et envoie un message indiquant que travail1 a démarré avec son *ID_processus* et son *horodatage*, sur le fichier Mailbox.msg du gestionnaire de domaine et sur le fichier Intercom.msg local de l'agent tolérant aux pannes.
5. Le processus **batchman** de l'agent tolérant aux pannes lit le message dans le fichier Intercom.msg et met à jour la copie locale du fichier Symphony.

6. Une fois que le traitement du `travail1` est terminé, le processus **jobman** met à jour le fichier `Mailbox.msg` avec les informations indiquant que `travail1` est terminé.
7. Le processus **mailman** lit ces informations dans le fichier `Mailbox.msg` et envoie un message indiquant que `travail1` est terminé à la fois au fichier `Mailbox.msg` du gestionnaire de domaine et au fichier `Intercom.msg` local de l'agent tolérant aux pannes.
8. Le processus **batchman** de l'agent tolérant aux pannes lit le message dans le fichier `Intercom.msg`, met à jour la copie locale du fichier `Symphony` et détermine le travail suivant à exécuter.

Pour plus d'informations sur l'ajustement du traitement du travail sur un poste de travail, voir *IBM Tivoli Workload Scheduler - Guide d'administration*.

Prise en charge d'IPv6 (Internet Protocol version 6)

Tivoli Workload Scheduler prend en charge le protocole IP version 6 (IPv6) en complément de la version IPv4. Pour aider les clients à passer d'un environnement IPv4 à un environnement IPv6 complet, Tivoli Workload Scheduler prend en charge l'utilisation des protocoles IP en double pile. Cela signifie que le produit est conçu pour l'utilisation simultanée des adresses IPv4 et IPv6 lors de la communication avec les autres applications utilisant le protocole IPv4 ou IPv6.

Pour cela, les fonctionnalités `gethostbyname` et `gethostbyaddr` ont été supprimées de Tivoli Workload Scheduler car elle ne prenaient en charge que le protocole IPv4. Elles sont remplacées par la nouvelle interface API `getaddrinfo`, qui rend le mécanisme de communications client/serveur entièrement indépendant du protocole.

La fonction `getaddrinfo` traite à la fois les conversions de type nom/adresse et service/port et renvoie des structures `sockaddr` à la place d'une liste d'adresses. Ces structures `sockaddr` peuvent ensuite être utilisées directement par les fonctions de socket. Ceci permet à `getaddrinfo` de masquer toutes les dépendances vis-à-vis des protocoles dans les fonctions des bibliothèques auxquelles elles appartiennent. L'application gère uniquement les structures d'adresse de socket renseignées par `getaddrinfo`.

Chapitre 3. Configuration de l'environnement de travail

Le présent chapitre décrit la personnalisation de la gestion des travaux sur un poste de travail. Cette personnalisation consiste à attribuer sur chaque poste de travail des valeurs à des variables qui ont une incidence sur le traitement de **jobman**. Ce chapitre contient les sections suivantes :

Présentation de l'environnement de travail

Sur chaque poste de travail, les travaux sont lancés par le processus de contrôle de la production **batchman**. Le processus **batchman**résout toutes les dépendances des travaux pour que les travaux soient traités dans le bon ordre, puis met en file d'attente un message de lancement de travail destiné au processus **jobman**.

Chaque processus lancé par **jobman** (scripts de configuration et travaux inclus) conserve le nom d'utilisateur enregistré lors de l'ouverture de session du travail. Les travaux soumis (travaux, fichiers ou commandes soumis(es) *manuellement* par un utilisateur et non par un plan programmé) conservent le nom de l'utilisateur qui a effectué la soumission.

Le processus **jobman** démarre un processus de surveillance des travaux qui commence par définir un groupe de variables d'environnement, puis exécute un script de configuration standard appelé *racine_TWS*/**jobmanrc**, qui peut être personnalisé. Le script **jobmanrc** définit des variables utilisées pour configurer localement sur le poste de travail la façon dont les travaux sont lancés, quel que soit l'utilisateur.

Sur les postes de travail UNIX, si l'utilisateur est autorisé à utiliser un script de configuration local et que le script *RACINE_UTILISATEUR*/**.jobmanrc** existe, le script de configuration local **.jobmanrc** est *également* exécuté. Le travail est ensuite exécuté par le script de configuration standard ou par le script de configuration local. Les résultats du traitement du travail sont signalés à **jobman** qui, à son tour, met à jour le fichier `Mailbox.msg` avec les informations sur l'état d'achèvement du travail. Pour exécuter des travaux sur l'environnement de l'utilisateur, ajoutez l'instruction suivante au script de configuration local :

```
. $USER_home/.profile
```

Remarque : Avant d'ajouter `.profile` au fichier `.jobmanrc`, assurez-vous qu'il ne contient pas de paramètre *stty* ni d'étapes qui exigent une intervention manuelle de l'utilisateur. Le cas échéant, ajoutez au fichier `.jobmanrc` uniquement les étapes nécessaires de `.profile`.

Sur les postes de travail Windows, le script de configuration local djobmanrc.cmd est exécuté s'il existe dans le répertoire Documents and Settings de l'utilisateur qui est représenté par la variable d'environnement %USERPROFILE% et dépend de l'installation du langage Windows. Le script djobmanrc.cmd sera exécuté par le script jobmanrc.cmd.

Variables d'environnement exportées par jobman

Les variables répertoriées ci-dessous (voir tableau 8) sont définies localement sur le poste de travail et exportées par **jobman** sur les systèmes d'exploitation Windows.

Tableau 8. Variables d'environnement de travail sous Windows

Nom de la variable	Valeur
COMPUTERNAME	Valeur de *COMPUTERNAME* définie dans l'environnement utilisateur.
HOME	Chemin d'installation de l'instance Tivoli Workload Scheduler.
HOMEDRIVE	Valeur de *HOMEDRIVE* définie dans l'environnement utilisateur.
HOMEPATH	Valeur de *HOMEPATH* définie dans l'environnement utilisateur.
LANG	Valeur de *LANG* définie dans l'environnement utilisateur. Si elle n'est pas spécifiée, sa valeur est définie sur C.
LOGNAME	Nom d'utilisateur de la connexion.
MAESTRO_OUTPUT_STYLE	Paramètre définissant le style de sortie pour les noms d'objet longs.
SystemDrive	Valeur de *SYSTEMDRIVE* définie dans l'environnement utilisateur.
SystemRoot	Valeur de *SYSTEMROOT* définie dans l'environnement utilisateur.
TEMP	Valeur de *TEMP* définie dans l'environnement utilisateur. Si elle n'est pas spécifiée, sa valeur est définie sur `c:\temp`.
TIVOLI_JOB_DATE	Date planifiée d'un travail.
TMPTEMP	Valeur de *TMP* définie dans l'environnement utilisateur. Si elle n'est pas spécifiée, sa valeur est définie sur `c:\temp`.
TMPDIR	Valeur de *TMPDIR* définie dans l'environnement utilisateur. Si elle n'est pas spécifiée, sa valeur est définie sur `c:\temp`.
TWS_PROMOTED_JOB	S'applique aux fonctions Assurance de service de charge de travail. Peut prendre la valeur `YES` ou `No`. Si la valeur est `YES`, le travail (un travail critique ou l'un de ses prédécesseurs) a été promu.
TZ	Le fuseau horaire, s'il a été défini dans l'environnement du système d'exploitation.
UNISON_CPU	Nom de ce poste de travail.
UNISON_DIR	Valeur de *UNISON_DIR* définie dans l'environnement utilisateur.
UNISON_EXEC_PATH	Chemin complet de `jobmanrc`.

Tableau 8. Variables d'environnement de travail sous Windows (suite)

Nom de la variable	Valeur
UNISONHOME	Chemin d'installation de l'instance Tivoli Workload Scheduler.
UNISON_HOST	Nom du poste de travail hôte.
UNISON_JOB	L'identificateur de travail absolu : `poste_de_travail#id_plan.travail`.
UNISON_JOBNUM	Numéro du travail.
UNISON_MASTER	Nom du poste de travail maître.
UNISON_RUN	Numéro d'exécution de production en cours de Tivoli Workload Scheduler.
UNISON_SCHED	Nom du flot de travaux.
UNISON_SCHED_DATE	Date de production de Tivoli Workload Scheduler (aammjj) signalée dans l'en-tête du fichier `Symphony`.
UNISON_SCHED_ID	*ID_flot_travaux* du flot de travaux qui contient le travail en cours.
UNISON_SCHED_IA	Heure de début (*StartTime*) du flot de travaux qui contient le travail en cours.
UNISON_SCHED_EPOCH	Date de production Tivoli Workload Scheduler (exprimée au format Epoch).
UNISON_SHELL	Shell de connexion de l'utilisateur qui exécute le travail.
UNISON_STDLIST	Nom du chemin d'accès du fichier de liste standard du travail.
UNISON_SYM	Numéro d'enregistrement `Symphony` du travail lancé.
USERDOMAIN	Valeur de *USERDOMAIN* définie dans l'environnement utilisateur.
USERNAME	Valeur de *USERNAME* définie dans l'environnement utilisateur.
USERPROFILE	Valeur de *USERPROFILE* définie dans l'environnement utilisateur.

Les variables répertoriées ci-dessous (voir tableau 9) sont définies localement sur le poste de travail et exportées par **jobman** sur les systèmes d'exploitation UNIX.

Tableau 9. Variables d'environnement de travail pour UNIX

Nom de la variable	Valeur
HOME	Répertoire principal de l'utilisateur.
LANG	Valeur de *LANG* définie dans l'environnement utilisateur.
LD_LIBRARY_PATH	Valeur de *LD_LIBRARY_PATH* définie dans l'environnement utilisateur.
LD_RUN_PATH	Valeur de *LD_RUN_PATH* définie dans l'environnement utilisateur.
LOGNAME	Nom d'utilisateur de la connexion.
MAESTRO_OUTPUT_STYLE	Paramètre définissant le style de sortie pour les noms d'objet longs.
PATH	`/bin:/usr/bin`

Tableau 9. Variables d'environnement de travail pour UNIX (suite)

Nom de la variable	Valeur
TIVOLI_JOB_DATE	Date planifiée d'un travail.
TWS_PROMOTED_JOB	S'applique aux fonctions Assurance de service de charge de travail. Peut prendre la valeur `YES` ou `No`. Si la valeur est `YES`, le travail (un travail critique ou l'un de ses prédécesseurs) a été promu.
TWS_TISDIR	Valeur de *TWS_TISDIR* définie dans l'environnement utilisateur.
TZ	Fuseau horaire défini.
UNISON_CPU	Nom de ce poste de travail.
UNISON_DIR	Valeur de *UNISON_DIR* définie dans l'environnement utilisateur.
UNISON_EXEC_PATH	Chemin complet de `.jobmanrc`.
UNISONHOME	Chemin d'installation de l'instance Tivoli Workload Scheduler.
UNISON_HOST	Nom du poste de travail hôte.
UNISON_JOB	L'identificateur de travail absolu : `poste_de_travail#id_plan.travail`.
UNISON_JOBNUM	Numéro du travail.
UNISON_MASTER	Nom du poste de travail maître.
UNISON_RUN	Numéro d'exécution de production en cours de Tivoli Workload Scheduler.
UNISON_SCHED	Nom du flot de travaux.
UNISON_SCHED_DATE	Date de production de Tivoli Workload Scheduler (aammjj) signalée dans l'en-tête du fichier `Symphony`.
UNISON_SCHED_ID	*ID_flot_travaux* du flot de travaux qui contient le travail en cours.
UNISON_SCHED_IA	Heure de début (*StartTime*) du flot de travaux qui contient le travail en cours.
UNISON_SCHED_EPOCH	Date de production Tivoli Workload Scheduler exprimée au format Epoch.
UNISON_SHELL	Shell de connexion de l'utilisateur qui exécute le travail.
UNISON_STDLIST	Nom du chemin d'accès du fichier de liste standard du travail.
UNISON_SYM	Numéro d'enregistrement `Symphony` du travail lancé.

Personnalisation du format des dates dans stdlist

Vous pouvez utiliser une variable d'environnement nommée ***UNISON_DATE_FORMAT*** pour définir le format de date utilisé pour la date de l'en-tête et du bas de page du fichier `stdlist`. Cette variable peut être définie sur les postes de travail UNIX et Windows, impérativement avant le lancement des processus Tivoli Workload Scheduler sur ce poste de travail pour être effective. Pour définir cette variable, procédez comme suit :

Sous UNIX

1. Ajoutez l'instruction pour exporter la variable *UNISON_DATE_FORMAT* dans le fichier `.profile` racine.
2. Exécutez le fichier `.profile`.
3. Exécutez **conman shutdown**, puis **./StartUp.sh**.

Sous Windows

1. Dans les Propriétés système, définissez la variable système *UNISON_DATE_FORMAT*.
2. Exécutez **conman shutdown**, puis **StartUp.sh**.

Voici quelques exemples de paramètres utilisés pour afficher le format de l'année dans la zone de date de l'en-tête et du bas de page du fichier `stdlist`. Le paramètre :

```
UNISON_DATE_FORMAT = "%a %x %X %Z %Y"
```

produit une sortie au format suivant :

```
Ven 15/10/04 11:05:24 AM GMT 2004
```

Le paramètre :

```
UNISON_DATE_FORMAT = "%a %x %X %Z"
```

produit une sortie au format suivant :

```
Ven 15/10/04 11:05:24 AM GMT
```

Définissez cette variable localement sur chaque poste de travail pour lequel vous souhaitez afficher le format de l'année à 4 chiffres. Si ce paramètre est omis, le format standard à deux caractères est utilisé.

Personnalisation du traitement des travaux sur un poste de travail UNIX - jobmanrc

Tivoli Workload Scheduler propose un modèle de script de configuration standard nommé *racine_TWS*/config/**jobmanrc**. Il est installé automatiquement sous le nom *racine_TWS*/jobmanrc. L'administrateur système peut utiliser ce script pour définir l'environnement nécessaire avant l'exécution de chaque travail. Pour modifier le script, apportez vos modifications dans la copie de travail (*racine_TWS*/jobmanrc), sans manipuler le fichier modèle. Le fichier contient des variables que vous pouvez configurer et quelques commentaires qui vous aident à assimiler la méthodologie. Pour découvrir les variables jobmanrc, voir tableau 10.

Tableau 10. Variables définies par défaut dans le fichier jobmanrc

Nom de la variable	Valeur
UNISON_JCL	Le nom du chemin d'accès au fichier script du travail
UNISON_STDLIST	Le nom du chemin d'accès au fichier liste standard du travail

Tableau 10. Variables définies par défaut dans le fichier jobmanrc (suite)

Nom de la variable	Valeur
UNISON_EXIT	**yes** | **no** Si cette variable a pour valeur **yes**, le travail s'arrête immédiatement lorsqu'une commande renvoie un code de sortie non nul. Si elle est définie sur **no**, le programme continue l'exécution du travail même si une commande renvoie un code de sortie non nul. Les autres valeurs sont interprétées comme étant définies sur **no**.
LOCAL_RC_OK	**yes** | **no** Si la valeur **yes** est attribuée à cette variable, le script de configuration local de l'utilisateur s'exécute (s'il existe) et passe *$UNISON_JCL* comme premier argument. Dans certains cas, l'utilisateur peut ne pas être autorisé à utiliser cette option. Pour plus d'informations, voir «Personnalisation du traitement des travaux pour un utilisateur sous UNIX - .jobmanrc», à la page 54. Si la valeur **no** est attribuée à cette variable, le programme ignore l'existence d'un script de configuration local et exécute *$UNISON_JCL*. Les autres valeurs sont interprétées comme étant définies sur **no**.
MAIL_ON_ABEND	**yes** | **no** Pour les systèmes d'exploitation UNIX : Si la variable est définie sur **yes**, le programme envoie un message à la boîte aux lettres de l'utilisateur connecté dès que le travail se termine avec un code de sortie non nul. Vous pouvez également définir la variable sur un ou plusieurs noms d'utilisateur (préalablement séparés par des espaces) de façon à envoyer un message à chacun d'entre eux. Par exemple, vous pouvez taper "root mis sam mary". Si la variable est définie sur **no**, le programme n'envoie aucun message en cas de fin anormale du travail. Les messages d'arrêt anormal utilisent le format suivant : *cpu#sched.job* *fichier-jcl* `failed with` *code-sortie* `Please review` *nom_fichier_liste-standard* Vous pouvez modifier ou traduire le texte du message dans une autre langue. Pour plus d'informations à ce sujet, voir «Personnalisation de la section MAIL_ON_ABEND de jobmanrc», à la page 53.

Tableau 10. Variables définies par défaut dans le fichier jobmanrc (suite)

Nom de la variable	Valeur
SHELL_TYPE	**standard | user | script** Si cette variable a pour valeur **standard**, la première ligne du fichier JCL est lue pour déterminer quel shell utiliser pour exécuter le travail. Si la première ligne ne commence pas par le symbole **#!**, le programme utilise le répertoire `/bin/sh` pour exécuter le script de configuration local ou *$UNISON_JCL*. Les commandes sont répercutées sur le fichier de liste standard du travail. Si la variable est définie sur user, le shell de connexion de l'utilisateur (*$UNISON_SHELL*) exécute le script de configuration local ou *$UNISON_JCL*. Les commandes sont répercutées sur le fichier de liste standard du travail. Si la valeur **script** est attribuée à la variable, le programme exécute directement le script de configuration local ou *$UNISON_JCL*, et les commandes ne sont pas répercutées tant que le script de configuration local ou *$UNISON_JCL* contient une commande **set -x**. Les autres valeurs sont interprétées comme étant définies sur **standard**.
USE_EXEC	**yes | no** Si cette variable a pour valeur **yes**, le travail ou le script de configuration local de l'utilisateur est exécuté à l'aide de la commande **exec**, ce qui élimine un processus supplémentaire. Cette option est remplacée si *MAIL_ON_ABEND* est également défini sur **yes**. Les autres valeurs sont interprétées comme étant définies sur **no**, auquel cas le travail ou le script de configuration local est exécuté par un autre processus du shell.

Personnalisation de la section MAIL_ON_ABEND de jobmanrc

Vous pouvez modifier le texte utilisé dans le message envoyé aux utilisateurs spécifiés dans la zone ***MAIL_ON_ABEND*** du fichier de configuration *racine_TWS*`/jobmanrc` en accédant à ce fichier et en modifiant le texte dans les parties qui apparaissent en caractères gras :

```
# Mail a message to user or to root if the job fails.

if [ "$MAIL_ON_ABEND" = "YES" ]
then
  if [ $UNISON_RETURN -ne 0 ]
  then
    mail $LOGNAME <<-!
        $UNISON_JOB
        \'$UNISON_JCL\' failed with $UNISON_RETURN
        Please review $UNISON_STDLIST
!
  fi
elif [ "$MAIL_ON_ABEND" = "ROOT" ]
then
```

```
  if [ $UNISON_RETURN -ne 0 ]
  then
    mail root <<-!
        $UNISON_JOB
        \'$UNISON_JCL\' failed with $UNISON_RETURN
        Please review $UNISON_STDLIST
!
  fi
elif [ "$MAIL_ON_ABEND" != "NO" ]
then
  if [ $UNISON_RETURN -ne 0 ]
  then
    mail $MAIL_ON_ABEND <<-!
        $UNISON_JOB
        \'$UNISON_JCL\' failed with $UNISON_RETURN
        Please review $UNISON_STDLIST
!
  fi
fi
```

Personnalisation du traitement des travaux pour un utilisateur sous UNIX - .jobmanrc

Sur les postes de travail UNIX, le script de configuration local **.jobmanrc** permet aux utilisateurs de définir l'environnement nécessaire lors du traitement de leurs propres travaux. Contrairement au script **jobmanrc**, le script **.jobmanrc** peut être personnalisé pour effectuer différents actions pour différents utilisateurs. Chaque utilisateur défini comme *utilisateur_tws* peut personnaliser le script **.jobmanrc** de son répertoire personnel pour effectuer des actions de prétraitement et de posttraitement. Le script **.jobmanrc** est une étape supplémentaire qui s'exécute avant le lancement du travail proprement dit.

Le script **.jobmanrc** s'exécute uniquement dans les conditions suivantes :

- Le script de configuration standard `jobmanrc` est installé et la variable d'environnement ***LOCAL_RC_OK*** définie sur **yes** (voir tableau 10, à la page 51).
- Si le fichier *racine_TWS*`/localrc.allow` existe, le nom d'utilisateur doit figurer dans le fichier. Si le fichier *racine_TWS*`/localrc.allow` n'existe pas, le nom de l'utilisateur ne doit pas apparaître dans le fichier *racine_TWS*`/localrc.deny`. Si aucun des fichiers n'existe, l'utilisateur est autorisé à utiliser un script de configuration local.
- Le script de configuration local est installé dans le répertoire personnel de l'utilisateur (*racine_UTILISATEUR*`/.jobmanrc`) et dispose de droits en exécution.

Les travaux ne sont pas automatiquement exécutés ; la commande ou le script doit être lancé à partir de **.jobmanrc**. La commande ou le script est lancé différemment selon le type de processus que vous voulez exécuter. Suivez les règles générales ci-dessous lorsque vous lancez des scripts à partir de **.jobmanrc** :

- Utilisez **eval** si vous voulez lancer une commande.
- Utilisez soit **eval**, soit **exec** si vous voulez lancer un script qui n'a pas besoin d'activités de posttraitement.
- Utilisez **eval** si vous voulez lancer un script qui a besoin d'activités posttraitement.

Si vous avez l'intention d'utiliser un script de configuration local, il doit au minimum exécuter le fichier script du travail (*$UNISON_JCL*). Le script de configuration standard fourni par Tivoli Workload Scheduler, `jobmanrc`, exécute votre script de configuration local comme suit :

```
$EXECIT $USE_SHELL $USER_home/.jobmanrc "$UNISON_JCL" $IS_COMMAND
```

où :

- La valeur de *USE_SHELL* est celle de la variable ***SHELL_TYPE*** de `jobmanrc` (voir tableau 10, à la page 51).
- La valeur **yes** est attribuée à *IS_COMMAND* si le travail a été planifié ou soumis dans la production à l'aide de **submit docommand**.
- *EXECIT* a pour valeur **exec** si la variable ***USE_EXEC*** a pour valeur **yes** (voir tableau 10, à la page 51), sinon elle est indéfinie (null).

Toutes les variables exportées dans **jobmanrc** sont disponibles dans le shell **.jobmanrc** ; cependant, les variables définies mais non exportées ne sont pas disponibles.

Dans l'exemple suivant, nous expliquons comment exécuter le fichier script d'un travail ou une commande dans votre script de configuration local :

```
#!/bin/ksh
PATH=racine_TWS:racine_TWS/bin:$PATH
export PATH
/bin/sh -c "$UNISON_JCL"
```

Voici un exemple de script `.jobmanrc` qui effectue les traitements en fonction du code de sortie du travail de l'utilisateur :

```
#!/bin/sh
#
PATH=racine_TWS:racine_TWS/bin:$PATH
export PATH
/bin/sh -c "$UNISON_JCL"
#Ou utilisez eval "$UNISON_JCL" (guillemets obligatoires)
RETVAL=$?
if [ $RETVAL -eq 1 ]
then
 echo "Code de sortie 1 - Erreur non bloquante"
 exit 0
elif [ $RETVAL -gt 1 -a $RETVAL -lt 100 ]
then
 conman "tellop : Erreur de base de données. Contacter l'administrateur de la base de données"
elif [ $RETVAL -ge 100 ]
then
 conman "Abandon du travail tellop. Contactez l'administrateur"
fi
```

Personnalisation du traitement des travaux sur un poste de travail Windows - jobmanrc.cmd

Tivoli Workload Scheduler propose un modèle de script de configuration standard nommé *racine_TWS*\config\jobmanrc.cmd. Il est installé automatiquement sous le nom *racine_TWS*\jobmanrc.cmd. Vous pouvez utiliser ce fichier de commandes pour définir l'environnement nécessaire avant l'exécution de chaque travail. Pour modifier le fichier, apportez vos modifications dans la copie de travail (*racine_TWS*\jobmanrc.cmd), sans manipuler le fichier modèle. Le fichier contient

des variables que vous pouvez configurer et quelques commentaires qui vous aident à assimiler la méthodologie. Pour découvrir les variables `jobmanrc.cmd`, voir tableau 11.

Tableau 11. Variables définies par défaut dans le fichier jobmanrc.cmd

Nom de la variable	Valeur
HOME	Chemin d'accès au répertoire *racine_TWS*
POSIXHOME	Chemin d'accès au répertoire *racine_TWS* au format POSIX
LOCAL_RC_OK	• Si cette variable a pour valeur **yes**, le script de configuration local de l'utilisateur est exécuté, s'il existe. • Si cette variable a pour valeur **no**, le programme ignore l'existence d'un script de configuration local. Les autres valeurs sont interprétées comme étant définies sur **no**.
MAIL_ON_ABEND	• Si cette variable est définie sur **YES**, un courrier électronique est envoyé à l'ID de messagerie électronique défini dans la variable *email_ID*, si le travail se termine par une erreur. • Si cette variable est définie par une autre valeur que **YES** ou **NO**, un courrier électronique est envoyé à l'ID de messagerie électronique spécifié dans cette variable, si le travail se termine par une erreur. • Si cette variable est définie par **NO**, aucun message n'est envoyé si le travail se termine par une erreur. Pour plus de détails, voir «Personnalisation de la section MAIL_ON_ABEND de jobmanrc.cmd».

Personnalisation de la section MAIL_ON_ABEND de jobmanrc.cmd

Vous pouvez modifier le texte utilisé dans le message envoyé aux utilisateurs spécifiés dans la zone ***MAIL_ON_ABEND*** du fichier de configuration *TWS_home*`/jobmanrc.cmd` en accédant à ce fichier et en modifiant le texte dans les parties qui apparaissent en caractères gras. Pour expliquer plus clairement comment générer un message électronique, un modèle de programme de messagerie portant le nom bmail.exe est utilisé.

```
if /I "%MAIL_ON_ABEND%"=="NO" (goto :out) else (goto :mail_on_abend)

:mail_on_abend
REM ******email, task or other action inserted here ******************
if /I "%MAIL_ON_ABEND%"=="YES" (goto :email) else (goto :email_spec)

:email
c:\"Program Files"\utils\bmail.exe -s smtp.yourcompany.com -t %EMAIL_ID%
-f %COMPUTERNAME%@yourcompany.com -h -a "Subject: Job %UNISON_JOB% abended"
-b "Job %UNISON_JOB% Job Number %UNISON_JOBNUM% abended"
goto :out
```

```
:email_spec
REM set > c:\tmp\abended_jobs\%UNISON_JOB%.j%UNISON_JOBNUM%
c:\"Program Files"\utils\bmail.exe -s smtp.yourcompany.com -t %MAIL_ON_ABEND%
-f %COMPUTERNAME%@yourcompany.com -h -a "Subject: Job %UNISON_JOB% abended"
-b "Job %UNISON_JOB% Job Number %UNISON_JOBNUM% abended"
```

Personnalisation du traitement des travaux sur un poste de travail Windows - djobmanrc.cmd

Sur les postes de travail Windows, vous pouvez utiliser le script de configuration local djobmanrc.cmd pour établir un environnement spécifique lors du traitement de vos travaux personnalisés. Contrairement au script jobmanrc.cmd, vous pouvez personnaliser le script djobmanrc.cmd afin d'effectuer différentes actions pour différents utilisateurs.

Les conditions suivantes s'appliquent :

- Le script doit contenir toutes les variables ou chemins d'application d'environnement nécessaires pour que Tivoli Workload Scheduler soit lancé correctement.
- Le script doit exister si un environnement spécifique à l'utilisateur pour l'exécution du travail est requis ou si un courrier électronique doit être envoyé à l'utilisateur de connexion du travail lorsque le travail Tivoli Workload Scheduler se termine par une erreur.

Pour créer un script djobmanrc.cmd personnalisé, procédez comme suit :

1. Connectez-vous en tant que l'utilisateur qui définit les variables d'environnement pour le lancement des travaux Tivoli Workload Scheduler.
2. Ouvrez une invite de commande DOS.
3. Entrez la commande **set** qui réachemine la sortie standard vers un fichier à plat nommé `user_env`.
4. Créez un fichier appelé djobmanrc.cmd dans le répertoire Documents and Settings de l'utilisateur, comportant le texte par défaut suivant au début :

   ```
   @ECHO OFF
       echo Invoking %USERNAME% DJOBMANRC.CMD V.1
       set USERPROFILE=%USERPROFILE%
       ::Setup User Environment Phase
   ```

5. Editez le fichier `user_env` créé à l'étape 3.
6. Insérez la commande **set** sur chaque ligne avant chaque variable d'environnement.
7. Ajoutez les modifications à la variable PATH à la fin du fichier djobmanrc.cmd dans une chaîne du type suivant :

   ```
   set PATH=<TWSHOME>;<TWSHOME>\bin;%PATH%
   ```

8. Ajoutez le texte suivant à la fin du fichier `user_env` et remplacez la chaîne *user email id* par l'ID de messagerie électronique de l'utilisateur qui reçoit la notification par courrier électronique, si le travail se termine par une erreur.

   ```
   set EMAIL_ID=<user email id>
       ::Launch Operation Phase
       %ARGS%
       ::Post Operations Phase
       :out
   ```

9. Ajoutez le fichier `user_env` mis à jour à la fin du fichier djobmanrc.cmd. Le fichier djobmanrc.cmd modifié doit ressembler à l'exemple suivant :

   ```
   @ECHO OFF
       echo Invoking %USERNAME% DJOBMANRC.CMD V.1
       set USERPROFILE=%USERPROFILE%
   ```

```
::Setup User Environment Phase
set ALLUSERSPROFILE=C:\Documents and Settings\All Users
set APPDATA=C:\Documents and Settings\petes\Application Data
set CommonProgramFiles=C:\Program Files\Common Files
set COMPUTERNAME=PSOTOJ
set ComSpec=C:\WINDOWS\system32\cmd.exe
set CURDRIVE=C
set FP_NO_HOST_CHECK=NOset
set HOMEDRIVE=c:
set HOMEPATH=\docs
set LOGONSERVER=\\PSOTOJ
set NEWVAR=c:\tmp\tmp\mlist1
set NUMBER_OF_PROCESSORS=1
set OPC_CLIENT_ROOT=C:\opc\Client
set OS=Windows_NT
set Path=C:\Program Files\utils;C:\PROGRAM
FILES\THINKPAD\UTILITIES;C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem;C:\Program
Files\IBM\Infoprint Select;C:\Utilities;C:\Notes;C:\Program Files\IBM\Trace Facility\;C:\Program
Files\IBM\Personal Communications\;C:\Program Files\XLView\;C:\lotus\compnent\;C:\WINDOWS\Downloaded
Program Files;C:\Program Files\Symantec\pcAnywhere\;"C:\Program Files\Symantec\Norton Ghost
2003\";C:\Infoprint;
set PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH
set PCOMM_Root=C:\Program Files\IBM\Personal Communications\
set PDBASE=C:\Program Files\IBM\Infoprint Select
set PDHOST=
set PD_SOCKET=6874
set PROCESSOR_ARCHITECTURE=x86
set PROCESSOR_IDENTIFIER=x86 Family 6 Model 9 Stepping 5, GenuineIntel
set PROCESSOR_LEVEL=6
set PROCESSOR_REVISION=0905
set ProgramFiles=C:\Program Files
set PROMPT=$P$G
set SESSIONNAME=Console
set SystemDrive=C:
set SystemRoot=C:\WINDOWS
set TEMP=C:\DOCUME⌂1\petes\LOCALS⌂1\Temp
set TMP=C:\DOCUME⌂1\petes\LOCALS⌂1\Temp
set tvdebugflags=0x260
set tvlogsessioncount=5000
set TWS4APPS_JDKHOME=c:\win32app\TWS\pete\methods\_tools
set USERDOMAIN=PSOTOJ
set USERNAME=petes
set USERPROFILE=C:\Documents and Settings\petes
set windir=C:\WINDOWSPATH=c:\win32app\tws\twsuser:c:\win32app\tws\twsuser\bin:%PATH%
set PATH=c:\win32app\TWS\twsuser;c:\win32app\TWS\twsuser\bin;%PATH%
set EMAIL_ID=johndoe@yourcompany.com
::Launch Operation Phase
%ARGS%
::Post Operations Phase
:out
```

La phase des opérations de lancement dans le script a lieu quand le script, le fichier binaire ou la commande définie pour le travail s'est terminé. Le texte “%ARGS%” est obligatoire.

La phase des opérations ultérieures dans le script a lieu quand un code d'exit de travail peut être à nouveau réglé d'un état ABEND à SUCC, en remplaçant un code d'exit différent de zéro par un code d'exit zéro. Certaines applications peuvent comporter des codes d'exit pouvant être des avertissements. Tivoli Workload Scheduler évalue les codes d'exit à zéro ou une valeur différente de zéro. Les codes d'exit zéro indiquent un travail à l'état “SUCC”. Tous les autres codes indiquent un travail à l'état ABEND. Les codes d'exit autres que zéro peuvent être réglés si nécessaire. L'exemple suivant présente ce qui peut être inclus dans la phase d'opérations ultérieures. L'exemple extrait le code d'exit du travail défini pour déterminer comment le gérer en fonction des instructions `If` :

```
set EMAIL_ID=johndoe@yourcompany.com
    ::Launch Operation Phase
    %ARGS%
    ::Post Operations Phase
    set RETVAL=%ERRORLEVEL%
    if "%RETVAL%"=="0" goto out
    if "%RETVAL%"=="1" set RETVAL=0
    if "%RETVAL%"=="6" set RETVAL=0
    :out
    exit %RETVAL%
```

Configuration des options pour l'utilisation des interfaces utilisateur

Pour utiliser Dynamic Workload Console, les paramètres de connexion sont fournis au sein de la console et sauvegardés lors de sa configuration.

Pour utiliser le client de ligne de commande Tivoli Workload Scheduler, vous devez fournir les informations de configuration suivantes (appelées ***paramètres_connexion***) afin de vous connecter au gestionnaire de domaine maître via HTTP/HTTPS à l'aide de l'infrastructure WebSphere Application Server :

hostname
Nom d'hôte du gestionnaire de domaine maître.

numéro de port
Numéro de port utilisé lors de la connexion au gestionnaire de domaine maître.

nom d'utilisateur, mot de passe
Les données d'identification, le nom d'utilisateur et le mot de passe, de l'*utilisateur_TWS*.

nom d'hôte proxy
Le nom d'hôte proxy utilisé dans la connexion avec le protocole HTTP.

numéro de port proxy
Le numéro de port proxy utilisé dans la connexion avec le protocole HTTP.

protocole
Protocole utilisé lors de la communication. Il peut s'agir de HTTP avec une authentification standard ou de HTTPS avec une authentification par certificat.

délai d'attente
Le délai d'attente indiquant le temps maximum pendant lequel le programme de l'interface utilisateur peut attendre la réponse du gestionnaire de domaine maître avant de considérer la demande de communication comme un échec.

poste de travail par défaut
Le nom de poste de travail du gestionnaire de domaine maître auquel vous souhaitez vous connecter.

paramètres SSL
Si vous avez configuré votre réseau pour utiliser SSL afin de communiquer entre les interfaces et le gestionnaire de domaine maître, vous devez également fournir l'ensemble de paramètres SSL appropriés (qui dépend de la configuration de votre couche SSL).

Au cas où le client de ligne de commande est installé sur le gestionnaire de domaine maître, cette configuration est effectuée automatiquement lors de l'installation.

Pour le client de ligne de commande installé sur d'autres postes de travail, ces informations peuvent être fournies soit par stockage dans les fichiers de propriétés de ces postes de travail ou en les fournissant dans la chaîne des commandes que vous utilisez.

Les fichiers de propriétés auxquels il est fait référence sont les fichiers **localopts** et **useropts** :

localopts
Ce fichier contient un ensemble de paramètres applicables au poste de travail local pour une instance spécifique du produit installé.

useropts
Ce fichier contient un sous-ensemble des paramètres de localopts qui ont des valeurs personnalisées pour un utilisateur spécifique. Le chemin de ce fichier est compris dans le répertoire de base de l'utilisateur, ce qui conserve la confidentialité de ces informations.

Etant donné que Tivoli Workload Scheduler prend en charge l'installation de plusieurs instances du produit sur la même machine, il peut y avoir plusieurs instances de fichier `useropts` de chaque utilisateur. La possibilité d'avoir plusieurs fichiers `useropts` ayant chacun un nom différent permet de spécifier différents ensembles de paramètres de connexion pour les utilisateurs définis sur plusieurs instances du produit installées sur la même machine.

Dans le fichier `localopts` de chaque instance du produit installé, l'option nommée ***useropts*** identifie le nom du fichier `useropts` auquel le système doit accéder pour se connecter à cette instance d'installation.

Par conséquent, si deux Tivoli Workload Scheduler sont installées sur une machine et qu'un utilisateur système appelé `operator` est défini en tant qu'utilisateur dans les deux instances, l'option locale *`useropts`* = `useropts1` du fichier `localopts` de la première instance identifie le fichier `useropts1` qui contient les paramètres de connexion nécessaires à l'utilisateur `operator` pour se connecter à cette instance de Tivoli Workload Scheduler. D'autre part, l'option locale *`useropts`* = `useropts2` du fichier `localopts` de la deuxième instance de Tivoli Workload Scheduler identifie le fichier `useropts2` qui contient les paramètres de connexion nécessaires à l'utilisateur `operator` pour se connecter à cette instance de Tivoli Workload Scheduler.

Des détails complets sur la configuration de cet accès sont fournis dans *Tivoli Workload Scheduler - Guide d'administration*, dans la rubrique intitulée "Configuration de l'authentification de l'accès au client de ligne de commande".

Chapitre 4. Gestion du cycle de production

Le coeur d'une solution de gestion des travaux et de planification est la création et la gestion du *plan de production*. Le plan de production est une liste de tâches qui contient toutes les actions à effectuer dans un intervalle de temps donné sur les postes de travail du réseau de planification à l'aide des ressources disponibles et en respectant les relations et les restrictions définies.

Le présent chapitre décrit comment Tivoli Workload Scheduler gère les plans.

Il comprend les sections suivantes :

Concepts de base de la gestion de plan

Le *plan de production* contient des informations sur les travaux à exécuter, sur quel agent tolérant aux pannes et quelles dépendances doivent être satisfaites avant le lancement de chaque travail. Tivoli Workload Scheduler crée le plan de production à partir des données de modélisation stockées dans la base de données et d'un plan intermédiaire appelé *plan de préproduction* . Ce dernier est automatiquement créé et géré par le produit. Pour éviter les problèmes, la base de données est verrouillée pendant la génération du plan et déverrouillée lorsque la génération est terminée ou lorsqu'une erreur se produit. Le plan de préproduction permet d'identifier à l'avance les instances de flot de travaux et les dépendances externes de prédécesseur/successeur des flots de travaux intervenant pendant la période indiquée.

Vous devez utiliser le script **JnextPlan** sur le gestionnaire de domaine maître pour générer le plan de production et le distribuer sur le réseau Tivoli Workload Scheduler. Pour plus d'informations sur le script **JnextPlan**, voir «Création et extension d'un plan de production», à la page 84.

Pour générer et démarrer un nouveau plan de production, Tivoli Workload Scheduler exécute les étapes suivantes :

1. Il met à jour le plan de préproduction avec les objets définis dans la base de données qui ont été ajoutés ou mis à jour depuis la dernière création ou extension du plan.
2. Il extrait du plan de préproduction les informations sur les flots de travaux à exécuter dans la période spécifiée et enregistre ces informations dans un plan de production intermédiaire.
3. Il inclut dans le nouveau plan de production les flots de travaux inachevés du plan de production précédent.

4. Il crée le nouveau plan de production et le stocke dans un fichier nommé Symphony. Les données du plan sont également répliquées dans la base de données.
5. Il distribue une copie du fichier Symphony aux postes de travail concernés par le traitement du nouveau plan de production.
6. Journalise toutes les statistiques du plan de production précédent dans une archive
7. Met à jour les états de flot de travaux dans le plan de préproduction

La copie du fichier Symphony qui vient d'être générée est déployée en commençant par les agent tolérant aux pannes du domaine le plus haut et les gestionnaires de domaine des domaines enfants, puis en descendant dans l'arborescence vers tous les domaines subordonnés.

Chaque agent tolérant aux pannes et gestionnaire de domaine qui reçoit le nouveau fichier Symphony, archive le fichier Symphony précédent sous Symphony.last au chemin *<rép_base_TWA>*/TWS/, de sorte qu'une copie de sauvegarde soit conservée. Ce mécanisme permet d'afficher les données Symphony précédentes dans le cas où des mises à jour de message sont appliquées aux états de travail et de flot de travaux perdus entre l'agent et son gestionnaire de domaine maître.

Chaque agent tolérant aux pannes qui reçoit le plan de production peut poursuivre le traitement même si la connexion réseau avec le gestionnaire de domaine associé s'interrompt.

Sur chaque agent tolérant aux pannes destinataire, les processus Tivoli Workload Scheduler exécutent les actions suivantes pour gérer le traitement des travaux :

1. Accès à la copie du fichier Symphony et lecture des instructions sur les travaux à exécuter.
2. Appel du système d'exploitation pour lancer les travaux nécessaires.
3. Mise à jour de leur copie du fichier Symphony avec les résultats de traitement de travail et envoi d'une notification au gestionnaire de domaine maître et à tous les agent tolérant aux pannes de statut Intégral. La copie originale du fichier Symphony stockée sur le gestionnaire de domaine maître et les copies stockées sur les gestionnaires de domaine maître de sauvegarde, si définies, sont mises en jour en conséquence.

Ceci signifie que pendant le traitement du travail, chaque agent tolérant aux pannes a sa propre copie du fichier Symphony mise à jour avec les informations sur les travaux qu'il exécute (ou qui sont exécutés dans son domaine et ses domaines enfants si le agent tolérant aux pannes est de statut Intégral ou est un gestionnaire de domaine). De plus le gestionnaire de domaine maître (et le gestionnaire de domaine maître de sauvegarde s'il est défini) a la copie du fichier Symphony qui contient toutes les mises à jour venant de tous les agent tolérant aux pannes. Le fichier Symphony reste ainsi à jour sur le gestionnaire de domaine maître avec les travaux qui doivent être exécutés, ceux qui sont en cours d'exécution et ceux qui sont terminés.

Pour plus d'informations sur le traitement effectué sur chaque poste de travail concerné par les activités du plan de production courant, voir «Processus de poste de travail Tivoli Workload Scheduler», à la page 33.

Remarque : Les modifications apportées à l'aide de **conman** au plan de production en cours de traitement n'affectent pas les définitions dans la base de données, mais

la réplique des données du plan dans la base de données est mise à jour conformément aux modifications. Les modifications des objets dans la base de données n'affectent pas le plan tant qu'il n'est pas étendu ou recréé à l'aide du script **JnextPlan** ou de l'interface de ligne de commande **planman**. Les mises à jour des objets de la base de données n'affectent pas les instances de ces objets résidant déjà dans le plan de production.

Plan de préproduction

Le plan de préproduction permet d'identifier à l'avance les instances de flot de travaux et les dépendances des flots de travaux intervenant pendant la période indiquée.

Cette approche améliore les performances lors de la génération du plan de production en préparant à l'avance un calendrier à haut niveau de la charge de travail de production prévue.

Le plan de préproduction contient :

- Les instances de flot de travaux à exécuter pendant l'intervalle de temps couvert.
- Les dépendances de prédécesseur/successeur externes entre les flots de travaux et les travaux des différents flots.

Un travail ou un flot de travaux qui ne peut pas démarrer avant qu'un autre travail ou flot de travaux ne soit terminé est nommé *successeur*. Un travail ou un flot de travaux externe qui doit se terminer avant que le travail ou le flot de travaux successeur ne puisse démarrer est nommé *prédécesseur*.

Tivoli Workload Scheduler génère, étend et met à jour, si nécessaire, le plan de préproduction automatiquement en procédant comme suit :

- Il supprime les instances de flot de travaux à l'état COMPLETE et CANCEL.
- Sélectionne tous les flots de travaux planifiés après la fin du plan de production en cours et génère leurs instances.
- Il résout toutes les dépendances de travaux et de flots de travaux, y compris les dépendances de prédécesseur/successeur externes, en fonction des critères de correspondance définis.

Pour éviter les conflits, la base de données est verrouillée pendant la génération du plan de préproduction et elle est déverrouillée lorsque la génération est terminée ou lorsqu'une erreur se produit.

A cette étape, seuls les flots de travaux, l'heure à laquelle ils doivent commencer et leurs dépendances sont mis en évidence. Les autres informations concernant les flots de travaux et les autres objets de planification (agendas, invites, domaines, postes de travail, ressources, fichiers et utilisateurs) qui seront concernés par le plan de production pour cette période ne sont pas incluses. Elles seront cependant extraites de la base de données au début de la génération du plan de production.

Lors de l'extension du plan de production, les anciennes instances de flot de travaux sont automatiquement retirées. Les critères utilisés pour retirer ces instances prennent en compte les informations suivantes :

- La première instance de flot de travaux qui n'est pas à l'état COMPLETE au moment de la génération du nouveau plan (FNCJSI). Cette instance de flot de travaux peut être planifiée, à savoir ajoutée au plan lorsque le plan de production est généré. Elle peut aussi être soumise à partir de la ligne de commande pendant la production à l'aide de la commande **conman sbs**.

- La période entre le démarrage prévu de FNCJSI et l'heure de fin du plan de production précédent.

En supposant que cette période est **T**, l'algorithme utilisé pour définir les instances de flot de travaux à retirer du plan de préproduction est le suivant :

si T < 7
Toutes les instances de flot de travaux datant de plus de 7 jours avant l'heure de début du nouveau plan de production sont retirées du plan de préproduction ; toutes les instances de flot de travaux de moins de 7 jours avant cette heure sont conservées quel que soit leur état.

si T > 7
Toutes les instances de flot de travaux antérieures à FNCJSI sont retirées du plan de préproduction et toutes les instances de flot de travaux ultérieures sont conservées.

Cet algorithme est utilisé pour éviter un accroissement continu de la taille du plan de préproduction et, par la même occasion, pour éviter la suppression d'une instance de flot de travaux susceptible de précéder un flot de travaux récemment ajouté au nouveau plan de préproduction.

Pour plus d'informations sur la façon d'ouvrir le plan de préproduction en mode vue à partir de Dynamic Workload Console, voir Dynamic Workload Console - Guide d'utilisation, section "Affichage du plan de préproduction".

Remarque : Dans la terminologie de Tivoli Workload Scheduler for z/OS, le concept qui correspond au plan de préproduction est le *plan à long terme*.

Identification des instances de flot de travaux dans le plan

Dans les versions antérieures à la version 8.3 le plan avait une durée fixe d'un jour. Depuis la version 8.3 le plan peut couvrir une période durant plusieurs jours ou moins d'un jour. Ce changement a ajouté la possibilité d'avoir dans le même plan plus d'une instance du même flot de travaux avec le même nom et également la nécessité de définir une nouvelle convention pour identifier de façon unique chaque instance de flot de travaux dans le plan. Chaque instance de flot de travaux est identifiée dans le plan par les valeurs suivantes :

poste de travail
Indique le nom du poste de travail sur lequel l'exécution du flot de travaux est planifiée.

nom_flot_travaux
Nom du flot de travaux utilisé dans les versions antérieures de Tivoli Workload Scheduler.

scheddateandtime
Date et heure de démarrage prévues de l'instance de flot de travaux dans le plan de préproduction. Elles correspondent au jour indiqué par une clause **on** dans le cycle d'exécution dans la définition du flot de travaux et à l'heure indiquée par un mot clé **at** ou **schedtime** dans la définition du flot de travaux. S'il est défini, le mot clé **schedtime** est utilisé uniquement pour mettre dans l'ordre chronologique les instances de flot de travaux dans le plan de préproduction, alors que le mot clé **at**, s'il est défini, représente également une dépendance pour le flot de travaux. Pour plus d'informations sur ces mots clés, voir «on», à la page 267, «at», à la page 242 et «schedtime», à la page 277.

A partir de ces deux valeurs, que vous pouvez spécifier dans la définition du flot de travaux, Tivoli Workload Scheduler génère et affecte un identificateur alphanumérique unique à chaque instance de flot de travaux, à savoir***id_flot_travaux***, pour son usage interne. Pour plus d'informations relatives au format de l'*ID_flot_travaux*, voir «showjobs», à la page 451.

Vous pouvez utiliser au choix les deux types d'identificateur *poste_de_travail#nom_flot_travaux* et *date_et_heures_prévues* au lieu de *poste_de_travail#id_flot_travaux*, pour identifier de façon unique une instance de flot de travaux lorsque vous gérez des flots de travaux dans le plan avec le programme de ligne de commande **conman**. La convention par défaut utilisée pour identifier une instance de flot de travaux, à la fois dans le présent guide et dans les interfaces de ligne de commande du produit, est celle qui utilise *poste_de_travail_nom_flot_travaux* et *date_et_heures_prévues*. Pour plus d'informations sur la définition d'une instance de flot de travaux dans une commande à l'aide de **conman**, voir «Sélection de flots de travaux dans les commandes», à la page 388.

Gestion des dépendances externes de prédécesseur/ successeur des travaux et des flots de travaux

Lors de la création du plan de préproduction, toutes les dépendances externes de prédécesseur/successeur vers des flots de travaux et des travaux sont résolues avec quatre *critères de correspondance* possibles :

Même jour

Le programme prend en compte les instances de travail ou de flot de travaux qui doivent s'exécuter le même jour. Dans ce cas, la clause **follows**...**sameday** est indiquée dans la définition de l'objet. La figure 6 montre un flot de travaux nommé `Js1` doté d'une dépendance externe de prédécesseur/successeur avec l'instance du flot de travaux `Js2` planifiée pour démarrer le même jour.

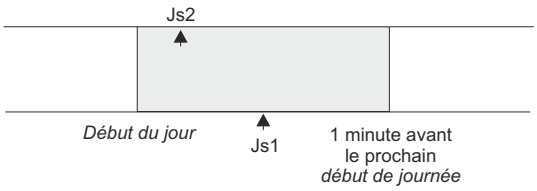


Figure 6. Instances devant s'exécuter le même jour utilisées comme critère de correspondance

Voici un exemple décrivant la définition des flots de travaux impliqués.

schedule Js2
on everyday
at 0700
:job2
end

schedule Js1
on everyday
at 1000
follows wk1#Js2 sameday
:job1
end

Le flot de travaux `Js1` n'est pas lancé tant que l'instance de flot de travaux `Js2` sur le poste de travail `wk1` n'a pas abouti.

Valeur précédente la plus proche

Utilisation de l'instance de flot de travaux ou de travail précédente la plus proche (plus tôt ou simultanément). L'instance de travail ou de flot de travaux utilisée par Tivoli Workload Scheduler pour résoudre la

dépendance est la plus proche dans le temps précédant l'instance qui inclut la dépendance. Dans ce cas, la clause **follows ... previous** est indiquée dans la définition de l'objet. La figure 7 illustre un flot de travaux nommé `Js1` doté d'une dépendance de prédécesseur/successeur externe provenant de l'instance précédente la plus proche du flot de travaux `Js2`. La fenêtre de temps dans laquelle le prédécesseur est recherché apparaît en gris dans la figure.

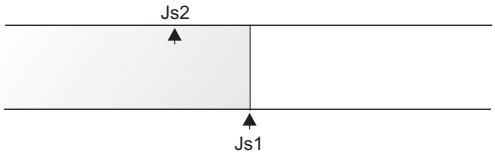


Figure 7. Instance précédente la plus proche utilisée comme critère de correspondance

Voici un exemple décrivant la définition des flots de travaux impliqués.

schedule Js2
on Th
at 0700
:job2
end

schedule Js1
on Fr
at 1000
follows wk1#Js2 previous
:job1
end

Le flot de travaux `Js1` n'est pas lancé tant que l'instance de flot de travaux `Js2` précédente la plus proche sur le poste de travail `wk1` n'a pas abouti.

Dans un intervalle relatif

Prend en compte les instances de travail ou de flot de travaux définies dans une plage avec un décalage relatif à l'heure de début du travail ou du flot de travaux dépendant. Par exemple, de -25 heures avant l'heure de début du flot de travaux dépendant jusqu'à -5 heures après l'heure de début du flot de travaux dépendant. Dans ce cas, la clause **follows ... relative from ... to ...** est indiquée dans la définition de l'objet. La figure 8 montre un flot de travaux nommé `Js1` doté d'une dépendance de prédécesseur/successeur externe sur l'instance de flot de travaux `Js2` qui commence avec un décalage de 2 heures par rapport à `Js1`. L'instance de travail ou de flot de travaux prise en compte par Tivoli Workload Scheduler pour résoudre la dépendance est la plus proche dans l'intervalle de temps relatif que vous avez choisi.

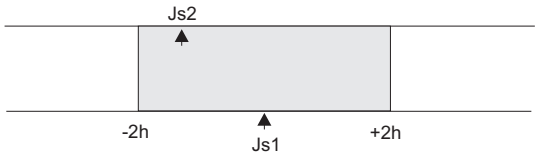


Figure 8. Instances définies dans un intervalle relatif utilisées comme critère de correspondance

Voici un exemple décrivant la définition des flots de travaux impliqués.

schedule Js2
on everyday
at 0900
:job2

schedule Js1
on everyday
at 1000
follows wk1#Js2 relative from 0200 to 0200

schedule Js2	**schedule Js1**
end	:job1
	end

Le flot de travaux Js1 n'est pas lancé tant que l'instance de flot de travaux Js2 du poste de travail wk1, s'exécutant entre 8h et 12h, n'a pas abouti.

Dans un intervalle absolu

Le programme utilise uniquement les instances de travail ou de flot de travaux définies dans un intervalle, par exemple entre aujourd'hui 6h et après-demain 5h59. Dans cas, la clause **follows ... from ... to ...** est indiquée dans la définition de l'objet. La figure 9 illustre un flot de travaux nommé Js1 doté d'une dépendance de prédécesseur/successeur externe sur l'instance du flot de travaux Js2 positionnée dans le plan de préproduction entre 7:00 et 9:00. L'instance de travail ou de flot de travaux prise en compte par Tivoli Workload Scheduler pour résoudre la dépendance est la plus proche dans l'intervalle de temps absolu que vous avez choisi. L'intervalle de temps indique le moment de la journée auquel l'intervalle commence et se termine, le même jour que l'instance qui inclut la dépendance ou un jour défini relatif à ce jour.

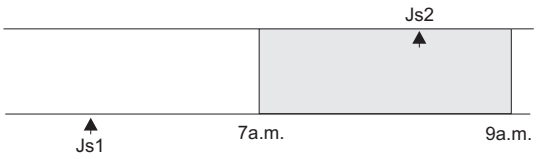


Figure 9. Instances définies dans un intervalle absolu utilisées comme critère de correspondance

Voici un exemple décrivant la définition des flots de travaux impliqués.

schedule Js1	**schedule Js2**
on everyday	on everyday
at 0800	at 1000
:job2	follows wk1#Js1 from 0700 to 0900
end	:job1
	end

Le flot de travaux Js1 n'est pas lancé tant que l'instance de flot de travaux Js2 du poste de travail wk1, s'exécutant entre 7h et 9h le même jour, n'a pas abouti.

Quels que soient les critères de correspondance utilisés, si plusieurs instances d'un flot de travaux predecessor potentielle existent dans l'intervalle de temps spécifié, la règle utilisée par le produit pour identifier l'instance predecessor correcte est la suivante :

1. Tivoli Workload Scheduler recherche l'instance la plus proche précédant l'heure de début du travail ou du flot de travaux dépendant. Si une telle instance existe, il s'agit de l'instance prédécesseur.
2. S'il n'y a pas d'instance précédente, Tivoli Workload Scheduler considère l'instance prédécesseur correcte comme étant l'instance la plus proche qui commence après l'heure de début du travail ou du flot de travaux dépendant.

Ce comportement s'applique aux dépendances externes de prédécesseur/successeur entre les flots de travaux. Pour les dépendances externes de prédécesseur/successeur d'un flot de travaux ou d'un travail avec un autre travail,

le programme met les critères en correspondance en prenant en compte l'heure de début du flot de travaux hébergeant le travail remplacé, au lieu de l'heure de début de ce travail proprement dit. La figure 10 indique en gras les instances de `job1` dont dépend le travail ou le flot de travaux remplaçant.

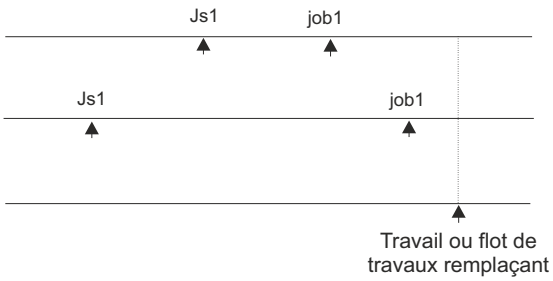


Figure 10. Travail remplacé le plus proche

Les dépendances externes de prédécesseur/successeur sont identifiées entre les travaux et les flots de travaux stockés dans la base de données et dont les instances sont ajoutées au plan de préproduction lors de sa création ou de son extension automatique. Les instances de travail et de flot de travaux soumises en production à partir de la ligne de commande **conman** sont écrites dans le plan de préproduction mais ne sont pas utilisées pour recalculer les prédécesseurs des dépendances externes de prédécesseur/successeur déjà résolues dans le plan de préproduction.

L'agenda classe les dépendances de prédécesseur/successeur comme *internes* lorsqu'elles ne sont spécifiées que par leur nom de travail dans le flot de travaux. Il les classe en temps que dépendances *externes* lorsqu'elles sont spécifiées dans le format *jobStreamName.workstationName.jobName*.

Lorsqu'un flot de travaux inclut un travail possédant une dépendance de prédécesseur/successeur qui partage le même nom de flot de travaux (par exemple, le flot de travaux `schedA` inclut un travail appelé `job6` qui possède une dépendance de prédécesseur/successeur sur `schedA.job2`), la dépendance est ajoutée au plan en tant que dépendance de prédécesseur/successeur *externe*. Depuis la version 8.3, contrairement aux versions précédentes, étant donné que l'agenda utilise le critère de correspondance `sameday` pour résoudre des dépendances externes, les dépendances générées de cette manière ne sont jamais ajoutées lors de la première soumission de l'objet.

Un travail ou un flot de travaux qui n'est pas encore inclus dans le plan de production mais constitue un prédécesseur potentiel d'instances de travaux et de flots de travaux ajoutés au plan de production lors de son extension, est appelé ***prédécesseur suspendu***. Un prédécesseur suspendu s'apparente à une occurrence factice créée par le processus de planification pour respecter une dépendance qui a été résolue dans le plan de préproduction, mais qui ne peut pas l'être dans le plan de production en cours parce que l'heure de début du prédécesseur n'est pas antérieure à l'heure de fin du plan de production en cours. La figure 11, à la page 69 montre comment un prédécesseur suspendu et son successeur sont positionnés dans le plan de préproduction.

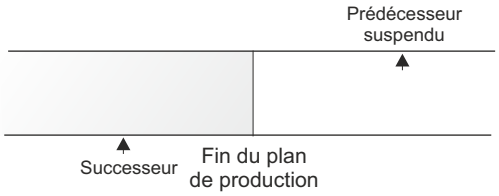


Figure 11. Instance de prédécesseur suspendu

La gestion des prédécesseurs suspendus est étroitement liée au report ou non du travail ou flot de travaux remplaçant dans le plan suivant :

- Si le successeur est reporté lors de l'extension du plan de production, le prédécesseur est inclus dans le nouveau plan de production et la dépendance devient valide. Un travail de prédécesseur ou un flot de travaux suspendu est indiqué par un `[P]` dans la colonne `Dependances` de la sortie des commandes conman showjobs et conman showschedules.
- Si le successeur n'est pas reporté lors de l'extension du plan de production, le prédécesseur est inclus dans le nouveau plan de production mais la dépendance devient ***orpheline***. Ce cas peut se présenter, par exemple, si lors de l'extension du plan de production, le successeur est reporté et le prédécesseur suspendu n'est pas ajouté au plan car il a été marqué *en projet* dans la base de données. Les dépendances orphelines sont indiquées par un `[O]` dans la colonne `Dependencies` de la sortie de la commande conman showjobs. Lorsque vous avez affaire à une dépendance orpheline, vous devez vérifier si elle peut être libérée et l'annuler, si c'est le cas.

Notez que, lorsqu'un réseau Tivoli Workload Scheduler comporte des agents s'exécutant sur des versions antérieures à la 8.3 et que la valeur `yes` est attribuée à l'option `enLegacyId` sur le gestionnaire de domaines maîtres, la présence de plusieurs instances d'un flot de travaux en tant que prédécesseurs suspendus génère des erreurs suite aux incidents d'identification au moment de la soumission.

Résolution de dépendances de prédécesseur/successeur et exemples de transition d'état

Cette section comprend des exemples de chacun des quatre critères de correspondance décrits dans les paragraphes précédents. Dans tous les exemples, le début de l'heure de journée est défini à 06:00 AM.

Même jour

L'instance du travail ou du flot de travaux à prendre en compte dans la résolution de la dépendance est le plus proche le jour où l'instance qui inclut la dépendance est planifiée pour s'exécuter. Dans cet exemple, deux flots de travaux, `Js1` et `Js2`, ont chacun un travail. Le flot de travaux `Js1` est planifié pour s'exécuter tous les jours à 08:00 et le jeudi également à 07:00. `Js1.Job1` s'exécute à 09:00. Le flot de travaux `Js2` n'a pas de restrictions de temps et est planifié par défaut à la première heure de journée définie. `Js2.Job2` est planifié pour s'exécuter à 15:00 et possède une dépendance de prédécesseur/successeur externe sur l'instance précédente la proche du flot de travaux `Js1` qui s'exécute le même jour. Les deux flots de travaux sont définis de la manière suivante :

```
SCHEDULE MY_MASTER#JS1
ON RUNCYCLE RULE1 "FREQ=WEEKLY;BYDAY=TH"
(AT 0700)
ON RUNCYCLE RULE2 "FREQ=DAILY"
(AT 0800)
```

```
:
MY_MASTER#JOB1
AT 0900
END

SCHEDULE MY_MASTER#JS2
ON RUNCYCLE RULE2 "FREQ=DAILY;"
FOLLOWS MY_MASTER#JS1.@ SAMEDAY
:
MY_MASTER#JOB2
AT 1500
END
```

Lorsque les calendriers sont inclus dans le plan, la séquence de graphiques illustre comment la dépendance est résolue :

1. Le jeudi, l'instance du flot de travaux Js2 planifié à 06:00 dépend de l'instance de Js1 planifié pour s'exécuter à 07:00. Tous les autres jours de la semaine, Js2 possède une dépendance sur l'instance de Js1 planifiée à 08:00. La figure 12 présente l'état des deux flots de travaux du plan à 06:00 (heure de début de journée) le jeudi :

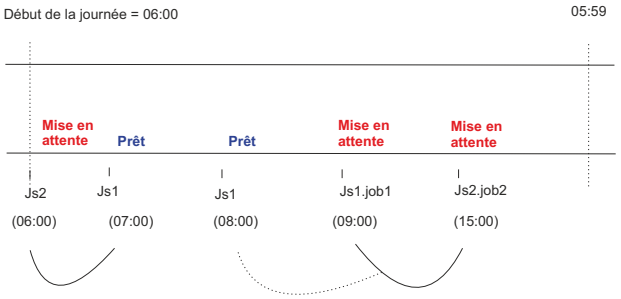


Figure 12. Critères de correspondance du même jour - Etape 1 : Au début de la journée le jeudi

2. A 09:00, Js1.job1 commence et Js1 change d'état. Js2.job2 est mis en attente jusqu'à son heure planifiée. La figure 13 présente l'état des flots de travaux dans le plan à 09:00.

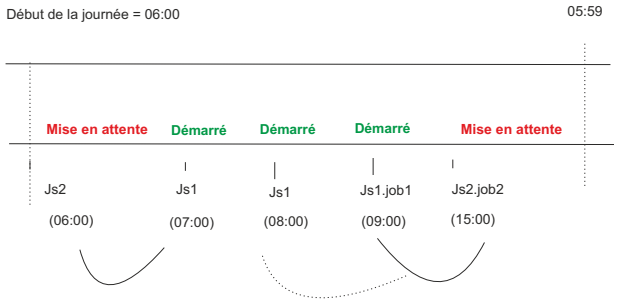


Figure 13. Critères de correspondance du même jour - Etape 2 : à 9:00

3. Le jeudi à 15:00, Js2 passe à l'état prêt et Js2.job2 commence. La figure 14, à la page 71 présente l'état des deux flots de travaux du plan à 15:00.

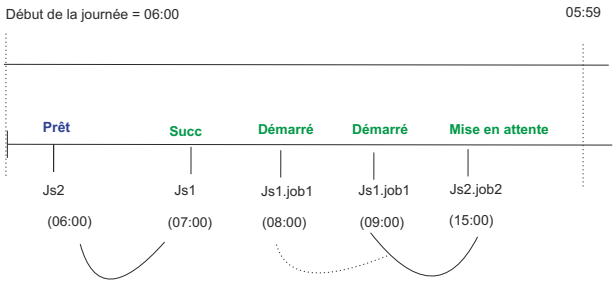


Figure 14. Critères de correspondance du même jour - Etape 3 : à 15:00

Valeur précédente la plus proche

- Dans cet exemple, deux flots de travaux, `Js1` et `Js2`, ont chacun un travail. Le travail dans `Js2` possède une dépendance de prédécesseur/successeur externe sur l'instance précédente la plus proche du travail dans `Js1`. Les deux flots de travaux sont définis de la manière suivante :

```
SCHEDULE MY_MASTER#JS1
ON RUNCYCLE RULE1 "FREQ=DAILY;"
(AT 0800)
ON RUNCYCLE RULE2 "FREQ=WEEKLY;BYDAY=TH,FR"
(AT 0900)
:
MY_MASTER#JOB1
END

SCHEDULE MY_MASTER#JS2
ON RUNCYCLE RULE1 "FREQ=DAILY;"
(AT 1200)
FOLLOWS MY_MASTER#JS1.@ PREVIOUS
:
MY_MASTER#JOB2
AT 1500
END
```

  Le flot de travaux `Js1` s'exécute tous les jours à 0800 et également le jeudi et le vendredi à 0900. Le flot de travaux `Js2` s'exécute tous les jours à 1200, et possède une dépendance externe sur l'instance précédente la plus proche de `Js1`. Lorsque les flots de travaux sont inclus dans le plan, la séquences de graphiques illustre comment la dépendance est résolue :

  1. Avant 12:00 le jeudi et le vendredi, il y a deux instances de `Js1.Job1`. Le flot de travaux `Js2` possède une dépendance sur l'instance de `Js1.Job1` qui est planifiée pour s'exécuter à 09:00, car il s'agit de l'instance précédente la plus proche en terme de temps. La figure 15, à la page 72 présente l'état des deux flots de travaux du plan le jeudi et le vendredi.

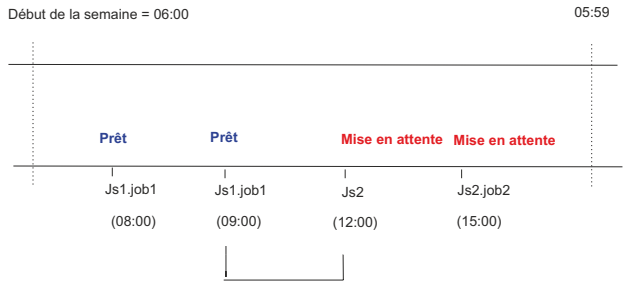


Figure 15. Critères de correspondance précédents les plus proches - Etape 1 : avant 08:00

2. Tous les autres jours de la semaine, la seule instance de `Js1.Job1` dans le plan est celle planifiée pour s'exécuter à 08:00. Dans ce cas, `Js2` possède une dépendance sur cette instance. Lorsque `Job1` aboutit, l'état de `Js2` devient **Ready** (prêt). La figure 16 présente l'état des deux flots de travaux du plan tous les autres jours de la semaine à l'exception du jeudi et du vendredi.

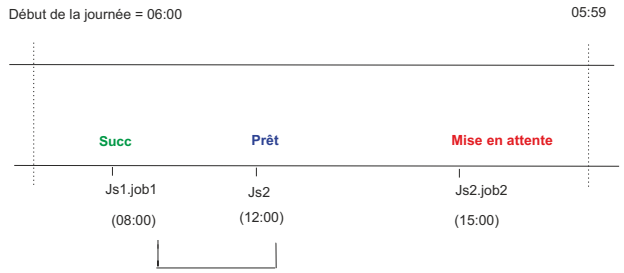


Figure 16. Critères de correspondance précédents les plus proches - Etape 2 : à 8h les jours de la semaine excepté le jeudi et le vendredi

3. Le jeudi et le vendredi à 09:00, la seconde instance de `Js1.Job1` s'exécute avec succès. Le flot de travaux `Js2` passe à **Ready**. `Js2.Job2` est mis en attente jusqu'à son heure de début planifiée. La figure 17 présente l'état des deux flots de travaux dans le plan.

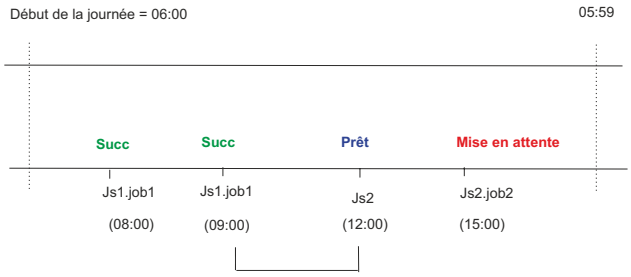


Figure 17. Critères de correspondance précédents les plus proches - Etape 3: à 09:00 le jeudi et le vendredi

4. A 15:00, la dépendance horaire de `Js2.Job2` est satisfaite et `Job2` démarre. La figure 18, à la page 73 présente l'état des deux flots de travaux dans le plan à 15:00.

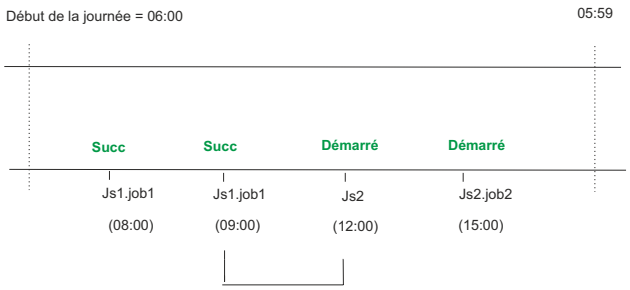


Figure 18. Critères de correspondance précédents les plus proches - Etape 4 : à 15:00 tous les jours

Dans la définition de flot de travaux, le cycle d'exécution *Rule1* peut être remplacé par les mots clés `ON EVERYDAY`.

- Dans ce second exemple, est décrite la différence entre l' utilisation de `sameday` et `closest preceding` correspondant à des critères dans un plan. Le flot de travaux`Js1` s'exécute tous les vendredi à 09h00, tandis que les flots de travaux `Js2` et `Js3` s'exécutent tous les samedi à 09h00. Les trois flots de travaux sont définis comme suit :

```
SCHEDULE ACCOUNTING#JS1
ON RUNCYCLE RULE1 "FREQ=WEEKLY;BYDAY=FR"
:
ACCOUNTING#JOB1
 AT 0900
END

SCHEDULE ACCOUNTING#JS2
ON RUNCYCLE RULE2 "FREQ=WEEKLY;BYDAY=SA"
FOLLOWS ACCOUNTING#JS1.@ PREVIOUS
:
ACCOUNTING#JOB1
 AT 0900
END

SCHEDULE ACCOUNTING#JS3
ON RUNCYCLE RULE2 "FREQ=WEEKLY;BYDAY=SA"
FOLLOWS ACCOUNTING#JS1.@
:
ACCOUNTING#JOB1
 AT 0900
END
```

Le flot de travaux `Js2` possède une dépendance externe sur l'instance précédente la plus proche de `Js1`, qui est résolue comme décrit dans l'exemple précédent. Le flot de travaux `Js3` est défini par le critère `sameday`, ainsi, il ne possède pas de dépendance avec le flot de travaux `Js1`, puisque `Js1` n'est pas défini pour s'exécuter le même jour que `Js2`.

Dans un intervalle relatif

Dans cet exemple, l'instance de travail ou de flot de travaux qui résout la dépendance est la plus proche, dans un intervalle de temps de votre choix, définie relativement à l'heure à laquelle l'instance qui inclut la dépendance est planifiée pour s'exécuter. Le flot de travail `Js1` est planifié pour s'exécuter tous les jours à 15:00 et également le jeudi à 08:00. `Js2` est planifié pour s'exécuter tous les jours à 13:00 et également le jeudi à 06:00 ; étant donné qu'aucune heure spécifique n'est définie dans le cycle d'exécution, il est planifié au début de la journée. `Js2` utilise le critère d'intervalle relatif (-04:00 à +04:00) pour déterminer l'instance qui est

utilisée pour résoudre la dépendance. L'intervalle est fonction de l'heure à laquelle le flot de travaux entre dans le plan. Les flots de travaux sont définis comme suit :

```
SCHEDULE MY_MASTER#JS1
ON RUNCYCLE RULE1 "FREQ=WEEKLY;BYDAY=TH"
(AT 0800)
ON RUNCYCLE RULE2 "FREQ=DAILY"
(AT 1500)
:
MY_MASTER#JOB1
END

SCHEDULE MY_MASTER#JS2
ON RUNCYCLE RULE3 "FREQ=WEEKLY;BYDAY=TH"
ON RUNCYCLE RULE2 "FREQ=DAILY;"
(AT 1300)
FOLLOWS MY_MASTER#JS1.@
RELATIVE FROM -0400 TO 0400
:
MY_MASTER#JOB2
AT 1300
END
```

Lors de la création du plan, **conman showjobs** génère le résultat suivant :

```
%sj @#@
                                                (Est) (Est)
CPU      Schedule SchedTime  Job      State Pr Start  Elapse RetCode Deps
MY_MASTER#JS1   0800 11/13 ******** READY 10    (00:06)
                             JOB1     HOLD  10       (00:06)
MY_MASTER#JS1   1500 11/13 ******** READY 10    (00:06)
                            JOB1     HOLD  10       (00:06)
MY_MASTER#JS2   0600 11/13 ******** HOLD 10       JS1(0800 11/13/09).@
                             JOB2     HOLD  10(13:00)
MY_MASTER#JS2   1300 11/13 ******** HOLD 10(13:00)    JS1(1500 11/13/09).@
                             JOB2     HOLD  10(13:00)
```

La figure 19 présente l'état des flots de travaux du plan au début de la journée le jeudi.

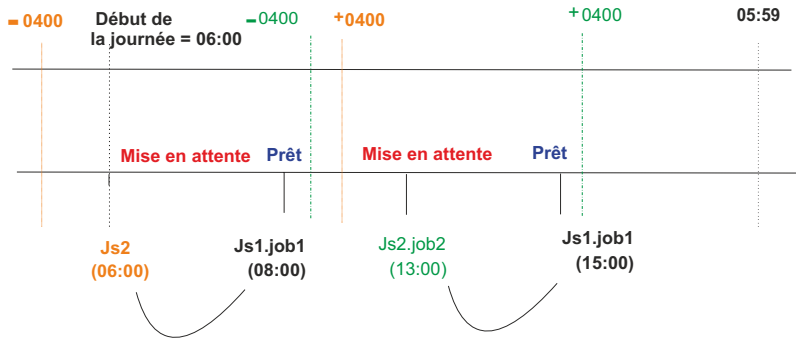


Figure 19. Critères de correspondance d'intervalle relatif - au début de la journée le jeudi

L'instance de Js2 planifié à 06:00 possède une dépendance sur Js1.job1 qui est planifié à 08:00, dans l'intervalle relatif basé sur l'heure planifiée (06:00). Js2.job2 dépend de l'instance de Js1.job1 dans l'intervalle relatif basé sur l'heure planifiée (13:00). Lorsque l'instance de Js1.job1 commence à 08:00, l'état de Js2 passe à Ready (prêt). A partir de ce point, la séquence dans laquelle s'exécutent les travaux et flots de travaux suivent le processus classique.

Dans un intervalle absolu

Dans cet exemple, l'instance du travail ou du flot de travaux prise en compte pour résoudre la dépendance est la plus proche dans un intervalle de temps fixe de votre choix. L'intervalle de temps indique le moment de la journée auquel l'intervalle commence et se termine, le même jour que l'instance qui inclut la dépendance ou un jour défini relatif à ce jour. Js1 est planifié pour s'exécuter tous les jours à 08:00 et également le jeudi à 07:00. Le travail Js1.job1 est planifié pour s'exécuter à 09:00. Le flot de travaux Js2 est planifié tous les jours à 10:00 et également le jeudi au début de la journée (06:00) et possède une dépendance sur Js1 en fonction d'un intervalle absolu ayant lieu le même jour entre 06:00 et 11:00. Les flots de travaux sont définis comme suit :

```
SCHEDULE MY_MASTER#JS1
ON RUNCYCLE RULE1 "FREQ=WEEKLY;BYDAY=TH"
(AT 0700)
ON RUNCYCLE RULE2 "FREQ=DAILY"
(AT 0800)
:
MY_MASTER#JOB1
AT 0900
END

SCHEDULE MY_MASTER#JS2
ON RUNCYCLE RULE3 "FREQ=WEEKLY;BYDAY=TH"
ON RUNCYCLE RULE2 "FREQ=DAILY;"
(AT 1000)
FOLLOWS MY_MASTER#JS1.@ FROM 0600 TO 1100
:
MY_MASTER#JOB2
AT 1300
END
```

Lors de la création du plan, **conman showjobs** génère le résultat suivant :

```
%sj @#@
                                              (Est) (Est)
CPU      Schedule SchedTime  Job      State Pr Start  Elapse RetCode Deps
MY_MASTER#JS1   0700 11/13******** READY 10    (00:06)
                             JOB1     HOLD  10(09:00)(00:06)
MY_MASTER#JS1   0800 11/13 ******** READY 10    (00:06)
                             JOB1     HOLD  10(09:00)(00:06)
MY_MASTER#JS2   0600 11/13 ******** HOLD 10        JS1(0700 11/13/09).@
                             JOB2     HOLD  10(15:00)
MY_MASTER#JS2   1000 11/13 ******** HOLD 10(10:00)    JS1(0800 11/1309).@
                             JOB2     HOLD  10(15:00)
```

La figure 20 présente l'état des flots de travaux du plan au début de la journée le jeudi.

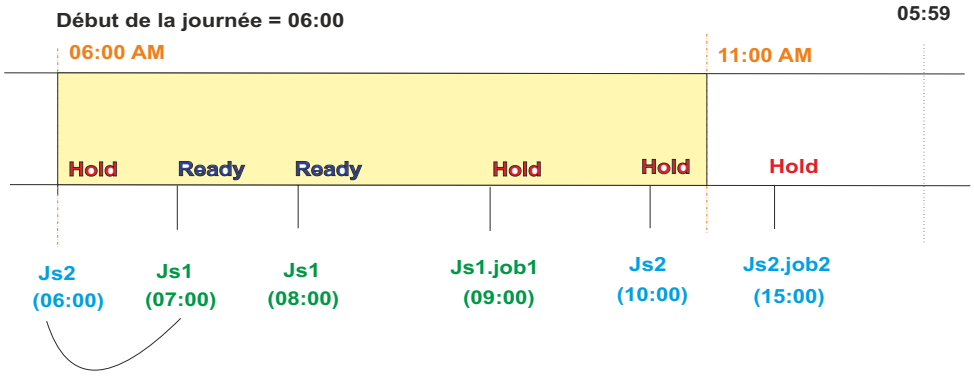


Figure 20. Critères de correspondance d'intervalle absolu - au début de la journée le jeudi

A 09:00, Js1.job1 commence et à 10:00 la dépendance est libérée et Js2 devient prêt. A partir de ce point, la séquence est la même que celle décrite dans les critères de correspondance précédents.

Plan de production

Après avoir créé ou mis à jour le plan de préproduction, Tivoli Workload Scheduler complète les informations stockées dans ce plan par des informations stockées dans la base de données sur les opérations à réaliser dans la période sélectionnée et les autres objets de planification concernés par le traitement du plan, puis le copie dans un nouveau fichier Symphony. Il ajoute également à ce fichier les dépendances entre les plans, comme les flots de travaux reportés à partir du plan de production déjà traité, puis il archive l'ancien fichier Symphony dans le répertoire schedlog.

A la fin de ce processus, le nouveau fichier Symphony contient toutes les informations permettant d'implémenter le nouveau plan de production. De plus, toutes les données du plan sont répliquées dans la base de données afin de faciliter les demandes émises à partir de Dynamic Workload Console et des API.

Une copie du fichier Symphony est ensuite distribuée à tous les postes de travail concernés par l'exécution de travaux ou de flots de travaux pour ce plan de production.

Dans le fichier de sécurité, l'autorisation utilisateur nécessaire pour générer le plan de production est le mot clé d'accès *build* sur les fichiers prodsked et Symphony.

Remarque : Pour éviter le manque d'espace disque, n'oubliez pas que chaque instance de travail ou de flot de travaux augmente la taille du fichier Symphony de 512 octets.

Pour plus d'informations sur la génération du plan de production, voir «Création et extension d'un plan de production», à la page 84.

Options de report

Les flots de travaux sont reportés lors de la génération du plan de production. Le report des flots de travaux dépend des éléments suivants :

- Mot clé **carryforward** du flot de travaux. Voir «carryforward», à la page 243.
- Option globale **enCarryForward**. Voir *IBM Tivoli Workload Scheduler - Guide d'administration*.
- Mot clé de ligne de commande **stageman -carryforward**. Voir «Commande stageman», à la page 100.
- Option globale **carryStates**. Voir *IBM Tivoli Workload Scheduler - Guide d'administration*.

Le tableau 12 montre comment les options globales de report fonctionnent ensemble.

Tableau 12. Paramètres des options globales de report

Options globales	Opération de report
enCarryForward=all carryStates=()	Les flots de travaux sont reportés uniquement s'ils ne sont pas terminés. Tous les travaux sont reportés avec les flots de travaux. Cette option est définie par défaut.

Tableau 12. Paramètres des options globales de report (suite)

Options globales	Opération de report
enCarryForward=no	Aucun flot de travaux n'est reporté. Si cette option est réglée sur "Non", les travaux en cours d'exécution sont déplacés vers le flot de travaux USERJOBS.
enCarryForward=yes carryStates=(*states***)**	Les flots de travaux sont reportés uniquement s'ils ont des travaux dans les états indiqués et que le mot clé **carryforward** est présent dans la définition de flot de travaux. Seuls les travaux se trouvant dans les états indiqués sont reportés avec les flots de travaux.
enCarryForward=yes carryStates=()	Les flots de travaux sont reportés uniquement s'ils ne sont pas terminés et que le mot clé **carryforward** est présent dans la définition de flot de travaux. Tous les travaux sont reportés avec les flots de travaux.
enCarryForward=all carryStates=(*states***)**	Les flots de travaux ne sont reportés que s'ils comprennent des travaux dans les états indiqués. Seuls les travaux se trouvant dans les états indiqués sont reportés avec les flots de travaux.

Le tableau 13 montre le résultat des paramètres de report selon l'option globale **enCarryForward** et les mots clés **stageman -carryforward**.

Tableau 13. Paramètres de report résultants

enCarryForward	stageman -carryforward	Paramètre de report résultant
NO	YES	NO
NO	ALL	NO
YES	NO	NO
ALL	NO	NO
ALL	YES	ALL
YES	ALL	ALL
YES	YES	YES

L'option de report présente dans la définition du flot de travaux est permanente. En d'autres termes, un flot de travaux en échec marqué **carryforward** est continuellement reporté jusqu'à ce qu'un des événements ci-dessous se produise :

- Il se termine à l'état SUCC
- Son heure UNTIL est atteinte
- Il est annulé.

La convention de nom de flot de travaux exécutée est affectée par la valeur attribuée à l'option globale ***enLegacyId***. Pour davantage d'informations sur les différents paramètres de cette option, voir «Personnalisation de la gestion du plan à l'aide des options globales», à la page 80.

Remarque : Quelles que soient les options de report définies, les flots de travaux qui ne contiennent pas de travaux ne sont pas reportés.

Si vous définissez **carryStates=(succ)** et **enCarryForward=all** ou **enCarryForward=yes**, le plan de préproduction et le nouveau fichier Symphony ne correspondront pas lors de la prochaine exécution de **JnextPlan**. Effectivement, le

plan de préproduction ne contient pas les instances des flots de travaux qui se sont terminées correctement, contrairement au nouveau fichier Symphony. Suite à cette incohérence, les dépendances ne sont pas résolues en fonction des instances des flots de travaux reportées terminées correctement car celles-ci n'existent plus dans le plan de préproduction.

La décision de reporter un travail répétitif, c'est-à-dire un travail dont la définition contient un paramètre de temps **every** ou une chaîne de travaux réexécutés, repose sur l'état de son exécution la plus récente. Seuls les premier et dernier travaux de la chaîne sont reportés.

Plan d'essai

Un plan d'essai est une prévision de ce que serait un plan de production s'il couvrait une période plus longue. Par exemple, si vous générez un plan de production couvrant deux jours mais voulez savoir à quoi il ressemblerait sur trois jours, vous pouvez créer un plan d'essai.

Voici les caractéristiques d'un plan d'essai :

- Sa date de début concorde avec :
  - La date de début du plan de préproduction.
  - La date de fin du plan de production.
- Il s'appuie sur les informations statiques stockées dans le plan de préproduction courant.
- Il ne peut pas être exécuté pour gérer la production.
- Il peut être géré par des utilisateurs bénéficiant des droits d'accès *build* sur le type d'objet fichier **trialsked** défini dans le fichier de sécurité du gestionnaire de domaine maître.
- Il produit un fichier qui est stocké dans le répertoire schedlog et qui a les propriétés suivantes :
  - Même format que le fichier Symphony.
  - Le nom de ce fichier commence par un T.

Les générations de plan d'essai peuvent rallonger l'heure de fin du plan de préproduction. Cette situation dépend des valeurs des options globales minLen et maxLen. Dans ce cas, la base de données est verrouillée jusqu'à la fin de l'opération.

Toute période peut être sélectionnée sans restriction pour un plan d'essai mais la taille du fichier résultant qui contient toutes les informations du plan d'essai doit être prise en compte.

Comme le plan d'essai est basé sur les informations statiques du plan de préproduction, il ne prend pas en compte les mises à jour effectuées de manière dynamique sur le fichier Symphony pendant le traitement du plan de production. Par conséquent, tous les flots de travaux qu'il contient sont dans un des deux états suivants :

HOLD
S'ils sont dépendants d'autres flots de travaux ou si leur heure de début est postérieure à l'heure de début du plan.

READY
S'ils n'ont aucune dépendance et que leur heure de début est dépassée.

Les opérations qui peuvent être effectuées sur un plan d'essai du gestionnaire de domaine maître sont les suivantes :

Création
Permet de créer un plan d'essai pour avoir un aperçu de la production lorsque le plan de production n'existe pas encore.

extension
Permet de créer un plan d'essai de l'extension du plan de production courant pour avoir un aperçu de l'évolution future de la production.

Pour plus d'informations sur la création ou l'extension d'un plan d'essai, voir «Ligne de commande Planman», à la page 87.

Plan prévisionnel

Le *plan prévisionnel* est une prévision de ce que serait le plan de production sur une période donnée. Par exemple, si vous avez généré un plan de production qui couvre deux journées et que vous souhaitez savoir ce que serait ce plan la semaine suivante, vous pouvez générer un plan prévisionnel.

Voici les caractéristiques d'un plan prévisionnel :

- Il couvre n'importe quelle période future, passée ou recouvrant partiellement le plan courant.
- Il s'appuie sur un échantillon du plan de préproduction qui couvre la même période que le plan prévisionnel. Cet échantillon du plan de préproduction est supprimé après la création du plan prévisionnel.
- Il ne peut pas être exécuté pour gérer la production.
- Il peut être géré par des utilisateurs bénéficiant des droits d'accès *build* sur le type d'objet fichier **trialsked** défini dans le fichier de sécurité du gestionnaire de domaine maître.
- Il produit un fichier qui est stocké dans le répertoire `schedlog` et qui a les propriétés suivantes :
  - Même format que le fichier `Symphony`.
  - Le nom de ce fichier commence par un F.
- Lorsque l'assurance de service de charge de travail est activée, elle peut calculer l'heure de début prévue de chaque travail du flot de travaux. Vous pouvez activer et désactiver cette fonction à l'aide de l'option globale **enForecastStartTime**. Tivoli Workload Scheduler calcule la durée d'exécution moyenne de chaque travail en fonction de toutes les exécutions précédentes. Pour les plans complexes, activer cette fonction peut avoir un impact négatif sur le temps nécessaire pour générer le plan prévisionnel.

Pendant la création du plan prévisionnel, la base de données est verrouillée jusqu'à la fin de l'opération.

Toute période peut être sélectionnée sans restriction pour un plan prévisionnel mais la taille du fichier résultant qui contient toutes les informations du plan prévisionnel doit être prise en compte.

Comme le plan prévisionnel est basé sur les informations statiques de la base de données, il ne prend pas en compte les mises à jour effectuées de manière dynamique sur le fichier `Symphony` pendant le traitement du plan de production ou du plan de préproduction. Par conséquent, tous les flots de travaux qu'il contient sont dans un des deux états suivants :

HOLD
S'ils sont dépendants d'autres flots de travaux ou si leur heure de début est postérieure à l'heure de début du plan.

READY
S'ils n'ont aucune dépendance et que leur heure de début est dépassée.

La seule opération qui peut être effectuée sur un plan prévisionnel du gestionnaire de domaine maître est la suivante :

Création
Permet de créer un plan prévisionnel pour avoir un aperçu de la production à une période donnée.

Pour plus d'informations sur la création d'un plan prévisionnel, voir «Ligne de commande Planman», à la page 87.

Personnalisation de la gestion du plan à l'aide des options globales

Vous pouvez personnaliser certains critères à utiliser par Tivoli Workload Scheduler lors de la gestion des plans en définissant des options spécifiques dans le gestionnaire de domaine maître à l'aide du programme de ligne de commande **optman**. Vous devez générer le plan à nouveau pour activer les nouveaux paramètres. Vous pouvez personnaliser les options suivantes :

Propriétés ayant une incidence sur la génération du plan de préproduction :

minLen
Longueur minimale, en jours, du plan de préproduction laissé sous la forme de tampon à la fin du plan de production qui vient d'être généré. La valeur affectée à cette option est utilisée lors de l'exécution du script **UpdateStats** dans **JnextPlan**. La valeur peut être de 7 à 365 jours. La valeur par défaut est 8 jours.

maxLen
Longueur maximale, en jours, du plan de préproduction laissé sous la forme de tampon à la fin du plan de production qui vient d'être généré. La valeur peut être de 8 à 365 jours. La valeur par défaut est 8 jours.

Si les valeurs `minLen` et `maxLen` sont égales, le plan de préproduction est mis à jour lors de la phase `MakePlan`. En général, la valeur de `maxLen` doit dépasser la valeur de `minLen` d'au moins 1 jour afin que le plan de préproduction puisse être mis à jour lors de la phase `UpdateStats`.

Propriétés ayant une incidence sur la génération ou l'extension du plan de production :

startOfDay
Heure de début de la journée de traitement de Tivoli Workload Scheduler, au format 24 heures : hhmm (0000-2359). Le paramètre par défaut est 0000.

enCarryForward
Cette option affecte la façon dont la commande **stageman** reporte les flots de travaux. Sa valeur détermine si les flots de travaux non terminés sont reportés depuis l'ancien plan de production vers le nouveau. Les valeurs possibles de *enCarryForward* sont **yes, no,** et **all**. La valeur par défaut est **all**.

carryStates
Cette option affecte la façon dont la commande **stageman** gère les travaux des flots de travaux reportés. Sa valeur détermine selon leur état les travaux qui doivent être inclus dans les flots de travaux à reporter. Par exemple si :

```
carryStates='abend exec hold'
```

tous les travaux à l'état `abend`, `exec` ou `hold` sont inclus dans les flots de travaux reportés. La valeur par défaut est :

```
carryStates=null
```

ce qui signifie que tous les travaux sont inclus, quels que soient leurs états.

enCFInterNetworkDeps
Cette option affecte la façon dont la commande **stageman** gère les dépendances interréseaux. Saisissez **yes** pour que tous les flots de travaux `EXTERNES` soient reportés. Saisissez **no** pour désactiver entièrement la fonction de report pour les dépendances interréseaux. La valeur par défaut est **yes.**

enCFResourceQuantity
Cette option affecte la façon dont la commande **stageman** gère les ressources. Lors de l'extension du plan de production, une des situations suivantes se présente :

- Une ressource utilisée par aucun flot de travaux reporté du plan de production précédent est référencée par de nouvelles instances de travail ou de flot de travaux ajoutées au nouveau plan de production. Dans ce cas, la quantité de cette ressource figure dans la définition de ressource dans la base de données.
- Une ressource utilisée par un ou plusieurs flots de travaux reportés du plan de production précédent n'est référencée par aucune instance de travail ou de flot de travaux ajoutée au nouveau plan de production. Dans ce cas, la quantité de cette ressource figure dans l'ancien fichier `Symphony`.
- Une ressource utilisée par un ou plusieurs flots de travaux reportés du plan de production précédent est référencée par des instances de travaux ou de flots de travaux ajoutées au nouveau plan de production. Dans ce cas, la quantité de cette ressource dépend de la valeur affectée à l'option *enCFResourceQuantity* :

  Si la valeur YES est attribuée à *enCFResourceQuantity*
  La quantité de la ressource figure dans l'ancien fichier `Symphony`.

  Si la valeur NO est attribuée à *enCFResourceQuantity*
  La quantité de la ressource figure dans la définition de ressource dans la base de données.

La valeur par défaut est **yes.**

enEmptySchedsAreSucc
Cette option régit le comportement des flots de travaux qui ne contiennent pas de travaux. Les valeurs possibles sont les suivantes :

Yes Les flots de travaux qui ne contiennent pas de travaux sont marqués SUCC lorsque leurs dépendances sont résolues.

No Les flots de travaux qui ne contiennent pas de travaux restent à l'état READY.

enPreventStart
Cette option permet de gérer, pour les plans de production étalés sur plusieurs jours, les flots de travaux sans contrainte temporelle **at** définie. Elle est utilisée pour éviter que les instances de flot de travaux non soumises à une dépendance horaire ne démarrent toutes en même temps lors de la création ou de l'extension du plan de production. Les valeurs possibles sont les suivantes :

Yes Un flot de travaux ne peut pas démarrer avant le début de la journée *startOfDay* défini dans son heure planifiée, même s'il n'a pas de dépendances.

No Un flot de travaux peut démarrer immédiatement au lancement du plan de production si toutes ses dépendances sont résolues.

enLegacyId
Cette option affecte la manière de nommer les flots de travaux dans le plan. Elle a pour fonction de maintenir la cohérence lors de l'identification des flots de travaux dans le plan dans les environnements mixtes avec des versions de Tivoli Workload Scheduler antérieures à la 8.3 gérées par la version 8.4 gestionnaire de domaine maîtres. Cette option n'est pas prise en charge par le catalogue libre-service, lequel l'ignore, même si sa valeur est définie sur OUI. La valeur affectée à cette option est lue soit lors de la création ou de l'extension du plan de production, soit lors de la soumission des flots de travaux dans la production via la commande **conman**. Les valeurs possibles sont les suivantes :

Yes L'identificateur *ID_flot_de_travaux* du flot de travaux nommé *nom_flot_de_travaux* est défini sur la valeur *jobstream_nameN*, où *N* est un nombre incrémentiel attribué par un compteur interne. Ce paramètre vaut *null* s'il n'existe qu'une seule instance de ce flot de travaux dans le plan.

Si le flot de travaux intitulé *nom_flot_de_travaux* est reporté par la suite, son identificateur est défini sur la valeur *CFjobstream_nameN*.

Ce paramètre est utile pour assurer la cohérence de gestion des flots de travaux, même en cas de report de ceux-ci via la commande **conman**, lors d'une connexion à un agent Tivoli Workload Scheduler 8.2.x agent sur un réseau Tivoli Workload Scheduler doté de gestionnaire de domaine maître 8.4.

Si, notamment, la période de production dure une journée et qu'un même flot de travaux n'est pas soumis sous forme d'instances multiples, la compatibilité avec les versions antérieures est complète lors de la gestion de flots de travaux en mode production à partir d'un agent Tivoli Workload Scheduler version 8.2.x.

No L'identificateur de flot de travaux *ID_flot_de_travaux* est généré conformément à «showjobs», à la page 451. Les flots

de travaux reportés conservent leurs noms et identificateurs d'origine, et signalent entre accolades ({}) la date de leur report.

logmanSmoothPolicy
Cette option affecte la façon dont la commande **logman** gère les statistiques et l'historique. Elle définit le facteur de pondération qui favorise le travail le plus récemment exécuté lors du calcul du délai d'exécution normal (moyen) d'un travail. Le résultat de ce calcul est exprimé sous la forme d'un pourcentage. La valeur par défaut est **-1**, ce qui signifie que cette propriété est désactivée.

logmanMinMaxPolicy
Cette option permet de définir comment les temps d'exécution minimal et maximal des travaux sont consignés et signalés par **logman**. Les valeurs possibles de l'option *logmanMinMaxPolicy* sont les suivantes :

elapsedtime
Les temps d'exécution minimal et maximal et les dates consignés sont basés uniquement sur le temps écoulé d'un travail. L'activité du système affecte de façon significative le temps écoulé, exprimé en minutes. Ce temps inclut non seulement le temps pendant lequel un travail a utilisé l'UC, mais aussi le temps pendant lequel il a dû attendre que d'autres processus libèrent l'UC. Pendant les périodes d'activité système élevée, par exemple, le temps écoulé pour un travail a pu être long alors que le temps du poste de travail n'aura pas été plus long que dans les périodes d'activité système basse. Les valeurs sont mises à jour uniquement si le temps écoulé de l'exécution de travail la plus récente est supérieur au maximum existant ou inférieur au minimum existant.

cputime
Les temps d'exécution minimal et maximal et les dates consignés sont basés uniquement sur le temps UC d'un travail. Le temps UC est une mesure en secondes du temps pendant lequel un travail a utilisé l'UC ; il ne prend pas en compte les attentes. Les valeurs sont mises à jour uniquement si le temps du poste de travail de l'exécution de travail la plus récente est supérieur au maximum existant ou inférieur au minimum existant.

both Le temps écoulé et le temps UC sont mis à jour indépendamment pour indiquer leurs valeurs maximales et minimales mais les dates d'exécution correspondent uniquement à la valeur du temps écoulé. Dans ce cas, aucun enregistrement des dates d'exécution n'est conservé pour les temps UC maximal et minimal.

La valeur par défaut est **both.**

enTimeZone
Cette option permet d'activer ou désactiver la gestion des fuseaux horaires sur le réseau Tivoli Workload Scheduler. Les valeurs possibles de l'option *enTimeZone* sont les suivantes :

No Désactivation de la gestion des fuseaux horaires. En

d'autres termes, les valeurs attribuées à tous les mots clés **timezone** dans les définitions sont ignorées.

Yes Activation de la gestion des fuseaux horaires. En d'autres termes, les valeurs attribuées aux paramètres **timezone** sont utilisées pour calculer l'heure où les travaux et flots de travaux s'exécuteront sur les postes de travail cibles.

Voir «Activation de la gestion des fuseaux horaires», à la page 653 pour plus d'informations sur la variable *enTimeZone*.

enLegacyStartOfDayEvaluation
Cette option influe sur la manière dont la variable *startOfDay* est gérée sur le réseau Tivoli Workload Scheduler. Pour fonctionner, la valeur **yes** doit être attribuée à la variable *enTimeZone*. Les valeurs possibles de l'option *enLegacyStartOfDayEvaluation* sont les suivantes :

No La valeur affectée à l'option *startOfDay* sur le gestionnaire de domaine maître n'est pas convertie au fuseau horaire local défini sur chaque poste de travail du réseau.

Yes La valeur affectée à l'option *startOfDay* sur le gestionnaire de domaine maître est convertie au fuseau horaire local défini sur chaque poste de travail du réseau.

Voir «Comment Tivoli Workload Scheduler gère les fuseaux horaires», à la page 654 pour plus d'informations sur la variable *enLegacyStartOfDayEvaluation*.

Pour plus d'informations sur la définition des options à l'aide du programme de ligne de commande **optman**, voir *IBM Tivoli Workload Scheduler - Guide d'administration*.

Création et extension d'un plan de production

L'ensemble du processus qui permet de passer d'un ancien plan de production à un nouveau, y compris l'activation sur l'ensemble du réseau Tivoli Workload Scheduler, est géré par le script **JnextPlan**. Vous pouvez exécuter **JnextPlan** à tout moment pendant la journée de traitement. Le nouveau plan de production généré est immédiatement activé sur les postes de travail cible, quelle que soit l'heure définie dans la variable ***startOfDay***. Lorsque la commande **JnextPlan** est exécutée, la variable *$MANAGER* est gérée comme suit :

- La variable est résolue si le poste de travail et un agent tolérant aux pannes d'une version antérieure à 8.6.
- La variable est laissée non résolue pour les postes de travail d'agents tolérants aux pannes version 8.6.

Lorsque vous exécutez le script **JnextPlan**, les processus des postes de travail sont arrêtés et démarrés à nouveau sur tous les postes de travail du réseau Tivoli Workload Scheduler. Pour plus d'informations sur les processus de poste de travail, voir Chapitre 2, «Description des processus et des commandes de base», à la page 33.

Le script **JnextPlan** peut uniquement être exécuté à partir du gestionnaire de domaine maître. Il utilise les paramètres de connexion par défaut définis dans le fichier `localopts` ou `useropts` (voir «Configuration des options pour l'utilisation des interfaces utilisateur», à la page 59). Si vous souhaitez exécuter **JnextPlan** avec

des paramètres de connexion différents, vous pouvez éditer le script **MakePlan** et changer l'appel à l'instruction **planman** comme décrit dans «Ligne de commande Planman», à la page 87.

Le script **JnextPlan** est composé de la séquence de commandes et des scripts spécialisés suivants, chacun gérant un aspect spécifique de la génération du plan de production.

conman startappserver
Cette commande est appelée pour lancer WebSphere Application Server s'il n'est pas déjà en cours d'exécution.

MakePlan
Ce script reçoit de **JnextPlan** les indicateurs et les valeurs qui leurs sont affectés. Sa syntaxe est :

MakePlan [**-from** *mm/jj/[aa]aa[hh[:]mm[**tz** | **timezone** nom_fh]]*] {**-to** *mm/jj/[aa]aa[hh[:]mm[**tz** | **timezone** nom_fh]]* | **-for** *[h]hh[:]mm* [**-days** *n*] | **-days** *n*}

Le script **MakePlan** appelle en interne la ligne de commande **planman**. Il exécute les actions suivantes :

1. Il crée un nouveau plan ou étend le plan courant et stocke les informations dans un plan de production intermédiaire qui contient :
   - Tous les objets de planification (travaux, flots de travaux, agendas, invites, ressources, postes de travail, domaines, fichiers, utilisateurs et dépendances) définis pour la période sélectionnée.
   - Toutes les dépendances entre les nouvelles instances des travaux et flots de travaux et les travaux et flots de travaux de l'ancien plan de production.
   - Toutes les demandes de liaison dont l'heure planifiée est incluse dans la période de temps sélectionnée.
2. Supprime toutes les demandes de liaison à l'état final.
3. Il imprime les rapports de préproduction.

SwitchPlan
Ce script appelle en interne la commande **stageman**. Pour plus d'informations, voir «Commande stageman», à la page 100. Le script **SwitchPlan** exécute les actions suivantes :

1. Arrête les processus Tivoli Workload Scheduler.
2. Il génère le nouveau fichier `Symphony` à partir du plan de production intermédiaire créé par **MakePlan**.
3. Il archive le fichier de l'ancien plan avec la date et l'heure actuelles dans le répertoire `schedlog`.
4. Il crée une copie du fichier `Symphony` à distribuer aux postes de travail.
5. Il redémarre les processus Tivoli Workload Scheduler qui distribuent la copie du fichier `Symphony` aux postes de travail cibles pour exécuter les travaux dans le plan.

Remarque : Assurez-vous qu'aucune commande **conman start** n'est exécutée pendant le traitement du plan de production.

CreatePostReports
Ce script imprime les rapports de postproduction.

UpdateStats
Ce script appelle en interne la commande **logman**. Pour plus

d'informations, voir «Commande stageman», à la page 100. Le script **UpdateStats** exécute les actions suivantes :

1. Il journalise les statistiques des travaux.
2. Il vérifie les règles et étend le plan de préproduction le cas échéant.
3. Il met à jour le plan de préproduction en signalant les états des instances de flot de travaux.

Pour plus d'informations sur l'utilisation du script **JnextPlan**, voir «JnextPlan».

Remarque : Pour obtenir des informations sur les scénarios spécifiques qui peuvent nécessiter la personnalisation **JnextPlan**, consultez *IBM Tivoli Workload Scheduler - Guide d'administration*

JnextPlan

Le script **JnextPlan** gère l'ensemble du processus qui permet de passer d'un ancien plan de production à un nouveau (Symphony), notamment l'activation sur tout le réseau Tivoli Workload Scheduler. Chaque fois que vous exécutez **JnextPlan**, tous les postes de travail sont arrêtés et redémarrés.

Lorsque vous exécutez la commande **JnextPlan**, un fichier journal des travaux est créé dans le répertoire *<REP_INST_TWS>*\TWS\stdlist*<DATE>*, où *<REP_INST_TWS>* est le répertoire d'installation de Tivoli Workload Scheduler et *<DATE>* est la date d'exécution du script.

Autorisation

Vous pouvez exécuter la commande **JnextPlan** à partir d'un shell d'invite de commande sur le gestionnaire de domaine maître si vous êtes l'un des utilisateurs suivants :

- L'utilisateur *utilisateur_TWS* pour lequel vous avez installé le produit sur la machine, s'il n'est pas désactivé par les paramètres définis dans le fichier de sécurité.
- L'utilisateur Root sur les systèmes d'exploitation UNIX ou l'utilisateur administrateur sur les systèmes d'exploitation Windows, s'il n'est pas désactivé par les paramètres définis dans le fichier de sécurité.

Syntaxe

JnextPlan
[**-V** | **-U**] |
[**-from** *mm/jj/[aa]aa[hh[:]mm[**tz** | **timezone** nom_fh]]*]
{**-to** *mm/jj/[aa]aa[hh[:]mm[**tz** | **timezone** nom_fh]]* |
-for *[h]hh[:]mm* [**-days** *n*] | **-days** *n*}
[**-noremove**]

Arguments

-V Affiche la version de la commande et quitte l'application.

-U Affiche des informations sur la syntaxe de la commande et quitte l'application.

-from Définit l'heure de début du plan de production. Le format de la date est défini dans le fichier `localopts`. *hhmm* indique les heures et minutes et

fuseau_horaire est le fuseau horaire. Cet indicateur n'est utilisé que si aucun plan de production n'existe. Si l'argument **-from** n'est pas spécifié, la valeur par défaut est `"today +startOfDay"`.

Si le fuseau horaire n'est pas spécifié, le fuseau GMT est utilisé par défaut.

-to heure de fin du plan de production Le format pour la date est le même que celui utilisé pour l'argument **-from**. L'argument **-to** est mutuellement exclusif avec les arguments **-for** et **-days**.

Si le fuseau horaire n'est pas spécifié, le fuseau GMT est utilisé par défaut.

-for Durée de l'extension de plan. Le format est *hhhmm*, où *hhh* représente les heures et *mm* les minutes. L'argument **-for** est mutuellement exclusif avec **-to**.

-days *n*
Nombre de jours souhaités pour créer ou étendre le plan de production. Le paramètre **-days** est mutuellement exclusif avec le paramètre **-to**.

-noremove
Vérifie que les instances de flot de travaux terminées ne sont pas supprimées du nouveau plan de production.

Si aucun argument **-to**, **-for**, ou **-days** n'est spécifié, la durée du plan de production par défaut est de un jour.

JnextPlan -for 0000

La commande `JnextPlan - for 0000` étend de 0 heures et 0 minutes le plan de production et ajoute dans le plan de production (Symphony) les définitions de poste de travail, d'utilisateur et d'agenda nouvellement créées de la base de données. Elle supprime également toutes les instances de flot de travaux correctement terminées.

Si vous utilisez la commande `JnextPlan - for 0000 - noremove`, toutes les instances de flot de travaux correctement terminées du fichier Symphony ne sont pas supprimées.

Le paramètre de l'option globale `enCarryForward` détermine si les flots de travaux non terminés sont reportés de l'ancien plan de production vers le nouveau. Avant d'exécuter la commande, vérifiez que l'option `enCarryForward` est définie sur `ALL` pour que toutes les instances des flots de travaux non terminés soient reportées dans le nouveau plan de production, ou bien utilisez l'option `-noremove`.

Exemple

En supposant que la valeur affectée à *startOfDay* est 00:00 a.m. et que le format de date défini dans le fichier `localopts` est *mm/jj/aaaa*, si les valeurs définies sont **-from** `07/05/2011` et **-to** `07/07/2011`, alors le plan est créé pour couvrir le cadre temporel allant de 07/05/2011 à 00:00 a.m. à 07/06/2011 à 11:59 p.m. et non pas au 07/07/2011 à 11:59 p.m.

Ligne de commande Planman

La ligne de commande **planman** est utilisée pour gérer les plans de production *intermédiaires*, les plans *d'essai* et les plans *prévisionnels*. Elle est également utilisée pour obtenir des informations sur le plan de production actif, pour déverrouiller les entrées de base de données verrouillées par le processus de gestion de plan

pour déployer les règles d'événement de planification, et pour répliquer les données de plan dans la base de données. Cette commande s'exécute sur le gestionnaire de domaine maître. Utilisez la syntaxe suivante lorsque vous exécutez **planman** :

planman -U

planman -V

planman [*paramètres_connexion*] *commande*

où :

-U Permet d'afficher des informations sur la syntaxe de la commande et de quitter.

-V Permet d'afficher la version de la commande et de quitter.

paramètres_connexion
Si vous utilisez **planman** à partir du gestionnaire de domaine maître, les paramètres de connexion ont été configurés lors de l'installation et n'ont pas besoin d'être fournis, sauf si vous ne voulez pas utiliser les valeurs par défaut.

Si vous utilisez **planman** à partir du client de ligne de commande sur un autre poste de travail, les paramètres de connexion peuvent être fournis par l'une ou plusieurs de ces méthodes :

- Ils sont stockés dans le fichier `localopts`
- Ils sont stockés dans le fichier `useropts`
- Ils sont fournis à la commande dans un fichier de paramètres
- Ils sont fournis à la commande au sein d'une chaîne de commande

Pour une présentation de ces options, voir «Configuration des options pour l'utilisation des interfaces utilisateur», à la page 59. Pour des détails complets sur les paramètres de configuration, voir la rubrique relative à la configuration de l'accès du client de ligne de commande dans *Tivoli Workload Scheduler - Guide d'administration*.

commande
Représente la commande que vous exécutez sur les plans à l'aide de l'interface **planman**. Voici les actions que vous pouvez effectuer avec les plans :

- «Création d'un plan de production intermédiaire», à la page 89
- «Création d'un plan intermédiaire pour une extension de plan», à la page 90
- «Extraction des informations du plan de production», à la page 91
- «Création d'un plan d'essai», à la page 92
- «Création d'un plan d'essai avec une extension de plan de production», à la page 93
- «Création d'un plan prévisionnel», à la page 94
- «Déverrouillage du plan de production», à la page 96
- «Suppression du plan de préproduction», à la page 97
- «Réinitialisation du plan de production», à la page 97
- «Réplication des données de plan dans la base de données», à la page 98

- «Surveillance de la réplication des données de plan dans la base de données», à la page 99

Vous pouvez également utiliser **planman** pour déployer les règles d'événement de planification.. La commande est décrite dans : «Déploiement de règles», à la page 95. Consultez les sous-sections appropriées pour plus d'informations sur ces commandes.

Création d'un plan de production intermédiaire

La commande **planman** avec l'option **crt** est appelée à partir de la commande **JnextPlan** dans une de ces deux situations :

- La première fois que la commande **JnextPlan** est exécutée après l'installation du produit.
- Lors de la génération d'un plan de production après réinitialisation du plan de production à l'aide de la commande **ResetPlan**.

Cette commande entraîne la création d'un nouveau plan de production intermédiaire, appelé `Symnew`, qui couvre la totalité de la durée couverte ultérieurement par le plan de production en cours de génération. La syntaxe utilisée est la suivante :

planman [*paramètres_connexion*] **crt**

[**-from** *mm/jj/[aa]aa* [*hh[:]mm* [**tz** | **timezone** *nom_fh*]]]

{**-to** *mm/jj/[aa]aa[hh[:]mm[**tz** | **fuseau horaire** nom_fh]]* |

-for *[h]hh[:]mm* [**-days** *n*] |

-days *n*}

où :

paramètres_connexion
Définit les paramètres utilisés lors de l'établissement de la connexion avec HTTP ou HTTPS par WebSphere Application Server vers le gestionnaire de domaine maître. Pour plus d'informations, voir «Ligne de commande Planman», à la page 87.

-from Définit l'heure de début du nouveau plan de production.

Si l'argument **-from** est ignoré, alors :

- La date par défaut est *aujourd'hui*.
- L'heure par défaut est la valeur définie dans l'option globale ***startOfDay*** via **optman** sur le gestionnaire de domaine maître.

-to Heure de fin du nouveau plan de production. Le format pour la date est le même que celui utilisé pour l'argument **-from**. L'argument **-to** est mutuellement exclusif avec les arguments **-for** et **-days**.

-for Durée de l'extension de plan. Le format est *hhhmm*, où *hhh* représente les heures et *mm* les minutes. L'argument **-for** est mutuellement exclusif avec l'argument **-to**.

-days *n*
Nombre de jours souhaités pour créer le plan de production. L'argument **-days** est mutuellement exclusif avec l'argument **-to**.

Remarque :

1. Veillez à exécutez la commande **planman** à partir de la commande **JnextPlan**.
2. Le format utilisé pour la date dépend de la valeur attribuée à la variable ***format_date*** dans le fichier `localopts`.

Si aucun argument **-to**, **-for**, ou **-days** n'est spécifié, la durée du plan de production par défaut est de un jour.

Voici quelques exemples d'utilisation de la commande **planman**, supposant que le format de date défini dans le fichier `localopts` est mm/jj/aaaa :

1. Cette commande crée le plan de production à partir du 03/21/2011 à 23:07 jusqu'au 03/22/2011 à 23:06, heure locale :

   ```
   planman crt –from 03/21/05 2307
   ```

2. Cette commande crée le plan de production à partir du 03/21/2011 à 09:00 jusqu'au 03/21/2011 à 15:00 :

   ```
   planman crt –from 03/21/2011 0900 for 0600
   ```

3. Si nous sommes le 03/21/05 et que la valeur définie pour la variable *startOfDay* dans la base de données est 0600, cette commande crée le plan de production à partir du 03/21/2011 à 6:00 jusqu'au 03/25/2011 à 5:59 :

   ```
   planman crt –to 03/25/2011
   ```

4. Cette commande crée un plan de production à partir du 03/21/2011 à 18:05 jusqu'au 03/24/2011 à 23:00 dans le fuseau horaire Europe\Paris :

   ```
   planman crt –from 03/21/2011 1805 tz Europe\Rome
               –to 03/24/2011 2300 tz Europe\Rome
   ```

Création d'un plan intermédiaire pour une extension de plan

La commande **planman** avec l'option **ext** est appelée à partir de la commande **JnextPlan** lorsque :

- **JnextPlan** est appelé.
- Un plan de production, représenté par le fichier `Symphony` sur le gestionnaire de domaine maître, existe déjà.

Cette commande entraîne la création d'un nouveau plan de production intermédiaire, appelé `Symnew`, qui couvre la durée supplémentaire couverte ultérieurement par le plan de production en cours de génération. La syntaxe utilisée est la suivante :

planman [*paramètres_connexion*] **ext**

{**-to** *mm/jj/[aa]aa[hh[:]mm[**tz** | **fuseau horaire** nom_fh]]* |

-for *[h]hh[:]mm* [**-days** *n*] |

-days *n*}

où :

paramètres_connexion
: Définit les paramètres utilisés lors de l'établissement de la connexion avec HTTP ou HTTPS par WebSphere Application Server vers le gestionnaire de domaine maître. Pour plus d'informations, voir «Ligne de commande Planman», à la page 87.

-to Définit l'heure de fin du plan de production étendu. L'argument **-to** est mutuellement exclusif avec les arguments **-for** et **-days**.

-for Définit la longueur de l'extension du plan de production. Le format est *hhhmm*, où *hhh* représente les heures et *mm* les minutes. L'argument **-for** est mutuellement exclusif avec l'argument **-to**.

-days *n*
Nombre de jours souhaités pour étendre le plan de production. L'argument **-days** est mutuellement exclusif avec l'argument **-to**.

Remarque :

1. Veillez à exécutez la commande **planman** à partir de la commande **JnextPlan**.
2. Le format utilisé pour la date dépend de la valeur attribuée à la variable ***format_date*** dans le fichier `localopts`.
3. Lors de l'extension du plan de production, les chiffres associés aux invites déjà présentes dans le plan sont modifiés.

Si aucun argument **-to**, **-for** ou **-days** n'est spécifié, le plan de production est étendu d'une journée.

Extraction des informations du plan de production

La syntaxe ci-dessous est utilisée pour afficher des informations sur le plan de production courant :

planman [*paramètres_connexion*] **showinfo**

où :

paramètres_connexion
Définit les paramètres utilisés lors de l'établissement de la connexion avec HTTP ou HTTPS par WebSphere Application Server vers le gestionnaire de domaine maître. Pour plus d'informations, voir «Ligne de commande Planman», à la page 87.

Remarque : Vous pouvez installer la fonction Tivoli Workload Scheduler Command Line Client sur les agents tolérants aux pannes et les systèmes extérieurs au réseau Tivoli Workload Scheduler pour émettre à partir de ces systèmes la commande **planman showinfo**.

Le résultat de cette commande affiche :

- Le chemin de l'installation.
- L'heure de début du plan de production.
- L'heure de fin du plan de production.
- La durée du plan de production, après la dernière extension, le cas échéant.
- La date et l'heure de la dernière mise à jour du plan effectuée via **JnextPlan** ou via **planman**.
- L'heure de fin du plan de préproduction.
- L'heure de début de la première instance de flot de travaux non terminée.
- Le *nombre d'exécutions*, c'est à dire le nombre de fois où le plan a été généré.
- Le *nombre d'exécutions confirmées*, c'est à dire le nombre de fois où le plan a été généré correctement.

Les heures de début et de fin des plans de production et de préproduction sont affichées au format spécifié par la variable ***format_date*** définie dans le fichier `localopts` et dans le fuseau horaire de la machine locale.

Voici un exemple de résultat de cette commande :

```
# planman showinfo
Tivoli Workload Scheduler (UNIX)/PLANMAN 8.6 (20100715)
Licensed Materials - Property of IBM*
5698-WSH
(C) Copyright IBM Corp. 1998, 2011 All rights reserved.
* Trademark of International Business Machines
Installed for user "aix61usr".
Locale LANG set to the following: "en"
Plan creation start time: 07/21/2010 06:00 TZ Europe/Rome
Production plan start time of last extension: 07/21/2010 06:00 TZ Europe/Rome
Production plan end time: 07/22/2010 05:59 TZ Europe/Rome
Production plan time extension: 024:00
Plan last update: 07/21/2010 10:05 TZ Europe/Rome
Preproduction plan end time: 08/05/2010 06:00 TZ Europe/Rome
Start time of first not complete preproduction plan job stream instance:
   07/21/2010 10:30 TZ Europe/Rome
Run number: 1
Confirm run number: 1
```

Création d'un plan d'essai

La syntaxe ci-dessous est utilisée pour créer un plan d'essai :

planman [*paramètres_connexion*] **crttrial** *file_name*

[-from *mm/jj/[aa]aa* [*hh[:]mm* [**tz** | **fuseau horaire** *nom_fh*]]]

{**-to** *mm/jj/[aa]aa[hh[:]mm[**tz** | **fuseau horaire** nom_fh]]* |

-for *[h]hh[:]mm* [**-days** *n*] |

-days *n*}

où :

paramètres_connexion
: Définit les paramètres utilisés lors de l'établissement de la connexion avec HTTP ou HTTPS par WebSphere Application Server vers le gestionnaire de domaine maître. Pour plus d'informations, voir «Ligne de commande Planman», à la page 87.

file_name
: Attribue un nom au fichier à créer dans le répertoire *racine_TWS*`/schedTrial` et qui contient le plan d'essai. Le fichier qui contient le plan d'essai s'appelle **T*file_name***. Cela signifie que si la valeur attribuée à *file_name* est `myfile`, le nom du fichier qui contient le plan d'essai généré est `Tmyfile`.

-from
: Définit l'heure de début du plan d'essai.

  Si l'argument **-from** est ignoré, alors :

  - La date par défaut est *aujourd'hui*.
  - L'heure par défaut est la valeur définie dans l'option globale ***startOfDay*** via **optman** sur le gestionnaire de domaine maître.

-to Définit l'heure de fin du plan d'essai. L'argument **-to** est mutuellement exclusif avec les arguments **-for** et **-days**.

-for Définit la durée du plan d'essai. Le format est *hhhmm*, où *hhh* représente les heures et *mm* les minutes. L'argument **-for** est mutuellement exclusif avec l'argument **-to**.

-days *n*
Nombre de jours souhaités pour le plan d'essai. L'argument **-days** est mutuellement exclusif avec l'argument **-to**.

Remarque : Le format utilisé pour la date dépend de la valeur attribuée à la variable ***format_date*** dans le fichier `localopts`.

Si aucun argument **-to**, **-for**, ou **-days** n'est spécifié, la durée du plan d'essai par défaut est de un jour.

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle décrite dans :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Génération de plans d'essai et prévisionnels".

Création d'un plan d'essai avec une extension de plan de production

La syntaxe ci-dessous est utilisée pour créer un plan d'essai avec une extension du plan de production courant :

planman [*paramètres_connexion*] **exttrial** *file_name*

{**-to** *mm/jj/[aa]aa[hh[:]mm[**tz** | **fuseau horaire** nom_fh]]* |

-for *[h]hh[:]mm* [**-days** *n*] |

-days *n*}

où :

paramètres_connexion
Définit les paramètres utilisés lors de l'établissement de la connexion avec HTTP ou HTTPS par WebSphere Application Server vers le gestionnaire de domaine maître. Pour plus d'informations, voir «Ligne de commande Planman», à la page 87.

file_name
Attribue un nom au fichier à créer dans le répertoire *racine_TWS*`/schedTrial` et qui contient le plan d'essai. Le fichier qui contient le plan d'essai s'appelle **T*file_name***. Cela signifie que si la valeur attribuée à *file_name* est `myfile`, le nom du fichier qui contient le plan d'essai généré est `Tmyfile`.

-to Définit l'heure de fin du plan d'essai contenant l'extension du plan de production. L'argument **-to** est mutuellement exclusif avec les arguments **-for** et **-days**.

-for Définit la longueur du plan d'essai contenant l'extension du plan de

production. Le format est *hhhmm*, où *hhh* représente les heures et *mm* les minutes. L'argument **-for** est mutuellement exclusif avec l'argument **-to**.

-days *n*
Définit le nombre de jours souhaités pour le plan d'essai contenant l'extension du plan de production. L'argument **-days** est mutuellement exclusif avec l'argument **-to**.

Remarque : Le format utilisé pour la date dépend de la valeur attribuée à la variable ***format_date*** dans le fichier `localopts`.

Si aucun argument **-to**, **-for**, or **-days**n'est spécifié, par défaut, le plan de production contenu dans le plan d'essai est étendu d'une journée.

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle décrite dans :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Génération de plans d'essai et prévisionnels".

Création d'un plan prévisionnel

La syntaxe ci-dessous est utilisée pour créer un plan prévisionnel :

planman [*paramètres_connexion*] **crtfc** *file_name*

[**-from** *mm/jj/[aa]aa* [*hh[:]mm* [**tz** | **fuseau horaire** *nom_fh*]]]

{**-to** *mm/jj/[aa]aa[hh[:]mm[**tz** | **fuseau horaire** nom_fh]]* |

-for *[h]hh[:]mm* [**-days** *n*] |

-days *n*}

où :

paramètres_connexion
Définit les paramètres utilisés lors de l'établissement de la connexion avec HTTP ou HTTPS par WebSphere Application Server vers le gestionnaire de domaine maître. Pour plus d'informations, voir «Ligne de commande Planman», à la page 87.

file_name
Attribue un nom au fichier à créer dans le répertoire *racine_TWS*/`schedForecast` et qui contient le plan prévisionnel. Le fichier qui contient le plan prévisionnel s'appelle **F*file_name***. Cela signifie que si la valeur attribuée à *file_name* est `myfile`, le nom du fichier qui contient le plan prévisionnel généré est `Fmyfile`.

La longueur maximum de *file_name* peut être de 148 caractères.

-from Définit l'heure de début du plan prévisionnel.

Si l'argument **-from** est ignoré, alors :

- La date par défaut est *aujourd'hui*.
- L'heure par défaut est la valeur définie dans l'option globale ***startOfDay*** via **optman** sur le gestionnaire de domaine maître.

-to Définit l'heure de fin du plan prévisionnel. L'argument **-to** est mutuellement exclusif avec les arguments **-for** et **-days**.

-for Définit la durée du plan prévisionnel. Le format est *hhhmm*, où *hhh* représente les heures et *mm* les minutes. L'argument **-for** est mutuellement exclusif avec l'argument **-to**.

-days *n*
Nombre de jours souhaités pour le plan prévisionnel. L'argument **-days** est mutuellement exclusif avec l'argument **-to**.

Remarque : Le format utilisé pour la date dépend de la valeur attribuée à la variable ***format_date*** dans le fichier `localopts`.

Si aucun argument **-to**, **-for**, ou **-days** n'est spécifié, la durée par défaut du plan prévisionnel est de un jour.

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle décrite dans :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Génération de plans d'essai et prévisionnels".

Déploiement de règles

La commande `planman deploy` est utilisée dans la gestion d'événements. Vous pouvez l'utiliser pour déployer manuellement toutes les règles qui ne sont pas à l'état Brouillon (Draft) (la propriété `isDraft` à la valeur `NO` dans leur définition). La commande fonctionne comme suit :

1. Sélectionne toutes les définitions de règles d'événement qui ne sont pas à l'état Brouillon dans la base de données Tivoli Workload Scheduler.
2. Construit des fichiers de configuration de règles d'événement.
3. Déploie les fichiers de configuration sur les moteurs de surveillance s'exécutant sur les agents Tivoli Workload Scheduler.

Les nouveaux fichiers de configuration mettent à jour les règles d'événement s'exécutant sur chaque moteur de surveillance en termes de :
- Nouvelles règles
- Règles modifiées
- Règles supprimées ou ramenées à l'état `draft`

Vous pouvez utiliser cette commande en complément ou en remplacement de l'option de configuration `deploymentFrequency (df) optman`, qui vérifie périodiquement toute modification apportée aux définitions de règle d'événement à déployer (pour plus d'informations sur cette option, voir le *Guide d'administration*).

Les modifications appliquées aux définitions de règles d'événement dans la base de données prennent effet uniquement une fois le déploiement effectué.

La syntaxe de la commande est :

planman [*paramètres_connexion*] **deploy** [**-scratch**]

où :

paramètres_connexion
Définit les paramètres utilisés lors de l'établissement de la connexion avec HTTP ou HTTPS par WebSphere Application Server vers le gestionnaire de domaine maître. Pour plus d'informations, voir «Ligne de commande Planman», à la page 87.

-scratch
Sans cette option, la commande affecte uniquement les règles qui ont été ajoutées, changées, supprimées ou ramenées à leur état `draft`.

Avec cette option, la commande déploie toutes les règles non draft de la base de données y compris celles qui sont déjà déployées et n'ont pas changé.

Remarque : le temps de déploiement augmente de manière proportionnelle avec le nombre de règles actives à déployer. Si vous avez besoin de déployer manuellement un grand nombre de règles nouvelles ou modifiées, exécutez `planman deploy -scratch` avec cette option pour réduire le temps de déploiement.

L'utilisation de cette option aboutit à une réinitialisation complète du processeur d'événements et doit donc être utilisée avec précaution. La commande peut provoquer la perte de toutes les instances de règle en cours au moment de son émission. Le cas typique est une règle séquentielle qui a été déclenchée et attend des événements additionnels : si vous utilisez l'option à ce moment là l'environnement de règles d'événement est réinitialisé et les éléments suivis sont perdus.

Pour exécuter cette commande, vous devez avoir un accès **build** sur le fichier **prodsked**.

Déverrouillage du plan de production

Lorsque Tivoli Workload Scheduler commence à créer le plan de production, il verrouille les définitions des objets de planification dans la base de données puis les déverrouille soit lorsque la création du plan est terminée, soit si une erreur se produit. Le verrouillage est appliqué pour éviter que les définitions des objets ne soient modifiées lors de la génération ou de l'extension du plan de production. Si le traitement ne se termine pas correctement, les entrées de la base de données risquent de rester verrouillées. Seuls les utilisateurs ayant un droit d'accès **build** au type d'objet fichier **prodsked** défini dans le fichier de sécurité sur le gestionnaire de domaine maître peuvent déverrouiller la base de données. La commande à utiliser est la suivante :

planman [*paramètres_connexion*] **unlock**

où :

paramètres_connexion
Définit les paramètres utilisés lors de l'établissement de la connexion avec HTTP ou HTTPS par WebSphere Application Server vers le gestionnaire de domaine maître. Pour plus d'informations, voir «Ligne de commande Planman», à la page 87.

Remarque : Vous pouvez installer la fonction Tivoli Workload Scheduler Command Line Client sur les agents tolérants aux pannes et les systèmes extérieurs au réseau Tivoli Workload Scheduler pour émettre à partir de ces systèmes la commande **planman unlock**.

Réinitialisation du plan de production

Le script ci-dessous permet de réinitialiser ou de supprimer le plan de production :

ResetPlan [*paramètres_connexion*] [**-scratch**]

où :

paramètres_connexion
Définit les paramètres utilisés lors de l'établissement de la connexion avec HTTP ou HTTPS par WebSphere Application Server vers le gestionnaire de domaine maître. Pour plus d'informations, voir «Ligne de commande Planman», à la page 87.

La différence entre une réinitialisation et une suppression du plan de production est la suivante :

- Si vous *réinitialisez* le plan de production, le plan de préproduction est conservé, mis à jour avec les statistiques des travaux, puis utilisé plus tard pour générer un nouveau plan de production. Par conséquent, lorsque vous créerez le nouveau plan de production, il contiendra toutes les instances de flot de travaux qui n'étaient pas à l'état COMPLETE lors de l'exécution de **ResetPlan**. Lors de la réinitialisation du plan de production, le produit procède comme suit :
  1. Il archive le fichier `Symphony` courant.
  2. Il met à jour les statistiques des travaux.
- Si vous *supprimez* le plan de production, le plan de préproduction est également supprimé. Le plan de préproduction sera recréé sur la base des informations de modélisation stockées dans la base de données lors de la prochaine génération du plan de production. Par conséquent, le nouveau plan de production contiendra toutes les instances de flot de travaux devant être exécutées sur la période couverte par le plan, quel que soit leur état (COMPLETE ou autre) lors de la suppression du plan. Lors de la suppression du plan de production, le produit procède comme suit :
  1. Le fichier `Symphony` actuel est archivé et les données de plan répliquées dans la base de données sont supprimées.
  2. Il met à jour les statistiques des travaux.
  3. Il supprime le plan de préproduction.

  Remarque : Si vous utilisez l'option **-scratch**, veillez à exécuter **dbrunstats** avant le script **JnextPlan**. Pour plus d'informations sur `dbrunstats`, voir *Guide d'administration*.

Lorsque vous exécutez la commande **ResetPlan**, un fichier journal des travaux est créé dans le répertoire *<REP_INST_TWS>*`\TWS\stdlist\`*<DATE>*, où *<REP_INST_TWS>* est le répertoire d'installation de Tivoli Workload Scheduler et *<DATE>* est la date d'exécution du script.

Suppression du plan de préproduction

Le script suivant est utilisé pour supprimer le plan de préproduction lors de la gestion du fichier `Symphony` :

Planman reset -scratch

Lorsque vous exécutez cette commande, le plan de préproduction est supprimé. Le plan de préproduction sera recréé sur la base des informations de modélisation stockées dans la base de données lors de la prochaine génération du plan de

production. Par conséquent, le nouveau plan de production contiendra toutes les instances de flot de travaux devant être exécutées sur la période couverte par le plan, quel que soit leur état (COMPLETE ou autre) lors de la suppression du plan. Lors de la suppression du plan de production, le produit procède comme suit :

1. Il arrête tous les processus Tivoli Workload Scheduler sur le gestionnaire de domaine maître.
2. Le fichier `Symphony` est conservé.
3. Il met à jour les statistiques des travaux.
4. Il supprime le plan de préproduction.

Remarque : Si vous utilisez l'option **-scratch**, veillez à exécuter **dbrunstats** avant le script **JnextPlan**. Pour plus d'informations sur `dbrunstats`, voir le *Guide d'administration*.

Réplication des données de plan dans la base de données

Permet de réaligner les données du plan dans la base de données avec les données du fichier Symphony.

Le stockage d'informations sur des objets du plan dans une base de données rend l'accès aux données de plan beaucoup plus facile et rapide. Lorsque de nombreux utilisateurs accèdent à l'environnement de back end Tivoli Workload Scheduler en même temps, les performances et la fiabilité peuvent être compromises. Le fait de répliquer le plan dans une base de données relationnelle peut rendre l'accès aux données plus rapide et plus fiable.

Pour les versions antérieures à la version 9.1, les opérations suivantes nécessitaient un accès au plan `Symphony` :

- Exécution des rapports de version de référence
- Affichage du plan dans une vue graphique
- Affichage d'une vue Incidence
- Déclenchement des actions
- Actualisation des vues de surveillance des travaux et flots de travaux

Toutes ces activités de modélisation ou de surveillance doivent accéder au plan et, si vous multipliez ces scénarios par le nombre d'utilisateurs qui demandent en même temps un accès au plan pour effectuer une ou plusieurs de ces activités, cela résulte en des temps de réponse et en des performances globales lents.

Pour corriger ce problème, Tivoli Workload Scheduler réplique le plan dans une base de données relationnelle, dans laquelle des instructions SQL permettent de retrouver les données de façon rapide et fiable. Une nouvelle boîte de message, `mirrorbox.msg`, permet de synchroniser la base de données avec le fichier Symphony. Si `mirrorbox.msg` arrive à saturation, par exemple si la base de données reste indisponible pendant une longue période, le plan est automatiquement rechargé dans la base de données à l'aide des informations du fichier Symphony.

En particulier, les performances et les temps de réponse du script UpdateStats ont été considérablement améliorés avec cette nouvelle façon de gérer les données de plan.

En outre, à chaque fois que le plan est étendu, le plan dans la base de données est effacé et recréé, ce qui rend les informations les plus récentes accessibles pour tous

les utilisateurs. Pour répliquer manuellement les données de plan depuis le fichiers Symphony vers la base de données, exécutez la commande **planman resync**.

La synchronisation du fichier Symphony est automatiquement activée lorsque vous ajoutez le fichier Sfinal à la base de données avec la commande composer **add Sfinal**. Un nouveau travail, CHECKSYNC, a été ajouté au flot de travaux FINALPOSTREPORTS contenu dans le fichier Sfinal qui est responsable de la surveillance de l'état du processus de réplication du fichier Symphony dans la base de données. Il envoie le progrès et l'état de ce processus au journal de travail. Si le travail CHECKSYNC doit échouer, reportez-vous au journal du travail CHECKSYNC, ainsi qu'au journal WebSphere Application Server pour déterminer le problème. Après résolution du problème, exécutez la commande **planman resync** pour recharger les données du plan à partir du fichier Symphony dans la base de données.

Important : Si vous mettre à jour votre environnement Tivoli Workload Scheduler vers la version 9.2, vous devez alors implémenter quelques étapes manuelles avant que vos données de plan ne puissent être répliquées dans la base de données. Pour plus d'informations, voir Personnalisation et soumission du flot de travaux final facultatif et «Automatisation du traitement du plan de production», à la page 106.

Pour simplifier les intégrations, un ensemble de vues de la base de données pour un ensemble de tables contenant des données de plan dans la base de données Tivoli Workload Scheduler est fourni.

Pour afficher les vues de base de données contenant des informations sur les objets Tivoli Workload Scheduler dans le plan, reportez-vous aux vues qui commencent par "PLAN_" dans *Tivoli Workload Scheduler - Vues de la base de données*.

Lors de l'exécution des opérations qui récupèrent les données du plan en cours à partir de Dynamic Workload Console, si vous soupçonnez les données de ne pas être à jour, vous pouvez exécuter **planman resync** pour mettre à jour les données du plan dans la base de données avec les informations les plus récentes du fichier Symphony. Si la boîte de message mirrorbox.msg, chargée de synchroniser la base de données avec le fichier Symphony arrive à saturation (lorsque la base de données est par exemple indisponible pour une longue période), **planman resync** est automatiquement émis de sorte que le plan soit entièrement rechargé dans la base de données.

La syntaxe suivante permet de répliquer les données du plan dans la base de données avec les données du fichier Symphony :

planman [*paramètres_connexion*] **resync**

où :

paramètres_connexion
Définit les paramètres utilisés lors de l'établissement de la connexion avec HTTP ou HTTPS par WebSphere Application Server vers le gestionnaire de domaine maître. Pour plus d'informations, voir «Ligne de commande Planman», à la page 87.

Surveillance de la réplication des données de plan dans la base de données

Surveille la progression et l'issue du processus de réplication du fichier Symphony dans la base de données.

Surveille la progression et l'issue du processus de réplication du fichier Symphony dans la base de données. Les messages sont écrits dans la sortie standard avec la progression et le statut de la commande. La commande `planman checksync` est également définie dans le travail `CHECKSYNC`, contenu dans le flot de travaux FINALPOSTREPORTS au sein du fichier Sfinal. Si le travail `CHECKSYNC` doit échouer, alors reportez-vous au journal des travaux du travail `CHECKSYNC` ainsi qu'au journal WebSphere Application Server pour déterminer le problème. Après résolution du problème, exécutez la commande `planman resync` pour recharger les données du plan à partir du fichier Symphony dans la base de données.

La syntaxe suivante permet de surveiller la progression et l'issue de la réplication des données du plan dans la base de données avec les données du fichier Symphony :

planman [*paramètres_connexion*] **checksync**

où :

paramètres_connexion
Définit les paramètres utilisés lors de l'établissement de la connexion avec HTTP ou HTTPS par WebSphere Application Server vers le gestionnaire de domaine maître. Pour plus d'informations, voir «Ligne de commande Planman», à la page 87.

Commande stageman

La commande **stageman** reporte les flots de travaux non terminés, archive l'ancien plan de production et installe le nouveau. Une copie de `Symphony` est envoyée aux gestionnaires de domaine et aux agents dans le cadre du processus d'initialisation du nouveau plan de production. Lors de l'exécution de **JnextPlan**, **stageman** est appelé à partir du script **SwitchPlan**.

Vous devez avoir un droit d'accès ***build*** au fichier `Symphony` pour exécuter **stageman**.

Syntaxe

stageman -V | -U

stageman
[**-carryforward**{**yes** | **no** | **all**}]
[**-log** *fichier_journal* | **-nolog**]
[*symnew*]

Arguments

-V Affiche la version de la commande et quitte l'application.

-U Affiche des informations sur la syntaxe de la commande et quitte l'application.

-carryforward
Définit comment les flots de travaux non terminés sont traités lors du passage au nouveau plan de production. Les valeurs possibles sont les suivantes :

No Aucun flot de travaux n'est reporté.

Oui Seuls les flots de travaux non terminés dont les définitions contiennent le mot clé **carryforward** sont reportés.

all Tous les flots de travaux non terminés sont reportés, quelle que soit la valeur du mot clé **carryforward** dans leur définition.

Si vous omettez ce mot clé, la valeur définie globalement à l'aide de **optman** pour l'option *enCarryForward* lui est attribuée par défaut. Pour plus d'informations sur les paramètres de report utilisés lorsque l'option globale *enCarryForward* et les mots clés **-carryforward** sont tous deux définis, voir «Options de report», à la page 76.

-log Archive l'ancien plan de production dans le répertoire *racine_TWS*/schedlog dans un fichier nommé *fichier_journal*. Les plans de production archivés peuvent alors être répertoriés et sélectionnés (voir «listsym», à la page 427 et «setsym», à la page 440). Si aucun des mots clés **-log** ou **-nolog** n'est défini, Tivoli Workload Scheduler archive l'ancien plan de production selon la convention d'attribution de nom suivante :

```
Maaaammjjhhtt
```

où *aaaammjjhhtt* correspond à l'année, au mois, au jour, à l'heure et à la minute d'archivage de l'ancien plan de production. Si vous générez le plan de production via **JnextPlan**, vous pouvez personnaliser cette convention d'attribution de nom dans le script **SwitchPlan**.

Remarque : Veillez à contrôler l'espace disque du répertoire schedlog et à supprimer les fichiers journaux obsolètes de façon régulière.

-nolog Pas d'archivage de l'ancien plan de production.

symnew
Nom attribué au fichier du plan de production intermédiaire créé par **planman**. Si cette valeur n'est pas indiquée, **stageman** utilise le nom de fichier Symnew.

Commentaires

Pour que les procédures de report fonctionnent correctement en réseau, le fichier du plan de production du gestionnaire de domaine maître, Symphony, doit être mis à jour avec les statuts les plus récents des flots de travaux à partir des agents et des gestionnaires de domaine subordonnés. Exécutez la commande :

```
conman "link @"
```

avant de lancer **stageman**. Cette commande connecte les postes de travail non connectés pour que les messages sur le statut du traitement des travaux mis en file d'attente dans le fichier Mailbox.msg soient renvoyés au gestionnaire de domaine maître afin de mettre à jour le fichier Symphony.

Remarque : Sous UNIX uniquement, **stageman** détermine également les fichiers exécutables associés aux travaux soumis via les commandes d'utilitaire **at** et **batch** qui peuvent être supprimés lorsque Tivoli Workload Scheduler démarre pour la nouvelle période de production. Ces travaux ne sont pas reportés.

Exemples

Procédez au report de tous les flots de travaux non terminés (quel que soit l'état de l'option Carry Forward), consignez l'ancien fichier Symphony et créez le nouveau fichier Symphony :

```
DATE='datecalc today pic AAAAMMJJHHTT'
stageman -carryforward all -log schedlog/M$DATE
```

Procédez au report des flots de travaux non terminés conformément à l'option globale ***carryforward*** ; ne consignez pas l'ancien fichier Symphony mais créez un nouveau plan de production intermédiaire appelé mysym:

```
stageman -nolog mysym
```

Gestion des accès simultanés au fichier Symphony

La présente section décrit deux exemples de scénarios où Tivoli Workload Scheduler gère d'éventuels accès simultanés au fichier Symphony Symphony lors de l'exécution de **stageman**.

Scénario 1 : Accès au fichier Symphony verrouillé par d'autres processus Tivoli Workload Scheduler

Si des processus Tivoli Workload Scheduler sont encore actifs et accèdent au fichier Symphony lors de l'exécution de **stageman**, le message suivant s'affiche :

```
Impossible d'obtenir l'accès exclusif à Symphony.
Arrêt de batchman et de mailman.
```

Pour continuer, arrêtez Tivoli Workload Scheduler et réexécutez **stageman**. Si **stageman** se termine de manière anormale pour une raison quelconque, vous devez réexécuter **planman** et **stageman**.

Scénario 2 : Accès au fichier Symphony verrouillé par stageman

Si vous essayez d'accéder au plan par l'interface de ligne de commande pendant le remplacement du fichier Symphony, le message suivant s'affiche :

```
Le fichier Symphony est obsolète. Remplacement par un nouveau fichier Symphony.
Agenda mm/jj/aaaa (nnnn) sur poste de travail, changement de Symphony.
```

Gestion des dépendances de prédécesseur/successeur avec l'invite de report

Pour assurer la continuité lors du report des flots de travaux, **stageman** génère des invites pour chaque flot de travaux reporté qui a une dépendance prédécesseur/successeur avec un autre flot de travaux non reporté. Ces invites sont émises après le début de la nouvelle période de traitement, lorsque Tivoli Workload Scheduler vérifie si le travail ou le flot de travaux est prêt à être lancé. Les réponses ont la forme d'invites standard. L'exemple qui suit montre une *invite de report*:

```
INACT 1(SYS2#SKED2[(0600 01/11/06),(0AAAAAAAAAAAAA2Y)]) follows
      SYS1#SKED1, satisfied?
```

Cette invite indique qu'un flot de travaux, reporté à partir de l'ancien plan de production, (SYS2#SKED2[(0600 01/11/06),(0AAAAAAAAAAAAA2Y)]), a une dépendance de prédécesseur/successeur avec un flot de travaux nommé SYS1#SKED1 qui n'a pas été reporté. Pour plus d'informations sur la syntaxe permettant d'indiquer le flot de travaux reporté, voir «Sélection de flots de travaux dans les commandes», à la page 388.

L'état de l'invite, **INACT** dans ce cas, définit l'état de la dépendance de `prédécesseur/successeur` correspondante. Les différents états possibles sont les suivants :

INACTIVE
L'invite n'a pas été émise et la dépendance n'est pas satisfaite.

DEMANDEE
L'invite a été émise et attend une réponse. La dépendance n'est pas satisfaite.

NON Une réponse négative a été reçue ou bien le programme a constaté avant que le report ne s'effectue que le flot de travaux (`SKED3`) ne s'est pas achevé avec succès. La dépendance n'est pas satisfaite.

OUI Une réponse positive a été reçue ou bien le programme a constaté avant que le report ne s'effectue que le flot de travaux (`SKED3`) s'est achevé avec succès. La dépendance est satisfaite.

Commande logman

La commande **logman** consigne les statistiques de travaux depuis un fichier journal de plan de production.

Syntaxe

logman -V | -U

logman
[*paramètres_connexion*]
{**-prod** | *fichier-Symphony*]
[**-smooth** *pondération*]
[**-minmax** {*temps_écoulé* | *UC*}]}

Arguments

-U Affiche des informations sur la syntaxe de la commande et quitte l'application.

-V Affiche la version de la commande et quitte l'application.

paramètres_connexion
Représente le jeu de paramètres qui contrôle les interactions entre l'interface du produit, **logman** s'exécutant dans ce cas sur le gestionnaire de domaine maître, et l'infrastructure WebSphere Application Server utilisant HTTP ou HTTPS. Utilisez la syntaxe suivante pour définir les paramètres de connexion :

[**-host** *nom_hôte*] [**-port** *numéro_port*] [**-protocol** *nom_protocole*] [**-proxy** *nom_proxy*] [**-proxyport** *numéro_port_proxy*] [**-password** *mot_de_passe_utilisateur*] [**-timeout** *délai_attente*] [**-username** *nom_utilisateur*]

où :

nom_hôte
Nom d'hôte du gestionnaire de domaine maître.

numéro_port
Numéro de port utilisé lors de la connexion au gestionnaire de domaine maître.

nom_protocole
Protocole utilisé lors de la communication. Il peut s'agir de HTTP avec une authentification standard ou de HTTPS avec une authentification par certificat.

nom_proxy
Nom de l'hôte proxy utilisé pour la connexion.

numéro_port_proxy
Numéro de port proxy utilisé pour la connexion.

mot_de_passe_utilisateur
Mot de passe de l'utilisateur utilisé pour exécuter **logman**.

Remarque : Lorsque vous définissez, sur un poste de travail Windows, un mot de passe contenant deux apostrophes (") ou d'autres caractères spéciaux, assurez-vous qu'une séquence d'échappement est définie pour ces caractères. Si par exemple le mot de passe utilisé est `tws11"tws`, spécifiez-le sous la forme `"tws11\"tws"`.

délai_attente
Délai maximal, en secondes, pendant lequel le programme de ligne de commande de connexion peut attendre la réponse du gestionnaire de domaine maître avant de considérer que la demande de communication a échoué.

nom_utilisateur
Nom de l'utilisateur qui exécute **logman**.

Si un de ces paramètres est oublié lors de l'appel de la commande **logman**, Tivoli Workload Scheduler le recherche d'abord dans le fichier `useropts`, puis dans le fichier `localopts`. Si aucune valeur n'est trouvée pour ce paramètre, une erreur s'affiche. Pour plus d'informations sur les fichiers `useropts` et `localopts`, voir «Configuration des options pour l'utilisation des interfaces utilisateur», à la page 59.

-prod Met à jour le plan de préproduction avec les informations sur les flots de travaux à l'état COMPLETE en production. En procédant ainsi le plan de préproduction est maintenu à jour avec les toutes dernières informations de traitement. Ceci évite le risque que le nouveau plan de production n'exécute à nouveau les flots de travaux déjà complétés dans la période de production précédente.

-minmax {*temps_écoulé* | *UC*}
Définit le mode de journalisation et de consignation des temps d'exécution de travail minimum et maximum. Les valeurs possibles sont les suivantes :

temps_écoulé
Base les temps d'exécution minimum et maximum sur le temps écoulé.

UC Base les temps d'exécution minimum et maximum sur le temps UC.

Cette valeur est utilisée lorsque la commande **logman** est exécutée à partir de la ligne de commande et non à partir du script **JnextPlan**. Lorsque la commande **logman** est exécutée par **JnextPlan**, la valeur utilisée est celle spécifiée dans l'option globale *logmanMinMaxPolicy*.

-smooth *pondération*

Utilise un facteur de pondération qui favorise l'exécution du travail le plus récent lors du calcul du temps d'exécution normal (moyen) pour un travail. Le résultat de ce calcul est exprimé sous la forme d'un pourcentage. Par exemple, **-smooth 40** applique un facteur de pondération de 40% à l'exécution du travail le plus récent et de 60% à la moyenne existante. La valeur par défaut est 0%. Cette valeur est utilisée lorsque la commande **logman** est exécutée à partir de la ligne de commande et non à partir du script **JnextPlan**. Lorsque la commande **logman** est exécutée par **JnextPlan**, la valeur utilisée est celle spécifiée dans l'option globale *logmanSmoothPolicy*.

fichier-Symphony

Le nom d'un fichier Symphony archivé d'où sont extraites des statistiques de travaux.

Commentaires

Les travaux déjà consignés ne peuvent pas être consignés une nouvelle fois. Si vous tentez de le faire, le message d'erreur `0 travaux consignés` est généré.

Exemples

Consignez les statistiques de travaux à partir du fichier journal M199903170935 :

```
logman schedlog/M199903170935
```

Calcul des temps d'exécution moyens

La durée prévue de l'exécution d'un travail est fournie par la commande logman ; il s'agit d'une partie du cycle de planification quotidien. La durée prévue de l'exécution d'un travail est basée sur la moyenne de ses exécutions précédentes. Pour calculer le temps d'exécution moyen d'un travail, la commande logman divise le temps total d'exécution pour toutes les exécutions réussies par le nombre d'exécutions réussies. Si un grand nombre d'exécutions est utilisé pour calculer la moyenne, le changement du temps d'exécution d'un travail n'est pas immédiatement reflété dans la moyenne. Pour répondre plus rapidement à ces changements, vous pouvez utiliser l'option `smooth` afin que la moyenne puisse être pondérée en faveur des exécutions les plus récentes. Utilisez l'option `-smooth` pour saisir un facteur de pondération, tel qu'un pourcentage, pour les exécutions de travaux en cours. Par exemple, la commande `logman -smooth 40` oblige la commande logman à utiliser un facteur de pondération de 40 % pour les exécutions les plus récentes du travail et de 60 % pour la moyenne existante. La commande `logman -smooth 100` oblige les exécutions du travail les plus récentes à remplacer la moyenne existante.

La commande logman conserve les statistiques des exécutions du travail dans la base de données Tivoli Workload Scheduler. Le nombre d'instances de travail conservées dans l'historique des travaux n'est pas limité.

Démarrage du traitement du plan de production

Pour démarrer un cycle de production, procédez comme suit :

1. Ouvrez une session en tant qu'*utilisateur_TWS* sur le gestionnaire de domaine maître.
2. A l'invite, exécutez la commande de script **. ./***racine_TWS***/tws_env.sh** sous UNIX ou ***racine_TWS*\tws_env.cmd** sous Windows pour configurer

l'environnement, puis exécutez le travail JnextPlan en saisissant, par exemple sur un poste de travail UNIX, la commande suivante :

```
JnextPlan.sh -from 05/03/06 0400 -to 06/06/06
```

Elle crée un nouveau plan de production qui démarre le 3 mai 2006 à 04:00 et qui s'arrête le 6 juin 2006 à 03:59. La journée de traitement commencera à l'heure définie sur le gestionnaire de domaine maître par la variable ***startOfDay***.

3. Lorsque le travail JnextPlan est terminé, vérifiez l'état de Tivoli Workload Scheduler :

   ```
   conman status
   ```

   Si Tivoli Workload Scheduler a démarré correctement, le statut est

   ```
   Batchman=ACTIF
   ```

   Si mailman poursuit l'exécution d'un processus sur le poste de travail distant, vous pouvez constater que ce dernier ne s'initialise pas immédiatement. Cela se produit parce que le poste de travail doit terminer toutes les activités en cours impliquant le processus mailman avant de pouvoir être réinitialisé. Une fois que l'intervalle défini dans le paramètre ***mm retry link***, lui-même défini dans le fichier de configuration *racine_TWS*/localopts, s'est écoulé, le gestionnaire de domaine tente à nouveau d'initialiser le poste de travail. Dès que les activités en cours sont terminées, les activités du jour suivant sont initialisées. Pour plus d'informations sur le fichier de configuration localopts, voir *IBM Tivoli Workload Scheduler - Guide d'administration*.

4. Augmentez le nombre maximal de travaux de sorte que les travaux s'exécutent. Par défaut, le nombre maximal de travaux est nul après installation. Cela signifie qu'aucun travail ne s'exécutera.

   ```
   conman "limit;10"
   ```

Automatisation du traitement du plan de production

Si vous étendez votre plan de production à intervalle régulier, par exemple chaque semaine, vous pouvez automatiser cette extension. La présente section explique la procédure à suivre pour cela.

Dans Tivoli Workload Scheduler versions 9.1 et ultérieures, le fichier Sfinal a été modifié pour inclure deux exemples de flot de travaux nommés FINAL et FINALPOSTREPORTS qui vous aident à automatiser la gestion du plan. Une copie de ces flots de travaux se trouvent dans le fichier Sfinal du répertoire *rép_base_TWS*. De plus, une copie des scripts de travaux se trouve également au même emplacement.

Vous pouvez utiliser le fichier Sfinal ou créer votre propre fichier, que vous pourrez personnaliser à votre guise.

Important : Dans tous les cas, pour pouvoir exécuter ces flots de travaux avec succès, *utilisateur_TWS* doit détenir les droits d'accès Write au répertoire temporaire tmp par défaut.

Le flot de travaux FINAL exécute la suite de fichiers scripts décrite dans JnextPlan pour générer le nouveau plan de production. Pour plus de détails, voir «Création et extension d'un plan de production», à la page 84.

Le flot de travaux FINALPOSTREPORTS, chargé de l'impression des rapports de post-production, suit le flot de travaux FINAL et démarre seulement si le dernier

travail répertorié dans le flot de travaux FINAL (SWITCHPLAN) a abouti avec succès. Le flot de travaux FINALPOSTREPORTS comprend aussi un travail nommé CHECKSYNC qui surveille le progrès et le résultat de la commande **planman resync**. La commande **planman resync** charge les données de plan depuis le fichier Symphony vers la base de données.

Dans Tivoli Workload Scheduler versions 9.1 et ultérieures, les données de plan sont désormais complètement répliquées dans la base de données. A chaque fois que le plan est étendu, le plan dans la base de données est recréé, ce qui rend les informations les plus récentes accessibles à tous les utilisateurs. Si vous effectuez une nouvelle installation de Tivoli Workload Scheduler, cette synchronisation du fichier Symphony est alors automatiquement activée lorsque vous ajoutez le fichier Sfinal à la base de données avec la commande composer **add Sfinal**. Etant donné que le fichier Sfinal n'est pas écrasé lors d'une mise à niveau, vous devez ajouter les flots de travaux FINAL et FINALPOSTREPORTS mis à jour à la base de données en exécutant la commande composer add Sfinal. Ceci permet de s'assurer que le travail CHECKSYNC qui est responsable de la réplication des données du plan se trouve dans la base de données. Le fichier Sfinal final mis à jour se trouve alors dans le répertoire TWA_home/config/. Vous devez ensuite exécuter JnextPlan pour inclure les flots de travaux FINAL et FINALPOSTREPORTS dans le plan de production en cours. Voir Personnalisation et soumission du flot de travaux final facultatif.

Par défaut, le flot de travaux FINAL est paramétré pour s'exécuter une fois par jour, suivi par le flot de travaux FINALPOSTREPORTS. Vous pouvez modifier l'heure à laquelle les flots de travaux s'exécutent en modifiant deux paramètres dans la définition du flot de travaux. Pour ce faire (par exemple, pour que les flots de travaux s'exécutent tous les trois jours), procédez comme suit :

- Planifiez le flot de travaux de sorte qu'il s'exécute tous les trois jours en modifiant le cycle d'exécution dans la définition du flot de travaux.
- Dans l'instruction qui appelle **MakePlan** dans le flot de travaux FINAL, indiquez que le plan de production doit durer trois jours en spécifiant **-for 72**.

Vous devez ensuite ajouter les flots de travaux à la base de données en procédant comme suit :

1. Ouvrez une session en tant qu'*utilisateur_TWS*.
2. Exécutez le script **tws_env** pour installer ainsi l'environnement Tivoli Workload Scheduler :
   - UNIX : dans un shell C, lancez `. ./racine_TWS/tws_env.csh`
   - Sous UNIX : dans un shell Korn, lancez `. ./racine_TWS/tws_env.sh`
   - A partir d'une ligne de commande Windows, lancez `rép_base_TWS\tws_env.cmd`

   où *racine_TWS* représente le répertoire d'installation du produit.
3. Ajoutez les définitions des flots de travaux FINAL et FINALPOSTREPORTS à la base de données en exécutant la commande suivante :

   ```
   composer add Sfinal
   ```

   Si vous avez créé un nouveau fichier Sfinal plutôt que d'utiliser celui fourni avec le produit, utilisez le nom du nouveau fichier au lieu du nom Sfinal.
4. Démarrez le cycle de production en exécutant le script **JnextPlan**. Ainsi, les flots de travaux FINAL et FINALPOSTREPORTS seront inclus dans le plan de production en cours.

Remarque : Même si vous décidez d'automatiser l'extension du plan de production, vous pouvez toujours exécuter **JnextPlan** à tout moment.

Chapitre 5. Assurance de service de charge de travail

L'assurance de service de charge de travail est une fonction facultative permettant d'identifier les travaux critiques pour votre activité et de s'assurer qu'ils sont traités à temps. L'utilisation de cette fonction permet à votre personnel de planification des opérations d'améliorer sa capacité à atteindre les niveaux de service définis.

Lorsque la fonction d'assurance de service de charge de travail est activée, vous pouvez identifier les travaux critiques et vérifier qu'un délai de réalisation est spécifié pour la définition ou la soumission de ces travaux. Deux autres unités d'exécution, Time Planner et Plan Monitor, qui s'exécutent dans WebSphere Application Server, sont ensuite lancées pour garantir la réalisation des travaux critiques dans les délais.

La définition d'un travail critique et de son échéance déclenche le calcul des heures de début de tous les autres travaux qui sont des prédécesseurs du travail critique. L'ensemble de prédécesseurs d'un travail critique représente son *réseau critique*. Cela peut inclure des travaux provenant d'autres flots de travaux. A partir de l'échéance et de la durée du travail critique, Time Planner calcule son *heure de début critique*, qui est la dernière heure de début permettant au travail de respecter son échéance. En remontant à partir de l'heure de début critique, il calcule la dernière heure à laquelle chaque prédécesseur dans le réseau critique peut démarrer, de manière à pouvoir réaliser à temps le travail critique à la fin de la chaîne.

Pendant l'exécution du plan, Plan Monitor vérifie en permanence le réseau critique pour s'assurer que l'échéance peut être respectée. Lorsque des modifications ayant un impact sur les délais sont apportées au niveau du réseau critique (l'ajout ou la suppression de travaux ou de dépendances de prédécesseur/successeur, par exemple) Plan Monitor demande à Time Planner de recalculer les heures de début critiques. De même, à la fin d'un travail réseau critique, les délais des travaux lui succédant sont recalculés de façon à prendre en compte la durée réelle du travail.

Dans un réseau critique, l'ensemble de prédécesseurs les plus susceptibles de retarder directement l'heure de début critique est appelé *chemin critique*. Le chemin critique est mis à jour de manière dynamique, à mesure des modifications apportées à la réalisation ou aux risques de réalisation des prédécesseurs.

Le planificateur (batchman) s'exécute automatiquement de façon à corriger les retards en donnant priorité aux travaux représentant un risque réel ou potentiel pour l'échéance cible, bien que certaines conditions à l'origine de retards puissent faire l'objet d'une intervention de l'opérateur. Une série de vues de travaux critiques spécialisés, disponible sur la console Dynamic Workload Console, permet à l'opérateur de rechercher des travaux critiques, d'afficher leurs prédécesseurs et les chemins critiques qui leur sont associés, d'identifier les travaux entraînant des problèmes et d'explorer en aval de façon à identifier et corriger les problèmes.

Pour obtenir des informations détaillées, voir :

- «Activation et configuration de l'assurance de service de charge de travail», à la page 110
- «Planification de travaux critiques», à la page 114
- «Traitement et surveillance des travaux critiques», à la page 115
- «Scénario d'assurance de service de charge de travail», à la page 118

Pour plus d'informations sur l'identification et la résolution des problèmes et sur les problèmes habituels liés à l'assurance de service de charge de travail, voir le chapitre Assurance de service de charge de travail dans *Tivoli Workload Scheduler : Identification et résolution des problèmes*.

Activation et configuration de l'assurance de service de charge de travail

Un certain nombre d'options globales et locales permettent de contrôler la gestion des travaux critiques. Le fichier de sécurité Tivoli Workload Scheduler doit également autoriser les utilisateurs à accéder à tous leurs travaux, flots de travaux et postes de travail associés aux travaux critiques.

Options globales

La fonction d'assurance de service de charge de travail est activée et désactivée par l'option globale `enWorkloadServiceAssurance`. Elle est activée par défaut. D'autres options globales et locales permettent de contrôler les différents aspects du traitement des travaux critiques et de leurs prédécesseurs.

Le tableau 14 présente les options globales utilisées par l'assurance de service de charge de travail. Si vous souhaitez personnaliser les valeurs, modifiez les options globales dans le gestionnaire de domaine maître à l'aide de la ligne de commande **optman**. Dans la plupart des cas, les modifications prennent effet après l'exécution suivante de JnextPlan.

Tableau 14. Options globales de l'assurance de service de charge de travail

Option	Description
enWorkloadServiceAssurance | wa	Active ou désactive le traitement privilégié des travaux critiques et de leurs prédécesseurs. La valeur par défaut est YES. Entrez NO pour la désactiver.
promotionOffset | po	L'assurance de service de charge de travail calcule l'heure de début critique du travail critique lui-même et de chacun de ses prédécesseurs. Il s'agit de la dernière heure à laquelle peut démarrer le travail sans menacer le déroulement opportun du travail critique. Lorsque l'heure de début critique d'un travail approche et que le travail n'a pas démarré, le mécanisme de promotion est utilisé. Des ressources de système d'exploitation sont affectées au travail promu et sa soumission est priorisée. L'option globale `promotionoffset` détermine la longueur de temps avant l'heure de début critique au cours de laquelle le travail devient admissible pour promotion. La valeur par défaut est de 120 secondes.

Tableau 14. Options globales de l'assurance de service de charge de travail (suite)

Option	Description
longDurationThreshold | ld	Le calcul des heures de début critiques des travaux composant un réseau critique repose sur l'échéance définie pour le travail critique et les durées estimées du travail critique et de chacun de ses prédécesseurs. Si le déroulement d'un travail prend plus de temps que prévu, les travaux qui le suivent risquent de manquer leurs heures de début critiques et, par conséquent, de menacer le bon déroulement du travail critique. L'option globale `longDurationThreshold` est une valeur de pourcentage. La valeur par défaut est 150. Grâce à la valeur par défaut, si la durée réelle d'un travail représente 150 % de la durée estimée ou plus, le travail est considéré comme de longue durée. Il est donc ajouté à la liste d'accès direct qui peut s'afficher dans Tivoli Dynamic Workload Console.
approachingLateOffset | al	L'heure de début critique d'un travail du réseau critique est la dernière heure à laquelle le travail peut commencer sans obliger le travail critique à se terminer après l'échéance. Dans la plupart des cas, un travail commence bien avant l'heure de début critique afin que, s'il se prolonge plus longtemps que sa durée estimée, la situation ne devienne pas immédiatement critique. En conséquence, si un travail n'a pas commencé et que l'heure de début critique n'est qu'à quelques minutes, l'achèvement du travail dans les délais est considéré comme potentiellement menacé. L'option `approachingLateOffset` permet de déterminer combien de temps avant l'heure de début critique d'un travail du réseau critique vous devez être informé de ce risque potentiel. Si un travail n'a pas encore démarré dans le nombre spécifié de secondes avant l'heure de début critique, le travail est ajouté à une liste d'accès direct qui peut s'afficher dans la console Dynamic Workload. La valeur par défaut est 120 secondes. **Remarque :** La valeur de ce paramètre est vérifiée régulièrement. Il n'est pas utile d'exécuter JnextPlan pour que les modifications prennent effet.

Tableau 14. Options globales de l'assurance de service de charge de travail (suite)

Option	Description
deadlineOffset | do	En général, une échéance doit être précisée pour un travail marqué `critical`. Dans le cas contraire, le planificateur utilise l'échéance définie pour le flot de travaux. L'option `deadlineOffset` offre un décalage permettant de calculer l'heure de début critique en cas d'absence d'échéance pour le travail critique et le flot de travaux. La fin du plan plus le décalage sont considérés comme l'échéance du travail critique. Le décalage est exprimé en minutes. La valeur par défaut est de 2 minutes. **Important :** Si le plan est étendu, l'heure de début des travaux critiques dont l'échéance est calculée avec ce mécanisme est automatiquement modifiée en raison du fait qu'elle doit à présent concorder avec la nouvelle heure de fin du plan.

Pour de plus amples informations sur les options globales, voir *IBM Tivoli Workload Scheduler - Guide d'administration*.

Options locales

L'assurance de service de charge de travail utilise des options locales afin de contrôler l'allocation de priorité des ressources système aux travaux du réseau critique qui doivent être promus pour maintenir l'échéance critique. Le tableau 15, à la page 113 présente les options locales utilisées par la fonction d'assurance de service de charge de travail. Pour définir les options locales, éditez le fichier *twshome*`\localopts` de chaque poste de travail sur lesquels vont être exécutés les travaux critiques. Exécutez JnextPlan ou redémarrez l'agent pour que les modifications apportées aux options locales prennent effet.

Tableau 15. Options locales de l'assurance de service de charge de travail

Option	Description
jm promoted nice	Définissez la valeur `nice` à attribuer aux travaux critiques ou aux prédécesseurs du travail critique devant être promus sous UNIX et Linux, de manière à les attribuer à plusieurs ressources et à les traiter avant les autres travaux. Les valeurs spécifiques varient selon les plateformes, mais d'une manière générale, le paramètre doit être un entier négatif. La valeur par défaut est -1, des valeurs inférieures représentant des priorités plus élevées. Si vous indiquez un entier positif, la valeur par défaut est utilisée. Le rôle de l'option locale **jm nice** est analogue, donnant la priorité aux travaux ayant été soumis par le superutilisateur. Un travail critique qui a été soumis par le superutilisateur peut être admissible pour les mécanismes de priorisation. Dans ce cas, les valeurs sont ajoutées. Par exemple, si les valeurs -4 et -2 sont respectivement attribuées à **jm promoted nice** et **jm nice**, la priorité du travail critique soumis par le superutilisateur est de -6.
jm promoted priority	Définissez la valeur de priorité des travaux critiques ou des prédécesseurs du travail critique qui doivent être promus, de sorte que le système d'exploitation Windows puisse leur affecter d'autres ressources et les traiter avant les autres travaux. Les valeurs possibles sont : • High • AboveNormal • Normal • BelowNormal • Low ou Idle La valeur par défaut est AboveNormal. Notez que si vous définissez une valeur de priorité inférieure à celle qui peut être attribuée aux travaux non critiques, aucun avertissement n'est envoyé et aucun mécanisme comme celui disponible pour jm promoted nice ne rétablit la valeur par défaut.

Exigences relatives au fichier de sécurité

Les utilisateurs en possession des instances Tivoli Workload Scheduler qui exécutent les travaux critiques doivent obligatoirement être autorisés à utiliser tous les travaux, flots de travaux et postes de travail associés à ces travaux. Ces

utilisateurs doivent par conséquent disposer des droits DISPLAY, MODIFY et LIST, dans leur fichier de sécurité, pour tous les objets associés JOB, SCHEDULE et CPU.

Voir Tivoli Workload Scheduler - *Guide d'administration* pour plus d'informations sur le fichier de sécurité.

Planification de travaux critiques

L'assurance de service de charge de travail permet d'identifier des travaux critiques, définir des échéances et de calculer les délais de l'ensemble des travaux devant précéder le travail critique.

Si un travail doit absolument être terminé avant une heure spécifique, vous pouvez le signaler comme critique lorsque vous l'ajoutez à un flot de travaux à l'aide des fonctions du concepteur de charge de travail sur Dynamic Workload Console. Vous pouvez définir l'échéance au niveau du travail ou du flot de travaux.

Les travaux peuvent également être signalés comme critiques. Pour ce faire, indiquez le mot-clé **critique** lors de la création ou de la modification d'un flot de travaux à l'aide de la ligne de commande **composer**.

Lorsque la commande **JnextPlan** est exécutée afin d'inclure le nouveau travail dans le plan de production, tous les travaux correspondants à des prédécesseurs directs ou indirects du travail critique sont identifiés. Ces travaux, associés au travail critique lui-même, forment un `réseau critique`.

Puisque le délai des travaux du réseau critique doit être étroitement contrôlé, Time Planner calcule les tests de performances des délais suivants pour chaque travail réseau critique :

Démarrage critique
S'applique aux systèmes répartis uniquement et représente la dernière heure à laquelle le travail peut démarrer sans que le travail critique dépasse son échéance.

Les heures de début critiques sont calculées à partir de l'échéance définie pour le travail critique et fonctionnent de façon rétroactive à l'aide de la durée estimée pour chaque travail afin de déterminer sa durée critique. Par exemple, si l'échéance du travail critique s'opère à 19:00 et que la durée estimée du travail critique est de 30 minutes, le travail critique ne sera pas terminé au moment de l'échéance, à moins qu'il n'ait commencé à 18:30. Si le prédécesseur immédiat du travail critique dispose d'une durée estimée de 20 minutes, il doit démarrer au plus tard à 18:10.

Remarque : Seule l'échéance du travail critique est prise en compte lors du calcul des heures de début critiques des travaux dans le réseau critique. Si des échéances ont été définies pour d'autres travaux, leurs heures de début critiques peuvent être ultérieures à leurs échéances.

Premier démarrage
Représente l'heure la plus tôt à laquelle un travail du réseau critique peut démarrer, en prenant en compte toutes les dépendances et les besoins en ressources.

Heures de début et de fin estimées
Les heures de début estimées sont calculées à partir de l'heure la plus tôt à laquelle le premier travail ou les travaux du réseau critique peuvent

démarrer et fonctionnent de façon rétroactive à l'aide de la durée estimée pour chaque travail afin de déterminer l'heure de début du travail suivant.

Heures de début et de fin planifiées
Pour les calculs initiaux, ces valeurs sont définies sur les heures de début et de fin estimées. Elles sont ensuite recalculées afin de prendre en compte toute modification ou retard au niveau du plan.

Durée estimée
La durée estimée d'un travail 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. Prenez ceci en compte lorsque vous considérerez la précision des délais calculés pour les réseaux de travaux critiques incluant les travaux s'exécutant pour la première fois. Dans le cas d'un travail reflet, la durée estimée est toujours définie sur la valeur par défaut d'une minute. Ceci s'applique aux travaux reflets qui s'exécutent pour la première fois, aussi bien que pour toute exécution suivante de travail reflet.

Les délais de chaque travail du réseau critique sont ajoutés au fichier `Symphony`, qui inclut toutes les informations relatives au plan et qui sont réparties sur l'ensemble des postes de travail sur lesquels les travaux doivent être exécutés.

Lors de l'exécution du plan, Plan Monitor examine tous les réseaux critiques : les modifications ultérieures apportées au réseau critique ayant une incidence sur le délai des travaux entraîneront le calcul de l'heure de début estimée et critique. Les modifications peuvent inclure des modifications manuelles, par exemple la publication de dépendances ou la réexécution de travaux, et les modifications automatiquement effectuées par le système en réponse à un risque réel ou potentiel concernant l'achèvement ponctuel d'un travail critique.

Les vues spécifiques des travaux critiques et de leurs prédécesseurs, disponibles dans Dynamic Workload Console, vous permettent de suivre le traitement du réseau critique. Les vues permettent immédiatement d'identifier des problèmes concernant votre planification du travail critique. Par exemple, si l'heure de début estimée d'un travail du réseau critique est postérieure à l'heure de début critique, cela est immédiatement signalé comme risque potentiel au niveau du travail critique.

Traitement et surveillance des travaux critiques

L'assurance de service de charge de travail offre un suivi automatique et permet de prioriser les travaux réseau critiques ainsi que les fonctions en ligne vous permettant de surveiller et d'intervenir au cours du traitement des travaux réseau critiques.

Suivi automatique et priorisation

Pour veiller à ce que les délais critiques soient respectés, l'assurance de service de charge de travail offre les services automatisés suivants aux travaux critiques et aux travaux prédécesseurs formant leurs réseaux critiques :

Promotion
Lorsque l'heure de début critique d'un travail approche et que le travail n'a pas démarré, le mécanisme de promotion est utilisé. Des ressources de système d'exploitation sont affectées au travail promu et sa soumission est priorisée.

Le délai des promotions est contrôlé par l'option globale `promotionoffset`. Les travaux promus sont sélectionnés pour être soumis après des travaux possédant les priorités "Elevée" et "Aller", mais avant tous les autres travaux. La priorisation des ressources du système d'exploitation est contrôlée par les options locales `jm promoted nice` (UNIX et Linux) et `jm promoted priority` (Windows).

Calcul du chemin critique

Le chemin critique correspond à la chaîne des dépendances, menant au travail critique, le plus à même de dépasser l'échéance à un moment donné. Le chemin critique est calculé à l'aide des heures de fin des prédécesseurs du travail critique. Issu du travail critique, le chemin est construit en sélectionnant le prédécesseur avec l'heure de fin estimée la plus récente. Si l'heure de fin réelle diffère de façon substantielle de l'heure de fin estimée, le chemin critique est automatiquement recalculé.

La figure 21 présente le chemin critique via un réseau critique à un moment spécifique de cohérence lors du traitement du plan.

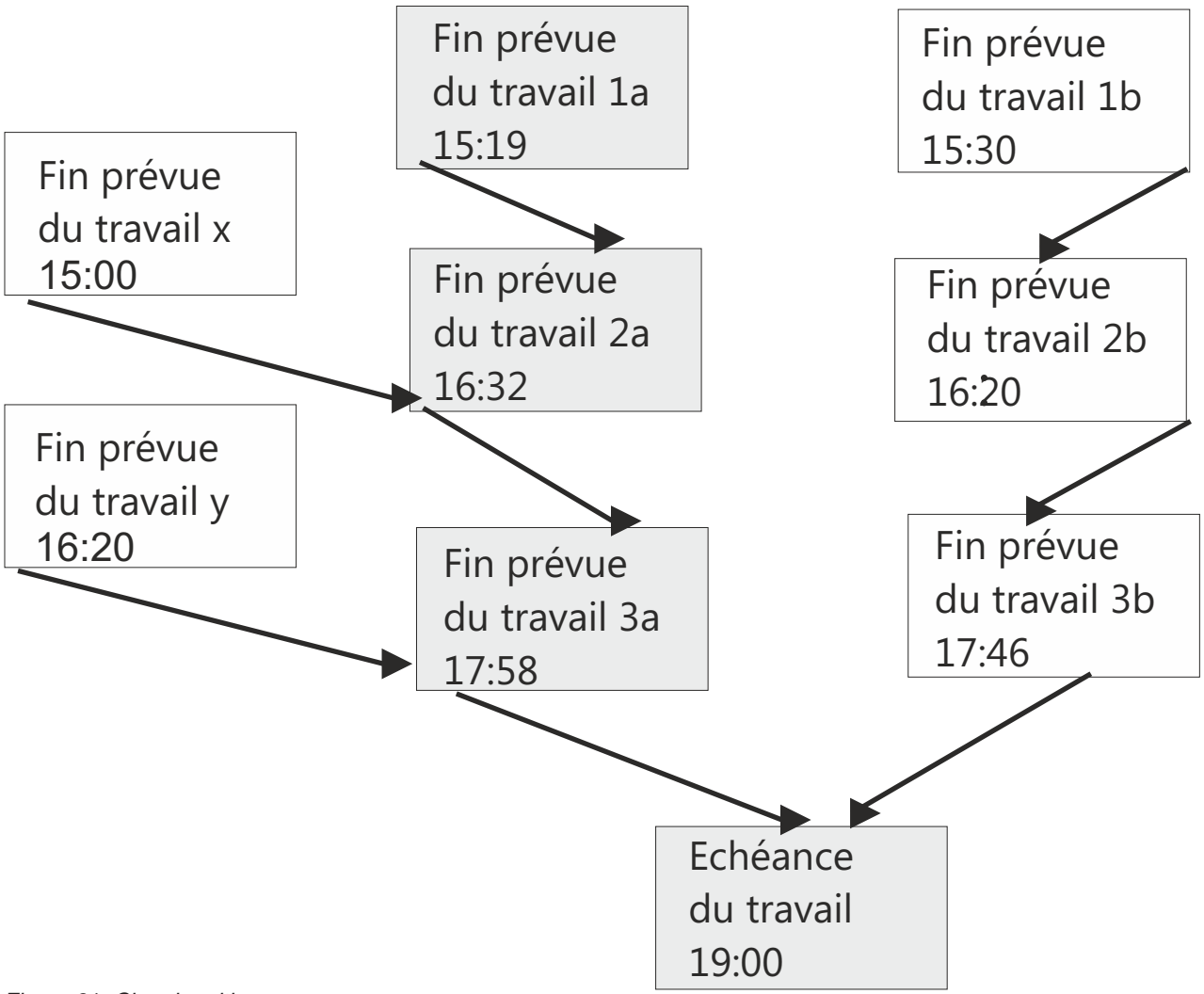


Figure 21. Chemin critique

A ce point de cohérence, le chemin critique inclut Job3a, Job2a et Job1a. Job3a et Job3b correspondent aux prédécesseurs immédiats du travail critique, Job4, et Job3a possède la date de fin estimée la plus récente. Job3a

possède deux prédécesseurs immédiats, Job2a et Job_y. Job2a possède l'heure de fin estimée la plus récente, etc.

Ajout de travaux à la liste d'accès direct

Les travaux inclus dans le réseau critique sont ajoutés à la liste d'accès direct associée au travail critique lui-même. La liste d'accès direct inclut les travaux réseau critiques possédant un impact réel ou potentiel sur l'achèvement ponctuel du travail critique. Les travaux sont ajoutés à la liste d'accès direct pour une ou plusieurs des raisons répertoriées ci-après. Notez que seuls les travaux commençant le réseau critique en cours, pour lesquels il n'y a pas de prédécesseur, peuvent être inclus dans la liste d'accès direct.

- Le travail a été arrêté avec une erreur. La durée comprise avant l'heure de début critique est déterminée par l'option globale `approachingLateOffset`.
- Le travail s'est exécuté pendant une durée supérieure à celle estimée par un facteur défini dans l'option globale `longDurationThreshold`.
- Le travail n'a toujours pas démarré, bien que toutes ses dépendances de prédécesseur ont été résolues ou publiées, et au moins l'une des conditions suivantes est vraie :
  - L'heure de début critique a pratiquement été atteinte.
  - Le travail est programmé pour s'exécuter sur un poste de travail pour lequel la limite est définie sur zéro.
  - Le travail appartient à un flot de travaux pour lequel la limite est définie sur zéro.
  - Le travail ou son flot de travaux a été supprimé.
  - Le travail ou son flot de travaux dispose actuellement d'une priorité inférieure à la priorité minimale ou est défini sur zéro.

Définition d'un statut de risque élevé ou potentiel pour un travail critique

Une statut de risque peut être défini pour le travail critique, comme suit :

Risque élevé
Les délais calculés indiquent que le travail critique se terminera après son échéance.

Risque potentiel
Les travaux prédécesseurs critiques ont été ajoutés à la liste d'accès direct.

Suivi en ligne des travaux critiques

Dynamic Workload Console fournit des vues spécialisées afin de suivre le déroulement des travaux critiques et de leurs prédécesseurs. Vous pouvez accéder aux vues depuis le tableau de bord ou en créant des tâches de surveillance des travaux critiques.

La vue initiale répertorie tous les travaux critiques associés au moteur et affichant l'état : normal, risque potentiel ou risque élevé. Vous pouvez parcourir cette vue et consulter :

- La liste d'accès direct des travaux représentant un risque pour l'échéance critique.
- Le chemin critique.
- Les détails relatifs à l'ensemble des prédécesseurs critiques.
- Les détails relatifs aux prédécesseurs critiques terminés.

- Les fichiers journaux des travaux déjà exécutés.

Ces vues vous permettent de suivre le déroulement du réseau critique, rechercher des problèmes actuels ou potentiels, publier des dépendances ou réexécuter des travaux.

Scénario d'assurance de service de charge de travail

Ce scénario illustre l'utilisation de l'assurance de service de charge de travail, en vérifiant que les échéances de production importantes peuvent être respectées.

Fine Cola utilise Tivoli Workload Scheduler pour gérer les délais et les interdépendances de sa production et de son processus d'approvisionnement.

Fine Cola dispose de contrats de service avec de nombreux clients prenant en charge le réassortiment "en flux tendu", ce qui signifie qu'en cas de retard sur un trajet de livraison, les produits Fine Cola n'arriveront sans doute pas à temps sur les étagères.

Le travail qui génère les commandes de chargement des camions doit être terminé au plus tard à 5h30 du matin. Ce travail dépend de la réussite des autres travaux. Par exemple, même si les commandes sont traitées à l'avance, une modification de dernière minute arrive souvent après le retour de livraison des camions. Fine Cola fournit également des factures avec des notes de livraison. Toute modification de commande doit donc être répercutée dans le tarif et peut être à l'origine d'offres spéciales.

Planification du travail critique

A l'aide du concepteur de charge de travail sur Dynamic Workload Console, le planificateur Fine Cola signale le travail de commande de chargement comme critique et définit l'échéance sur 5h du matin.

Lorsque **JnextPlan** est exécuté, les dates de début critiques de ce travail et de tous ceux qui sont signalés comme ses prédécesseurs sont calculées.

Suivi du travail critique

1. L'opérateur Tivoli Workload Scheduler vérifie les tableaux de bord et note que des travaux critiques sont planifiés sur l'un des moteurs.
2. Il voit qu'un travail critique est en état de risque potentiel. Il clique sur le lien du risque potentiel pour obtenir la liste des travaux critiques dans cet état.

   Le statut du travail de commandes de chargement est défini sur "Risque potentiel".
3. Il sélectionne le travail et clique sur **Accès direct** pour afficher le ou les travaux à l'origine du risque auquel est exposé le travail critique.

   Une erreur est indiquée pour le travail d'ajustement des commandes.
4. Il sélectionne le travail et clique sur **Journal des travaux**.

   Le journal indique que le travail a échoué en raison de données d'identification incorrectes fournies à la base de données des commandes.
5. Lorsqu'il s'aperçoit que le mot de passe de la base de données a été changé ce jour-là, il modifie la définition du travail dans le fichier Symphony, puis réexécute le travail.

6. Lorsqu'il revient au tableau de bord, il voit qu'il n'y a plus de travaux en état de risque potentiel. De plus, la liste des travaux critiques, ouverte en cliquant sur le lien du risque potentiel, n'affiche plus le travail critique une fois qu'il a été réexécuté.
7. Le travail est en cours d'exécution et il a fait l'objet d'une promotion automatique afin d'avoir priorité pour la soumission et les ressources système.
8. Il n'y a pas besoin de résoudre d'autres incidents et le travail critique s'achève à 4h45 du matin.

Chapitre 6. Personnalisation de votre charge de travail à l'aide des tables de variables

Ce chapitre présente le concept de table de variables permettant de regrouper les paramètres globaux, désormais appelés variables, afin de personnaliser votre charge de travail.

Dans Tivoli Workload Scheduler version 8.5, les paramètres globaux auxquels faisaient référence les versions précédentes sont désormais appelés variables. Les définitions de variable sont contenues dans les tables de variables. Une table de variables est un objet regroupant plusieurs variables. Grâce à ces tables, vous pouvez attribuer des valeurs différentes à la même variable et les utiliser dans les définitions de travail et de flot de travaux dans JCL, l'ouverture de session, les dépendances d'invite, les dépendances de fichier et les invites de reprise. Elles sont particulièrement utiles lorsqu'une définition de travail est utilisée en tant que modèle pour un travail appartenant à plusieurs flots de travaux. Par exemple, vous pouvez affecter différentes valeurs à la même variable et réutiliser la même définition de travail dans plusieurs flots de travaux, ce qui vous permet de gagner du temps et de l'argent.

Lorsque vous définissez une variable, vous l'attribuez à une table de variables étant donné que la même variable peut être définie dans différentes tables de variables avec différentes valeurs. Sinon, vous pouvez créer une ou plusieurs tables de variables, en spécifiant une liste de noms et de valeurs de variables pour chaque table. De plus, vous pouvez ajouter le même nom de variable avec différentes valeurs dans différentes tables. Lorsque vous demandez une liste de variables, vous obtenez des paires *table_variables.nom_variable* permettant d'identifier aisément à quelle table de variables appartient la variable.

Par exemple, la variable VAR1 définie dans la table de variables REP1_TABLE1 se présente de la manière suivante :

```
REP1_TABLE1.VAR1
```

Vous pouvez affecter des tables de variables aux cycles d'exécution, aux flots de travaux et aux postes de travail.

Grâce aux tables de variables, vous pouvez modifier le comportement de la charge de travail en fonction du moment, de la raison et de l'endroit où vous souhaitez exécuter votre planification, ce qui donne plus de souplesse quant à la personnalisation de votre charge de travail et satisfait le contrat de service. Plus précisément :

Quand
: Pour modifier le comportement des travaux et des flots de travaux en fonction du moment de la planification de leur exécution, c'est-à-dire du jour de leur exécution. Utilisation des tables de variables avec les cycles d'exécution.

Pourquoi
: Pour modifier le comportement des travaux et des flots de travaux selon la raison qui explique la planification de leur exécution (pour créer un travail qui exécute différentes commandes, par exemple). Utilisation des tables de variables avec des flots de travaux.

Où Pour modifier le comportement des travaux et des flots de travaux selon l'emplacement de leur exécution (sur différents postes de travail, par exemple). Utilisation des tables de variables avec des postes de travail.

Migration des paramètres globaux à partir de versions précédentes

Remarques lors de la migration des paramètres globaux à partir de versions précédentes

Lors de la mise à niveau à partir de versions antérieures à la version 8.5, les définitions de paramètre global (désormais appelées définitions de variable) dont dispose la base de données sont automatiquement migrées vers la table de variables par défaut appelée MAIN_TABLE. Après la mise à niveau :

- Toutes les variables sont précédées par le nom de table par défaut. Par exemple, après la migration, la variable *REP_PATH* porte le nom suivant :

  ```
  MAIN_TABLE.REP_PATH
  ```

  Lorsque vous demandez une liste de variables, vous obtenez des paires *table_variables.nom_variable* permettant d'identifier aisément à quelle table de variables appartient la variable.
- Votre charge de travail est résolue de la même manière qu'avant la migration, puisque tous les objets Tivoli Workload SchedulerTivoli Workload Scheduler contenant des variables font référence à MAIN_TABLE pour la résolution de variable.
- Pour toute section utilisateur incluant le mot clé `parameter`, la ligne suivante est ajoutée dans le fichier de sécurité :

  ```
  vartable name=@ access=add,delete,display,modify,list,use,unlock
  ```

Pour plus de détails sur le processus de mise à niveau, voir Tivoli Workload Scheduler*Tivoli Workload Scheduler - Guide de planification et d'installation*.

Lors de la mise à niveau à partir de la version 8.3 ou ultérieure, ne modifiez pas les variables tant que vous n'avez pas migré le gestionnaire de domaines maîtres et tous ses maîtres de sauvegarde. En effet, lors de cette phase de transition, vous disposez de deux versions différentes de la base de données. Si vous devez ajouter ou modifier des variables au cours de cette phase de transition, vérifiez que les modifications ont été effectuées dans la version 8.3 ou 8.4 et la version 8.5 des gestionnaires de domaine maître.

Le fonctionnement des paramètres locaux qui ont été créés et gérés avec la commande d'utilitaire **parms** dans la base de données de paramètre global sur les postes de travail ne change pas.

Dans la version 8.3 et 8.4, les paramètres sont stockés dans la table MDL.VAR_VARIABLES de la base de données. Après avoir effectué une mise à niveau vers la version 8.5 ou ultérieure, les mêmes paramètres sont stockés dans la MAIN_TABLE qui se trouve dans la table de base de données MDL.VAR_VARIABLES2. Si vous avez commencé à migrer votre environnement, mais n'avez pas terminé la migration, et si la version du gestionnaire de domaine maître est 8.3 ou 8.4 et que celle du gestionnaire de domaine maître de sauvegarde est 8.5 ou ultérieure, ou inversement, le gestionnaire de domaine maître version 8.3 ou 8.4 ne reconnaît pas la table MDL.VAR_VARIABLES2. Ainsi, si vous modifiez un paramètre dans le gestionnaire de domaine maître version 8.3 ou 8.4, cette modification est stockée dans l'ancienne table (MDL.VAR_VARIABLES). Si vous

modifiez le paramètre dans le gestionnaire de domaine maître version 8.5, la modification est stockée dans la table MDL.VAR_VARIABLES2.

Table de variables par défaut

Cette rubrique décrit la table de variables par défaut et son fonctionnement.

La table de variables par défaut est celle contenant toutes les variables que vous avez définies sans préciser un nom de table de variables. Le nom par défaut de la table de variables par défaut est **MAIN_TABLE**. Vous pouvez le modifier à tout moment ou définir une autre table de variables comme table de variables par défaut. Vous ne pouvez pas supprimer la table de variables par défaut. Si vous définissez une autre table de variables par défaut, l'originale n'est plus la table par défaut. Vous pouvez utiliser la table de variables par défaut comme toute autre table de variables. Vous pouvez aisément identifier la table de variables par défaut sur l'interface utilisateur car elle est marquée d'un **Y** dans la zone de **valeur par défaut**.

Exemple

L'exemple ci-dessous présente une liste de tables de variables

```
Variable Table Name            Default Updated On Locked By
-------------------------------------  -------  ----------  ----------------
MAIN_TABLE                 Y    05/07/2008 -
VT_1                         05/07/2008 -
VT_2                         05/07/2008 -
VARTABLE3                      05/07/2008 -
V4                           05/07/2008 -
VT5                           05/07/2008 -

AWSBIA291I Total objects: 6
```

Intégrité des données des tables de variables

Cette rubrique explique comment assurer l'intégrité des données à l'aide des tables de variables.

Comme pour les autres objets, Tivoli Workload Scheduler assure l'intégrité des données de la table de variables à chaque exécution des commandes permettant de créer, de modifier ou de supprimer la définition d'une table de variables.

Lors du référencement d'une table de variable à partir d'un cycle d'exécution, d'un flot de travaux ou d'un poste de travail, Tivoli Workload Scheduler vérifie que la table de variable existe et sauvegarde son lien avec le cycle d'exécution, le flot de travaux et le poste de travail. Cela signifie que vous ne pouvez pas supprimer une définition de table de variables s'il existe une référence à un cycle d'exécution, un flot de travaux ou un poste de travail.

L'intégrité référentielle est assurée au niveau de la table de variables et non à celui de la variable. En d'autres termes, si un cycle d'exécution, un flot de travaux et un poste de travail font référence à une table de variables, Tivoli Workload Scheduler s'assure de l'existence de la table de variables et non de la variable référencée.

Mécanisme de verrouillage des tables de variables

Cette rubrique explique le fonctionnement du mécanisme de verrouillage des tables de variables.

Le verrouillage est défini au niveau de la table de variables afin de s'assurer que les définitions présentes dans la base de données ne sont pas écrasées par différents utilisateurs qui accèdent en même temps à la même table de variables. Cela signifie que lorsque vous verrouillez une table de variables et une variable, vous détenez les droits d'accès exclusifs à toutes les variables de cette table de variables. En d'autres termes, vous pouvez exécuter des commandes sur la table de variables verrouillée et sur les variables qu'elle contient. Les autres utilisateurs ne détiennent qu'un accès en lecture seule à cette table de variables et aux variables qu'elle contient.

Cela empêche tous les autres utilisateurs de modifier la même table de variables que vous. Si un autre utilisateur tente de verrouiller une table de variables ou une variable déjà verrouillée, un message d'erreur s'affiche.

Sécurité de la table de variables

Cette rubrique explique comment définir les paramètres de sécurité des tables de variables.

L'accès utilisateur aux tables de variables doit être autorisé dans le fichier de sécurité Tivoli Workload Scheduler. Comme pour les autres objets, le connecteur vérifie l'existence de l'autorisation avant d'exécuter une action requérant un accès à une table de variables. Le nouveau mot clé est disponible dans le fichier de sécurité dans ce but :

```
vartable name=@ access=add,delete,display,modify,list,use,unlock
```

Vous avez besoin de l'accès `use` pour référencer une table de variables à partir d'autres objets (flots de travaux, cycles d'exécution et postes de travail). Les filtres de sécurité sont uniquement basés sur l'attribut `name`, mais votre administrateur Tivoli Workload Scheduler peut utiliser le mot clé **$default** pour spécifier des autorisations de sécurité dans la table par défaut, quel que soit son nom.

L'autorisation de travailler sur une variable n'est plus basée sur la variable individuelle mais sur la table correspondante. L'accès à une variable est uniquement autorisé si l'action correspondante de la table de variables est autorisée. Le tableau suivant présente les autorisations correspondantes pour les variables et les tables de variables :

Tableau 16. Relation entre les tables de variables et leurs variables correspondantes dans le fichier de sécurité Tivoli Workload Scheduler

Accès défini à vartable	**Action autorisée sur les variables correspondantes**
modify	add
	delete
	modify
display	display
unlock	unlock

Avec la version 8.5, le mot clé `parameter` du fichier de sécurité s'applique aux paramètres locaux uniquement.

Voir *Tivoli Workload Scheduler - Guide d'administration* pour plus d'informations sur le fichier de sécurité.

Résolution de variable

Cette rubrique explique comment les variables sont résolues lors de la génération d'un plan et de la soumission d'un travail et d'un flot de travaux à exécuter.

Le format utilisé pour spécifier une variable peut également déterminer le moment où une variable est résolue avec une valeur. Voir «Définition des variables et des paramètres», à la page 217 pour plus d'informations sur les formats que vous pouvez utiliser.

Lorsque vous générez un plan, Tivoli Workload Scheduler analyse les tables de variables dans l'ordre présenté ci-dessous pour la résolution de variable :
1. Dans le cycle d'exécution. Le cycle d'exécution du flot de travaux est vérifié en premier, puis le cycle d'exécution d'un groupe de cycle d'exécution, et enfin la table de variable définie au niveau du groupe de cycle d'exécution.
2. Dans le flot de travaux.
3. Dans le poste de travail. Voir «Poste de travail considéré pour la résolution de variable».
4. Dans la table de variables par défaut mais uniquement quand les variables sont spécifiées dans le format ^*nomvariable*^ dans la définition du travail. Les variables spécifiées dans le format ${*nomvariable*} ne sont pas résolues.

Au moment de la résolution du plan, chaque niveau est analysé dans l'ordre décrit ci-dessus. Si vous spécifiez une variable qui ne figurent dans aucune table de variables, y compris la table de variables par défaut, un message d'avertissement contenant le nom de la variable non résolue est écrit dans le fichier journal <*chemin_accès_profil_WAS*>\logs\twaserver\SystemOut.log, tandis que le nom de variable est conservé dans le plan, où le chemin d'accès par défaut de<*chemin_accès_profil_WAS*> est <*rép_principal_TWA*>/WAS/TWSprofile.

Lors de la soumission d'un flot de travaux, Tivoli Workload Scheduler résout les variables en analysant les tables de variables dans l'ordre ci-dessous :
1. Spécifiée lors de l'opération de soumission.
2. Dans le flot de travaux.
3. Dans le poste de travail. Voir «Poste de travail considéré pour la résolution de variable».
4. Dans la table de variables par défaut mais uniquement quand les variables sont spécifiées dans le format ^*nomvariable*^ dans la définition du travail. Les variables spécifiées dans le format ${*nomvariable*} ne sont pas résolues.

Lors de la soumission d'un travail, Tivoli Workload Scheduler résout les variables en analysant les tables de variables dans l'ordre ci-dessous :
1. Spécifiées durant l'opération de soumission mais uniquement quand les variables sont spécifiées dans le format ^*nomvariable*^ dans la définition du travail. Les variables spécifiées dans le format ${*nomvariable*} ne sont pas résolues.
2. Dans le poste de travail. Voir «Poste de travail considéré pour la résolution de variable».
3. Dans la table de variables par défaut mais uniquement quand les variables sont spécifiées dans le format ^*nomvariable*^ dans la définition du travail. Les variables spécifiées dans le format ${*nomvariable*} ne sont pas résolues.

Poste de travail considéré pour la résolution de variable

Lorsque la variable est résolue par la table de variables spécifiée dans le poste de travail, le poste de travail pris en compte est :

Pour la variable d'une dépendance de fichier
Celui sur lequel réside le fichier.

Pour la variable d'un travail
Celui sur lequel est défini le travail.

Pour la variable d'une dépendance d'invite

Invite globale
Aucun poste de travail n'est pris en compte. Les variables des invites globales sont toujours résolues à l'aide de la table de variables par défaut. En effet, l'invite globale est utilisée par tous les travaux et flots de travaux de manière à n'utiliser qu'une seule valeur pour la résolution des variables.

Invite Ad hoc
Celui sur lequel est défini le travail ou le flot de travaux qui dépend de la dépendance d'invite.

Chapitre 7. Exécution de l'automatisation de la charge de travail gérée par événements

L'automatisation de la charge de travail gérée par événement ajoute la capacité d'exécuter l'automatisation de la charge de travail à la demande en complément de la planification de travaux basée sur un plan. Elle offre la possibilité de définir des règles pouvant déclencher l'automatisation à la demande de la charge de travail.

L'objet de l'automatisation de la charge de travail gérée par événement dans Tivoli Workload Scheduler est de réaliser un ensemble d'actions prédéfinies en réponse aux événements se produisant sur les noeuds sur lesquels s'exécute Tivoli Workload Scheduler (ainsi que sur les systèmes non liés à Tivoli Workload Scheduler lors de l'utilisation de la ligne de commande `sendevt`). Ceci implique la capacité à soumettre la charge de travail et les commandes d'exécution à la volée, à avertir les utilisateurs par courrier électronique ou à envoyer des messages à Tivoli Enterprise Console.

Les tâches principales de l'automatisation de la charge de travail commandée par les événements sont les suivantes :

- Déclencher l'exécution de travaux et de flots de travaux par lots en fonction de la réception ou de la combinaison d'événements en temps réel.
- répondre à des invites ;
- Avertir les utilisateurs en cas d'anomalies dans l'environnement de planification Tivoli Workload Scheduler ou dans l'activité de planification par lots.
- Appeler un produit externe lorsqu'une condition d'événement particulière se produit.

L'automatisation de la charge de travail gérée par événement est basée sur le concept de la règle d'événement. Dans Tivoli Workload Scheduler, une règle d'événement est un objet de planification qui inclut les éléments suivants :

- Evénements
- Conditions de corrélation des événements
- Actions

Lorsque vous définissez une règle d'événement, vous devez indiquer un ou plusieurs événements, une règle de corrélation et la ou les actions déclenchées par ces événements. En outre, vous pouvez spécifier des dates de validité, un intervalle horaire journalier d'activité, ainsi qu'un fuseau horaire commun à toutes les restrictions de temps définies.

Les événements que Tivoli Workload Scheduler peut détecter pour déclenchement d'actions peuvent être :

Evénements internes
: Il s'agit d'événements impliquant le statut de l'application interne Tivoli Workload Scheduler et les changements de statut des objets Tivoli Workload Scheduler. Les événements de cette catégorie peuvent être les changements de statut des travaux ou flots de travaux, le retard de travaux critiques ou de flots de travaux, ainsi que les changements de statut des postes de travail.

Evénements externes
: Il s'agit d'événements n'impliquant pas directement Tivoli Workload

Scheduler mais qui peuvent néanmoins affecter la soumission de la charge de travail. Les événements de cette catégorie peuvent être des messages inscrits dans les fichiers journaux, des événements envoyés par des applications tierces ou encore un fichier créé, mis à jour ou supprimé.

Il est possible d'établir une corrélation entre deux ou trois événements contenus dans une règle grâce à des attributs de corrélation tels qu'un poste de travail ou un travail commun. Les attributs de corrélation permettent de diriger la règle pour créer une instance de règle séparée (ou une copie de cette règle) pour chaque groupe d'événements partageant des caractéristiques communes. Généralement, chaque règle active a une seule instance qui s'exécute sur le serveur de traitement d'événements. Cependant, il arrive que la même règle soit requise pour différents groupes d'événements, souvent liés à différents groupes de ressources. Utiliser un ou plusieurs attributs de corrélation permet de diriger une règle afin de créer une copie de règle séparée pour chaque groupe d'événements ayant des caractéristiques communes.

Les actions que Tivoli Workload Scheduler peut exécuter lorsqu'il détecte l'un de ces événements peuvent être :

Actions opérationnelles
Il s'agit d'actions qui provoquent le changement de statut des objets de planification. Les actions de cette catégorie sont la soumission d'un travail, d'un flot de travail, une commande ou la réponse à une invite.

Actions de notification
Il s'agit d'actions qui n'ont pas d'impact sur le statut des objets de planification. Les actions appartenant à cette catégorie sont l'envoi d'un courrier électronique, la journalisation d'un événement dans une base de données d'audit interne, la transmission d'un événement à Tivoli Enterprise Console, ou l'exécution d'une commande non Tivoli Workload Scheduler.

Cette classification des événements et des actions est conceptuelle. Elle n'a pas d'impact sur la façon dont ils sont gérés par le mécanisme commandé par les événements.

Scénarios de règles d'événement simples

Cette section répertorie certains scénarios simples impliquant l'utilisation de règles d'événement. Le codage XML correspondant est décrit dans la section «Exemples de règles d'événement», à la page 138.

Scénario 1 : Envoi d'une notification par courrier électronique

1. L'administrateur définit la règle d'événement suivante :
   - Lorsqu'un travail `job123` se termine par une erreur et renvoie le message d'erreur suivant :

     ```
     AWSBHT001E Le travail "MYWORKSTATION#JOBS.JOB1234" dans le fichier "ls" a
     échoué suite à l'erreur : AWSBDW009E L'erreur système suivante
     est survenue en récupérant la structure de mot de passe pour
     l'accès utilisateur...
     ```

     envoi d'un courrier électronique à l'opérateur `john.smith@mycorp.com`. Le sujet du courrier électronique inclut les noms de l'instance de travail et du poste de travail associé.

     La règle d'événement est valide du 1er au 31 décembre dans la fenêtre de temps EST 12:00 à 16:00.

2. L'administrateur enregistre la règle en tant que version non préliminaire dans la base de données et elle est immédiatement déployée par Tivoli Workload Scheduler.
3. Le planificateur commence à surveiller les travaux et à chaque fois que l'un d'eux se termine par une erreur, John Smith reçoit un courrier électronique lui permettant de vérifier le travail et de prendre les mesures appropriées.

Scénario 2 : Surveillance du rétablissement de la liaison d'un poste de travail

1. L'administrateur définit la règle d'événement suivante :
   - Si le poste de travail `CPU1` perd sa liaison et si la liaison n'est pas rétablie dans les 10 minutes, envoi d'une notification par courrier électronique à `chuck.derry@mycorp.com`.
2. L'administrateur enregistre la règle en tant que version non préliminaire dans la base de données et elle est immédiatement déployée par Tivoli Workload Scheduler.
3. Le planificateur commence à surveiller `CPU1`.

   Si le statut du poste de travail passe à unlinked, Tivoli Workload Scheduler fait démarrer le délai d'attente de 10 minutes. Si l'événement d'établissement de liaison `CPU1 linked` n'est pas reçu dans les 10 minutes, le planificateur envoie le courrier électronique de notification à Chuck Derry.
4. Chuck Derry reçoit le courrier électronique, émet une requête pour connaître les actions/règles qui ont été déclenchées dans les 10 dernières minutes et à partir de là, va jusqu'à l'instance `CPU1` et exécute une première identification de problème.

Scénario 3 : Soumission d'un flot de travaux après transmission FTP

1. L'administrateur définit la règle d'événement suivante :
   - o Lorsque le fichier `daytransac*` est créé dans le répertoire `SFoperation` du poste de travail `system1`, et lorsque les modifications apportées au fichier ont été effectuées, soumet le flot de travaux `calmonthlyrev`.

     La règle d'événement est valide toute l'année dans la fenêtre de temps EST 18:00 - 22:00.
2. L'administrateur enregistre la règle en tant que version non préliminaire dans la base de données et elle est immédiatement déployée par Tivoli Workload Scheduler.
3. Le planificateur commence à surveiller le répertoire `SFoperation`. Dès que le fichier `daytransac*` est créé et n'est plus utilisé, il soumet le flot de travaux `calmonthlyrev`.
4. L'opérateur peut vérifier les journaux pour déterminer si la règle d'événement ou le flot de travaux ont été exécutés.

Scénario 4 : Lancer des travaux de longue durée en fonction de délai d'attente

1. L'administrateur définit la règle d'événement suivante :
   - Si les événements `job-x=exec` et `job-x=succ/abend` sont reçus dans les 5 minutes, le planificateur doit répondre `Yes` à `prompt-1` et démarrer le flot de travaux `jobstream-z`, sinon il doit envoyer un courrier électronique à `twsoper@mycompany.com` prévenant que le travail est en retard.

2. L'administrateur enregistre la règle d'événement au statut draft. Après quelques jours, il modifie la règle, change le destinataire du courrier électronique et l'enregistre en tant que version non préliminaire (non-draft). La règle est déployée.
3. A chaque fois que le statut de `job-x` passe à `exec`, Tivoli Workload Scheduler démarre le délai d'attente de 5 minutes.

   Si l'état interne de `job-x` ne passe pas à `succ` ou `abend` dans les 5 minutes et si l'événement correspondant n'est pas reçu, Tivoli Workload Scheduler envoie le courrier électronique, sinon il répond `Yes` à l'invite et soumet `jobstream-z`.

Scénario 5 : Statuts des processus de surveillance et exécution d'un script de lots

L'administrateur crée une règle pour surveiller le statut du processus Tivoli Workload Scheduler et exécute un script de lots.

Scénario 6 : Intégration à SAP R/3 (en combinaison avec Tivoli Workload Scheduler for Applications)

1. L'administrateur définit la règle d'événement suivante :
   - Lorsqu'un événement appelé `ID3965` est généré sur le serveur SAP R/3 `Billing`, Tivoli Workload Scheduler doit :
     a. Exécuter la commande :

        ```
        "/usr/apps/helpDesk –openTicket –text
         'Processing error $parameter
          on SAP system $wsname'"
        ```

        pour ouvrir un ticket de service de maintenance
     b. Envoyer un événement à Tivoli Enterprise Console.
2. L'administrateur enregistre la règle en tant que version non préliminaire et elle est immédiatement déployée.
3. Tivoli Workload Scheduler commence à surveiller le statut des événements SAP R/3 activés sur le système `Billing`.

   Lorsque l'événement `ID3965` est détecté, Tivoli Workload Scheduler exécute la commande de service de maintenance spécifiée et envoie un événement à TEC.
4. Après un certain temps, l'administrateur ramène la règle d'événement au statut draft. La règle est automatiquement désactivée. Elle peut être à nouveau déployée lorsque nécessaire.

Scénario 7 : Surveillance du statut du fichier Symphony et consignation de l'occurrence d'un enregistrement endommagé

L'administrateur crée une règle permettant de surveiller le statut du fichier Symphony dans l'instance de Tivoli Workload Scheduler et consigne l'occurrence d'un enregistrement de dépendance Symphony endommagé dans la base de données d'audit interne.

Processus de gestion des règles d'événements

L'automatisation de la charge de travail gérée par événements est un processus continu et peut se résumer aux étapes ci-dessous :

1. Une définition de règles d'événement est créée ou modifiée avec Dynamic Workload Console ou avec le programme de ligne de commande composer et enregistrée dans la base de données d'objets. Les définitions de règle peuvent être enregistrées selon le statut draft ou non-draft.
2. Toutes les règles nouvelles et modifiées, autres qu'au format brouillon, qui sont enregistrées dans la base de données, sont périodiquement (par défaut toutes

les cinq minutes) détectées, construites et déployées par un processus interne nommé `rule builder`. A ce stade, ils sont activés. Dans l'intervalle, un `serveur de traitement d'événements`, qui est normalement situé dans le gestionnaire de domaine maître, reçoit tous les événements des agents et les traite.

3. Les configurations de surveillance mises à jour sont téléchargées vers les agents Tivoli Workload Scheduler et activées. Chaque agent Tivoli Workload Scheduler exécute un composant nommé `monman`, qui gère deux services intitulés `moteur de surveillance` et `ssmagent`, dont le rôle est d'intercepter les événements se produisant sur l'agent et de leur appliquer une action de filtrage préliminaire.
4. Chaque instance de `monman` détecte et envoie ces événements au serveur de traitement d'événements.
5. Le serveur de traitement d'événements reçoit les événements et vérifie leur concordance avec toute règle d'événement déployée.
6. Si une règle d'événement concorde, le serveur de traitement d'événements appelle un auxiliaire d'actions pour effectuer les actions.
7. Celui-ci crée une instance de règles d'événement et journalise la sortie de l'action dans la base de données.
8. L'administrateur ou l'opérateur vérifie le statut des instances de règles d'événement et des actions dans les journaux et dans la base de données.

La fonction d'automatisation de la charge de travail gérée par événements est installée automatiquement avec le produit. Vous pouvez, à tout moment, changer la valeur de l'option globale `enEventDrivenWorkloadAutomation` si vous ne souhaitez pas ajouter cette fonctionnalité à votre réseau Tivoli Workload Scheduler.

L'automatisation de la charge de travail gérée par événements est basée sur un grand nombre de services, sous-systèmes et mécanismes internes. Les éléments suivants sont déterminants car ils peuvent être gérés :

monman
Installé sur chaque agent Tivoli Workload Scheduler où il vérifie tous les événements locaux. Tous les événements détectés sont transmis au serveur de traitement d'événements. Les commandes `conman` suivantes sont disponibles pour gérer `monman`:

Tableau 17. Commandes conman pour gérer les moteurs de surveillance

Commande	Action
deployconf	Met à jour le fichier de configuration pour le moteur de surveillance d'événements sur un agent. Cette commande est optionnelle, puisque la configuration est normalement déployée de façon automatique.
showcpus getmon	Renvoie la liste des règles d'événement définies pour le moniteur s'exécutant sur un agent. Cette commande peut être exécutée à distance pour extraire les informations du fichier de configuration situé sur un autre agent du réseau.
startmon	Démarre `monman` sur un agent. Cette commande peut être exécutée à partir d'un autre agent.
stopmon	Arrête `monman` sur un agent. Cette commande peut être exécutée à partir d'un autre agent.

`monman` démarre automatiquement à chaque fois qu'un nouveau fichier Symphony est activé. Ce comportement est déterminé par la définition de

l'option locale autostart monman sur la valeur yes par défaut. Vous pouvez désactiver cette fonction si vous ne souhaitez pas surveiller les événements d'un agent particulier.

A la suite de chaque cycle de déploiement de règles, les configurations de surveillance mises à jour sont automatiquement réparties vers les agents qui hébergent des règles ayant subi des modifications depuis le déploiement précédent. Il est à noter que certaines situations provisoires peuvent survenir durant le déploiement. Si une règle se trouve par exemple en attente de désactivation, il se peut que les agents envoient des événements dans la fraction d'intervalle durant laquelle les nouveaux fichiers de configuration n'ont pas encore été déployés, alors que le processeur d'événements a déjà ignoré ces événements.

Si un agent ne parvient pas à envoyer des événements à l'event processing server durant une période de temps spécifiée, le statut de surveillance de l'agent est automatiquement désactivé. La période de temps peut être personnalisée (en secondes) avec le paramètre edwa connection timeout du fichier localopts. Par défaut, ce paramètre est défini à 300 secondes (5 minutes).

Les événements suivants peuvent être configurés dans le fichier BMEvents.conf afin d'envoyer les états de surveillance d'un agent :

- TWS_Stop_Monitoring (261) : envoyé lorsque l'état de surveillance d'un agent est désactivé (à cause d'une commande stopmon ou parce que l'agent ne parvient pas à envoyer d'événements à l'event processing server).
- TWS_Start_Monitoring (262): envoyé lorsque l'état de surveillance d'un agent est activé (grâce à la commande startmon ou parce que l'agent a recommencé à envoyer des événements à l'event processing server).

Ces événements ont les zones positionnels suivantes :

1. Numéro d'événement
2. Poste de travail concerné
3. Réservé, actuellement toujours réglé sur 1

Serveur de traitement d'événements

Peut être installé sur le gestionnaire de domaine maître, le maître de sauvegarde ou tout agent tolérant aux pannes installé en tant que maître de sauvegarde. L'exécution a lieu sur le serveur d'applications. Le serveur peut être actif sur un seul noeud du réseau. Il crée les règles, génère les fichiers de configuration des agents et signale le téléchargement de nouvelles configurations aux agents. Il assure la réception et la corrélation des événements envoyés par les moteurs de surveillance et exécute les actions. Les commandes conman suivantes sont disponibles pour gérer le serveur de traitement d'événements :

Tableau 18. Commandes conman pour gérer le serveur de traitement d'événements

Commande	Action
starteventprocessor	Démarre le serveur de traitement d'événements.
stopeventprocessor	Arrête le serveur de traitement d'événements.
switcheventprocessor	Bascule le serveur de traitement d'événements du gestionnaire de domaine maître vers le maître de sauvegarde ou l'agent tolérant aux pannes installé comme maître de sauvegarde, ou inversement

Le serveur de traitement d'événements démarre automatiquement avec le gestionnaire de domaine maître. Un seul processeur d'événement peut s'exécuter à tout moment dans le réseau. Si vous souhaitez exécuter le processeur d'événement installé sur un poste de travail différent du maître (sur le maître de sauvegarde ou sur un agent tolérant aux pannes installé comme tel), vous devez en premier lieu utiliser la commande switcheventprocessor pour en faire un serveur de traitement d'événement actif.

Remarque : Si vous définissez le mot clé **ignore** sur la définition de poste de travail de l'agent (installé comme maître de sauvegarde) qui, à cet instant, héberge le processeur d'événement actif, la première occurrence de **JnextPlan** suivante reconnaît que cet agent particulier n'entre pas dans le cadre du plan. En conséquence, elle ne peut pas redémarrer le processeur d'événement hébergé. C'est la raison pour laquelle le planificateur renvoie un message d'avertissement et démarre le processeur d'événement hébergé par le gestionnaire de domaine maître.

Utilisation des interfaces et commandes impliquées

L'exécution et la gestion de l'automatisation de la charge de travail gérée par événements nécessite les tâches suivantes :

- Modifier les paramètres de configuration
- Modéliser les règles d'événement
- Déployer manuellement ou annuler le déploiement de règles d'événement
- Gérer les systèmes de surveillance et de traitement d'événements
- Surveiller et gérer les instances de règles d'événement

Vous devez être prêt à utiliser plusieurs interfaces et commandes Tivoli Workload Scheduler pour exécuter ces tâches. Le tableau 19, à la page 134 résume celles dont vous avez besoins :

Tableau 19. Interfaces et commandes pour gérer l'automatisation de charge de travail gérée par événements

Interface ou commande	Utilisée pour...
optman	change les valeurs par défaut des options globales associées à la gestion d'événements. Les options globales sont utilisées pour configurer : • Fréquence à laquelle les définitions de règles sont vérifiées pour détecter les mises à jour (`deploymentFrequency`). Les définitions modifiées sont déployées dans le domaine Tivoli Workload Scheduler • Numéro de port EIF sur lequel le serveur de traitement d'événements reçoit les événements (`eventProcessorEIFPort`, ou `eventProcessorEIFSSLPort` en cas de protection SSL). • Gestion des stratégies de nettoyage des données liées aux instances de règles, exécutions d'actions et consignations de messages (`logCleanupFrequency`). • Propriétés du serveur SMTP si vous déployez des règles mettant en oeuvre des actions qui envoient des courriers électroniques via un serveur SMTP (`smtpServerName`, `smtpServerPort`, `smtpUseAuthentication`, `smtpUserName`, `smtpUserPassword`, `smtpUseSSL`, `smtpUseTLS`). • Propriétés du serveur Tivoli Enterprise Console si vous déployez des règles mettant en oeuvre des actions qui transmettent des événements à la console TEC (`TECServerName`, `TECServerPort`). • Possibilité de désactiver le mécanisme de gestion des règles d'événement (`enEventDrivenWorkloadAutomation`), qui est installé par défaut avec le produit. Voir le *Guide d'administration* pour la liste des options globales.
composer	Exécute la modélisation et les tâches de gestion des définitions de règles d'événement telles que add, create, delete, display, extract, list, lock, modify, new, print, unlock, validate. Les règles d'événement sont définies au format XML. émettent des requêtes vers la base de données relationnelle Tivoli Workload Scheduler pour : • les définitions de règles d'événement filtrées par : – propriétés de règle, d'événement et d'action – travaux et flots de travaux impliqués par l'action associée à la règle • instances de règles d'événement, actions exécutées et enregistrement du journal des messages Voir «Définition de règle d'événement», à la page 284 pour la définition des règles d'événement. Voir Chapitre 9, «Gestion des objets dans la base de données - composer», à la page 301 pour les références des commandes.

Tableau 19. Interfaces et commandes pour gérer l'automatisation de charge de travail gérée par événements (suite)

Interface ou commande	Utilisée pour...
Dynamic Workload Console	a une interface utilisateur graphique pour : • Modéliser et gérer les définitions de règles d'événement (browse, create, delete, modify, query, unlock) • émettent des requêtes vers la base de données relationnelle Tivoli Workload Scheduler pour : – les définitions de règles d'événement filtrées par : - propriétés de règle, d'événement et d'action - travaux et flots de travaux impliqués par l'action associée à la règle – instances de règles d'événement, actions exécutées et enregistrement du journal des messages • Gérer le serveur de traitement d'événements et les moteurs de surveillance tels que décrit dans le tableau 17, à la page 131 et le tableau 18, à la page 132 Voir la documentation Dynamic Workload Console : Dynamic Workload Console - Guide d'utilisation, section "Création d'une règle d'événement". Dynamic Workload Console - Guide d'utilisation, section "Tâches de gestion des événements".
conman	Gère les systèmes de surveillance, à savoir le serveur de traitement d'événements et les moteurs de surveillance tels que décrits dans le tableau 17, à la page 131 et le tableau 18, à la page 132. Voir Chapitre 11, «Gestion des objets dans le plan - conman», à la page 373 pour les références des commandes.
commandes d'utilitaire	Création de définitions d'événements personnalisées et envoi manuel de ces événements au serveur de traitement d'événements. Voir «evtdef», à la page 553 et «sendevent», à la page 573 pour plus de détails sur ces commandes.
planman	Déploiement manuel des règles nouvelles et modifiées. Pour plus d'informations, voir «Déploiement de règles», à la page 95.
fichier de sécurité	Définition des autorisations de sécurité pour gérer les règles d'événement, les événements, les actions et leurs instances. Voir *Tivoli Workload Scheduler - Guide d'administration* pour plus d'informations sur la configuration du fichier de sécurité Tivoli Workload Scheduler.

Important : Si vous utilisez un pare-feu de sécurité, assurez-vous que les ports définis dans l'option globale `eventProcessorEIFPort` et dans l'option locale `nm port` sur chaque agent sont ouverts pour les connexions entrantes et sortantes.

Définition de règles d'événement

Lorsque vous définissez une règle d'événement, vous spécifiez un ou plusieurs événements, une règle de corrélation et une ou plusieurs actions. Pour définir les règles d'événement, vous pouvez utiliser :
- La ligne de commande de `composer`
- L'Dynamic Workload Console
- Un ensemble d'interfaces API décrites dans un document séparé

Dans `composer`, vous modifiez les règles avec un éditeur XML de votre choix (option préférable mais facultative), car les définitions de règles d'événement sont créées avec la syntaxe XML.

La méthode d'utilisation de `composer` pour définir les règles d'événement est expliquée dans «Définition de règle d'événement», à la page 284, alors que la méthode d'utilisation de Dynamic Workload Console est décrite dans : Dynamic Workload Console - Guide d'utilisation, section Création d'une règle d'événement.

Les définitions de règles d'événement sont enregistrées dans la base de données Tivoli Workload Scheduler comme tous les autres objets de planification. Les options de sauvegarde disponibles sont les suivantes :

Brouillon
La règle est enregistrée dans la base de données, mais n'est pas prête à être déployée ou activée.

Cet état est déterminé par l'attribut `isDraft=yes`.

Version non préliminaire
Cette règle est déployée ou est prête à être déployée dans l'environnement de planification.

Cet état est déterminé par l'attribut `isDraft=no`.

Les règles de version non préliminaire sont prêtes à être activées. Le générateur de règles évalue le statut de chaque règle. L'état peut être :
- actif
- inactif
- mise à jour en attente
- erreur de mise à jour
- activation en attente
- erreur d'activation
- désactivation en attente
- erreur de désactivation

Périodiquement (toutes les cinq minutes ou conformément à une heure définie via l'option de configuration globale `deploymentFrequency`), le planificateur analyse la base de données pour détecter les règles de version non préliminaire, puis crée les fichiers de configuration de règles pour le déploiement. Les nouvelles configurations de surveillance ne sont téléchargées sur les agents (chaque agent obtient son propre fichier de configuration contenant exclusivement les règles qu'il est censé exécuter) que si des modifications ont été apportées par rapport aux fichiers de configuration précédents.

La commande complémentaire `planman deploy` permet à tout moment de déployer manuellement les règles actives.

Le temps requis par le déploiement augmente de manière proportionnelle avec le nombre de règles actives à déployer. Si vous avez besoin de déployer

manuellement un grand nombre de règles nouvelles ou modifiées et que vous souhaitez réduire le temps de déploiement, exécutez `planman deploy -scratch`.

Vous pouvez déployer ou annuler le déploiement de règles selon vos besoins en définissant l'attribut `isDraft` sur `no` ou sur `yes` avec `composer` ou avec Dynamic Workload Console.

En fonction de leurs caractéristiques, les règles sont classifiées comme suit :

filter La règle est activée sur détection d'un événement spécifique unique.

sequence
La règle est activée lorsqu'un groupe ordonné d'événements est détecté ou intervient dans un intervalle de temps spécifique.

set La règle est activée lorsqu'un groupe non ordonné d'événements est détecté ou intervient dans un intervalle de temps spécifique.

Les règles de filtrage reposent sur la détection d'un événement unique tel que le retard d'un travail, l'arrêt d'un poste de travail Tivoli Workload Scheduler, la modification d'un fichier, l'achèvement avec succès d'un flot de travaux, etc.

Les règles 'Set' et 'Sequence' reposent sur la détection d'un plus grand nombre d'événements. Ces règles peuvent également, en variante, dépendre d'une condition de dépassement de délai d'attente. Le dépassement du délai d'attente d'une règle a lieu lorsque le ou les premiers événements d'une règle Sequence, ou une partie des événements d'une règle Set sont reçus, mais que tous les événements ne sont pas reçus dans un intervalle de temps spécifié, lequel est calculé à partir de la réception du dernier événement.

Les définitions de règles peuvent inclure des attributs qui spécifient une période de validité et une fenêtre de temps d'activité pour chaque jour de validité. Si vous ne spécifiez pas ces attributs, la règle est active de façon perpétuelle à tout moment une fois déployée et jusqu'à ce qu'elle soit ramenée au statut draft ou supprimée de la base de données.

Vous pouvez utiliser la substitution de variable. Lorsque vous définissez des paramètres d'action, vous pouvez utiliser les attributs d'occurrences d'événements qui déclenchent la règle d'événement selon l'une des deux formes suivantes :

- `${event.property}`

  Remplace la valeur telle quelle. Cette option est utile pour transmettre les informations à une action liée à celles-ci par voie de programme, par exemple l'option `schedTime` d'un flot de travaux.

- `%{événement.propriété}`

  Remplace la valeur formatée en syntaxe lisible. Cette option est utile pour transmettre les informations à une action chargée de faire suivre les informations à un utilisateur, par exemple, pour formater le `schedTime` d'un flot de travaux dans le corps d'un courrier électronique.

où :

event est le nom défini pour le déclenchement `eventCondition`.

property
est le nom de `attributeFilter` dans le prédicat de filtrage de la condition d'événements déclenchante. La valeur prise par le filtre d'attribut lorsque la règle est déclenchée est remplacée en tant que valeur de paramètre dans la définition d'action avant l'exécution.

Il est à noter que vous pouvez également recourir à une substitution de variable si aucune valeur de `attributeFilter` n'a été spécifiée pour un attribut, ainsi que lorsque l'attribut est accessible en lecture seule.

Vous pouvez voir l'utilisation de la substitution de variables dans certains des exemples de définitions suivants où les valeurs de filtre d'attributs sont remplacées dans l'objet et le corps d'un courrier électronique.

Exemples de règles d'événement

Vous trouverez ci-dessous des exemples de définitions de règles d'événement qui s'appliquent aux scénarios décrits dans «Scénarios de règles d'événement simples», à la page 128.

Définition de règles d'événement pour le scénario 1

Lorsqu'un travail `job123` se termine par une erreur et renvoie le message d'erreur suivant :

```
AWSBHT001E The job "MYWORKSTATION#JOBS.JOB1234" in file "ls" has failed with
the error: AWSBDW009E The following operating system error occurred retrieving
the password structure for either the logon user...
```

envoi d'un courrier électronique à l'opérateur `john.smith@mycorp.com`. Le sujet du courrier électronique inclut les noms de l'instance de travail et du poste de travail associé.

La règle d'événement est valide du 1er au 31 décembre dans la fenêtre de temps EST 12:00 à 16:00.

```
<?xml version="1.0"?>
<eventRuleSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns="http://www.ibm.com/xmlns/prod/tws/1.0/event-management/rules"
      xsi:schemaLocation="http://www.ibm.com/xmlns/prod/tws/1.0/
                          event-management/rules/EventRules.xsd">
   <eventRule name="scenario1_rule" ruleType="filter" isDraft="no">
      <description>This is the definition for scenario1</description>
      <timeZone>America/Indianapolis</timeZone>
      <validity from="2010-12-01" to="2010-12-31" />
      <activeTime start="12:00:00" end="16:00:00" />
      <eventCondition name="event1"
                      eventProvider="TWSObjectsMonitor"
                      eventType="JobStatusChanged">
         <filteringPredicate>
            <attributeFilter name="JobStreamWorkstation" operator="eq">
               <value>*</value>
            </attributeFilter>
            <attributeFilter name="JobStreamName" operator="eq">
               <value>*</value>
            </attributeFilter>
            <attributeFilter name="JobName" operator="eq">
               <value>job123*</value>
            </attributeFilter>
            <attributeFilter name="Status" operator="eq">
               <value>Error</value>
            </attributeFilter>
            <attributeFilter name="ErrorMessage" operator="eq">
               <value>*AWSBDW009E*</value>
            </attributeFilter>
         </filteringPredicate>
      </eventCondition>
      <action actionProvider="MailSender"
              actionType="SendMail"
              responseType="onDetection">
```

```
         <description>Send email to John Smith including names of job
                      and associated workstation</description>
         <parameter name="To">
            <value>john.smith@mycorp.com</value>
         </parameter>
         <parameter name="Subject">
            <value>Job %{event1.JobName} on agent %{event1.Workstation} ended in error</value>
         </parameter>
      </action>
   </eventRule>
</eventRuleSet>
```

Important : Le message d'erreur expliquant pourquoi un travail s'est terminé par une erreur peut figurer dans le fichier journal TWSMERGE. Dans ce scénario, le fichier journal TWSMERGE contient l'instruction suivante :

```
BATCHMAN:+
BATCHMAN:+ AWSBHT001E The job "MYWORKSTATION#JOBS.JOB1234" in file "ls"
has failed with the error: AWSBDW009E The following operating system
error occurred retrieving the password structure for either the logon
user, or the user who owns a file or external dependency
BATCHMAN:+
```

où le message d'erreur est constitué de tout ce qui suit la chaîne :

```
has failed with the error:
```

Définition de règles d'événement pour le scénario 2

Si le poste de travail `CPU1` perd sa liaison et si la liaison n'est pas rétablie dans 1 heure, envoi d'une notification par courrier électronique à `chuck.derry@mycorp.com`.

```
<?xml version="1.0"?>
<eventRuleSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns="http://www.ibm.com/xmlns/prod/tws/1.0/event-management/rules"
      xsi:schemaLocation="http://www.ibm.com/xmlns/prod/tws/1.0/
                          event-management/rules/EventRules.xsd">
   <eventRule name="scenario2_rule" ruleType="filter" isDraft="no">
      <description>This is the definition for scenario2</description>
      <timeZone>America/Anchorage</timeZone>
      <timeInterval amount="1" unit="hours" />
      <eventCondition name="WSevent"
                      eventProvider="TWSObjectsMonitor"
                      eventType="ChildWorkstationLinkChanged">
         <filteringPredicate>
            <attributeFilter name="Workstation" operator="eq">
               <value>CPU1</value>
            </attributeFilter>
            <attributeFilter name="LinkStatus" operator="eq">
               <value>Unlinked</value>
            </attributeFilter>
         </filteringPredicate>
      </eventCondition>
      <action actionProvider="MailSender"
              actionType="SendMail"
              responseType="onDetection">
         <description>Send email to Chuck Derry with name of
                      unlinked workstation</description>
         <parameter name="To">
             <value>chuck.derry@mycorp.com</value>
         </parameter>
         <parameter name="Subject">
             <value>Agent CPU1 has been unlinked for at least 10 minutes</value>
         </parameter>
         <parameter name="Body">
             <value>The cause seems to be: %{WSevent.UnlinkReason}</value>
```

```
        </parameter>
      </action>
   </eventRule>
</eventRuleSet>
```

Définition de règles d'événement pour le scénario 3

Lorsque le fichier daytransac est créé dans le répertoire SFoperation du poste de travail system1, et lorsque les modifications apportées au fichier ont été effectuées, soumet le flot de travaux calmonthlyrev.

La règle d'événement est valide toute l'année dans la fenêtre de temps EST 18:00 - 22:00.

```
<?xml version="1.0"?>
<eventRuleSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns="http://www.ibm.com/xmlns/prod/tws/1.0/event-management/rules"
      xsi:schemaLocation="http://www.ibm.com/xmlns/prod/tws/1.0/
                          event-management/rules/EventRules.xsd">
   <eventRule name="scenario3_rule"
              ruleType="filter"
              isDraft="no">
      <description>This is the definition for scenario3</description>
      <timeZone>America/Louisville</timeZone>
      <validity from="2007-01-01" to="2007-12-31" />
      <activeTime start="18:00:00" end="22:00:00" />
      <eventCondition eventProvider="FileMonitor"
                      eventType="ModificationCompleted">
         <filteringPredicate>
            <attributeFilter name="FileName" operator="eq">
               <value>daytransac</value>
            </attributeFilter>
            <attributeFilter name="Workstation" operator="eq">
               <value>EVIAN1</value>
            </attributeFilter>
         </filteringPredicate>
      </eventCondition>
      <action actionProvider="TWSAction"
              actionType="sbs"
              responseType="onDetection">
         <description>Submit the calmonthlyrev job stream.</description>
         <parameter name="JobStreamName">
            <value>calmonthlyrev</value>
         </parameter>
         <parameter name="JobStreamWorkstationName">
            <value>act5cpu</value>
         </parameter>
      </action>
   </eventRule>
</eventRuleSet>
```

Définition de règles d'événement pour le scénario 4

Lorsque les événements job-x=exec et job-x=succ/abend sont reçues dans les 500 secondes, le planificateur doit répondre Yes à prompt-1 et démarrer le flot de travaux jobstream-z, sinon il doit envoyer un courrier électronique à twsoper@mycompany.com prévenant que le travail est en retard.

```
<?xml version="1.0"?>
<eventRuleSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns="http://www.ibm.com/xmlns/prod/tws/1.0/event-management/rules"
      xsi:schemaLocation="http://www.ibm.com/xmlns/prod/tws/1.0/
                          event-management/rules/EventRules.xsd">
   <eventRule name="scenario4_rule"
              ruleType="sequence"
```

```
        isDraft="yes">
      <description>This is the definition for scenario4</description>
      <timeZone>America/Buenos_Aires</timeZone>
      <timeInterval amount="500" unit="seconds" />
      <eventCondition eventProvider="TWSObjectsMonitor"
                      eventType="JobStatusChanged">
         <filteringPredicate>
            <attributeFilter name="JobName" operator="eq">
               <value>job-x</value>
            </attributeFilter>
            <attributeFilter name="InternalStatus" operator="eq">
               <value>EXEC</value>
            </attributeFilter>
         </filteringPredicate>
      </eventCondition>
      <eventCondition eventProvider="TWSObjectsMonitor"
                      eventType="JobStatusChanged">
         <filteringPredicate>
            <attributeFilter name="JobName" operator="eq">
                <value>job-x</value>
            </attributeFilter>
            <attributeFilter name="InternalStatus" operator="eq">
                <value>ABEND</value>
                <value>SUCC</value>
            </attributeFilter>
         </filteringPredicate>
      </eventCondition>
      <action actionProvider="MailSender"
              actionType="SendMail"
              responseType="onTimeOut">
         <description>Send email to operator saying that job-x is late</description>
         <parameter name="To">
            <value>twsoper@mycorp.com</value>
         </parameter>
         <parameter name="Subject">
            <value>Job-x is late by at least 5 minutes</value>
         </parameter>
      </action>
      <action actionProvider="TWSAction"
              actionType="Reply"
              responseType="onDetection">
         <description>Reply Yes to prompt-1</description>
         <parameter name="PromptName">
            <value>prompt-1</value>
         </parameter>
         <parameter name="PromptAnswer">
            <value>Yes</value>
         </parameter>
      </action>
      <action actionProvider="TWSAction"
              actionType="sbs"
              responseType="onDetection">
         <description>Submit jobstream-z</description>
         <parameter name="JobStreamName">
            <value>jobstream-z</value>
         </parameter>
         <parameter name="JobStreamWorkstationName">
            <value>act23cpu</value>
         </parameter>
      </action>
   </eventRule>
</eventRuleSet>
```

Définition de règles d'événement pour le scénario #5

Contrôlez l'état des processus Tivoli Workload Scheduler répertoriés dans ProcessName et exécutez le script par lots RUNCMDFM.BAT situé dans E:\production\eventRules.

Le mot clé TWSPATH indique le chemin qualifié complet où l'instance Tivoli Workload Scheduler contrôlée est installée, incluant le suffixe /TWS.

Sur les systèmes d'exploitation Windows, la règle d'événement est déclenchée chaque fois que l'agent est arrêté à l'aide de la commande ShutDownLwa et manuellement. Sur les systèmes d'exploitation UNIX, la règle d'événement est déclenchée lorsque vous arrêtez le processus manuellement, mais pas lorsque la commande ShutDownLwa est utilisée.

Si vous spécifiez ProcessName=agent, le composant agent est contrôlé, tandis que le processus TWS JobManager n'est pas contrôlé.

```
<?xml version="1.0"?>
<eventRuleSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns="http://www.ibm.com/xmlns/prod/tws/1.0/event-management/rules"
      xsi:schemaLocation="http://www.ibm.com/xmlns/prod/tws/1.0/
                          event-management/rules/EventRules.xsd">
    <eventRule name="scenario5rule" ruleType="filter" isDraft="no">
        <eventCondition name="twsProcMonEvt1"
        eventProvider="TWSApplicationMonitor"
        eventType="TWSProcessMonitor">
            <scope>
                AGENT, NETMAN DOWN ON WIN86MAS
            </scope>
            <filteringPredicate>
                <attributeFilter name="ProcessName" operator="eq">
                 <value>agent</value>
                 <value>appservman</value>
                 <value>batchman</value>
                 <value>jobman</value>
                 <value>mailman</value>
                 <value>monman</value>
                 <value>netman</value>
       </attributeFilter>
                <attributeFilter name="TWSPath" operator="eq">
                    <value>E:\Program Files\IBM\TWA\TWS</value>
                </attributeFilter>
                <attributeFilter name="Workstation" operator="eq">
                    <value>win86mas</value>
                </attributeFilter>
                <attributeFilter name="SampleInterval" operator="eq">
                    <value>5</value>
                </attributeFilter>

            </filteringPredicate>
        </eventCondition>
        <action actionProvider="GenericActionPlugin" actionType="RunCommand"
    responseType="onDetection">
            <scope>
                RUNCMDFM.BAT
            </scope>
            <parameter name="Command">
                <value>runCmdFM.bat</value>
            </parameter>
            <parameter name="WorkingDir">
                <value>E:\production\eventRules</value>
```

```
            </parameter>
        </action>
    </eventRule>
</eventRuleSet>
```

Définition de règle d'événement pour le scénario #7

Surveiller l'état du fichier Symphony du poste de travail *IT041866-9088* et consigner, dans la base de données d'audit, l'occurrence d'un enregistrement Symphony endommagé.

Sur chaque poste de travail, le processus batchman surveille le fichier Symphony. Lorsqu'il détecte un enregistrement endommagé, il envoie l'événement d'endommagement dans la file d'attente de messages du processus monman, puis au processeur d'événement sur le poste de travail maître.

La règle d'événement est déclenchée chaque fois que le processus batchman trouve un enregistrement de dépendance endommagé sur le poste de travail spécifié dans la définition de règle d'événement.

Si vous définissez la valeur du poste de travail sur IT041866-9088, le fichier Symphony de ce poste de travail est surveillé et la règle d'événement est déclenchée lorsque le processus batchman détecte un enregistrement endommagé dans le fichier Symphony.

L'occurrence de l'enregistrement endommagé est écrite dans le fichier d'audit de consignation de message.

```
?xml version="1.0"?>
<eventRuleSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.ibm.com/xmlns/prod/tws/1.0/event-management/rules"
xsi:schemaLocation="http://www.ibm.com/xmlns/prod/tws/1.0/event-management/rules
http://www.ibm.com/xmlns/prod/tws/1.0/event-management/rules/EventRules.xsd">
<eventRule name="TEST1" ruleType="filter" isDraft="no">
<eventCondition name="productAlertEvt1" eventProvider="TWSObjectsMonitor"
eventType="ProductAlert">
<scope>
IT041866-9088
</scope>
<filteringPredicate>
<attributeFilter name="Workstation" operator="eq">
<value>IT041866-9088</value>
</attributeFilter>

</filteringPredicate>
</eventCondition>
<action actionProvider="MessageLogger" actionType="MSGLOG"
responseType="onDetection">
<scope>
OBJECT=%{PRODUCTALERTEVT1.WORKSTATION} MESSAGE=%{PRODUCTALERTEVT1.WORKSTATION}
corruption
</scope>
<parameter name="ObjectKey">
<value>%{productAlertEvt1.Workstation}</value>
</parameter>
<parameter name="Message">
<value>%{productAlertEvt1.Workstation} corruption</value>
</parameter>
<parameter name="Severity">
<value>Info</value>
```

```
</parameter>
</action>
</eventRule>
</eventRuleSet>
```

Remarques sur le fonctionnement des règles

Le paragraphe suivant contient des informations essentielles sur le comportement des règles d'événement, qui peuvent vous aider à mieux comprendre l'origine de certains résultats inattendus :

Notes sur les règles d'état

- En fonction de ses dates de validité `from` et `to`, l'état d'une règle change à mesure du déploiement :
  - Si vous créez une règle dont les dates de validé `from` et `to` ont déjà expiré, la règle est sauvegardée à l'état `activation en instance`. Si la règle est déployée, elle reste à l'état `activation en instance`.
  - Si vous attribuez une date ultérieure à la zone de validité `to`, la règle est déployée à l'état `actif`. Si vous lui attribuez une date antérieure, la règle est déployée à l'état `inactif`.
- Les périodes d'activité de la règle (`start` et `end`) n'affectent pas son état. Tant que la règle se trouve dans les bonnes dates de validité, elle reste à l'état `actif`, qu'elle soit ou non dans sa période d'activité définie. Si le planificateur reçoit des événements définis par une règle hors de sa période d'activité, les événements sont annulés mais la règle reste à l'état `actif`.

Manque de persistance du statut qualifiant l'instance de règle d'événement

Tout arrêt ou défaillance du processeur d'événements entraîne la perte du statut des instances de règles pour lesquelles la correspondance est incomplète.

Répétition des instances de règles `set` et `sequence`

De façon générale, chaque règle active a une, et seulement une copie qui s'exécute sur le serveur de traitement d'événements. Les règles `Set` et `sequence` utilisent le mécanisme expliqué dans l'exemple suivant :

1. Vous définissez une règle `sequence` avec deux événements, A et B.
2. Lorsque le premier événement qui correspond à la séquence se produit (événement A), il active la règle et attend le second événement (événement B).

   Une fois la règle active, tout événement A additionnel pouvant survenir est ignoré. Aucune instance supplémentaire n'est créée pour aucun événement A nouvellement détecté tant que la règle attend l'événement B.
3. Lorsque l'événement B se produit, la règle est terminée et se réinitialise, attendant que l'événement A se produise à nouveau.

Le mécanisme des règles `set` et `sequence` est tel que toute occurrence additionnelle d'un événement déjà détecté est ignorée et rejetée si l'autre événement défini n'est pas arrivé.

Vous pouvez éviter ce problème en utilisant des attributs de corrélation. L'utilisation d'un ou plusieurs attributs de corrélation permet de diriger une règle de manière à créer une copie de règle séparée à la réception d'un événement A.

Les types de règle `Set` et `sequence` avec des fenêtres de temps d'activité inférieures à 24 heures

Les occurrences des règles `set` ou `sequence` qui ont été définies pour être actives pendant uniquement quelques heures chaque jour, ne sont pas purgées lorsque la période d'activité à l'intérieur de chaque jour expire si uniquement une partie des événements se produit. Elles restent en état de veille, attendant de recevoir les événements restants dans les jours qui suivent.

Par exemple, vous définissez une règle `set` qui inclut deux événements. La règle est valide du 1er janvier au 10 janvier et est active quotidiennement de 6 heures à 10 heures du matin.

Si le 1er janvier, le premier événement est reçu à 8 heures du matin, la règle attend le second événement et reste *active* au-delà de 10 heures du matin si l'événement n'est pas détecté.

Si le second événement se produit à 11 heures du matin (ce qui est en dehors de la fenêtre de temps d'activité), il est rejeté mais la règle reste *active*. Si le second événement est reçu à nouveau à 7 heures du matin le 2 janvier, la règle est déclenchée et les actions sont mises en oeuvre.

Si vous ne souhaitez pas que la règle soit *reportée* au jour suivant, vous devez la purger.

Eléments de règles déclenchés

Chaque fois que les conditions d'événements répertoriées dans une règle d'événement déployée sont remplies ou qu'un dépassement de délai a lieu, une instance de règle d'événement est créée. Une instance de règles d'événement inclut des informations telles que le nom de la règle d'événement, la date et l'heure à laquelle la concordance a eu lieu et la liste des instances d'actions, ainsi que leur statut,dont l'exécution a eu lieu à la suite d'un correspondance avec les conditions d'événement. L'objet `RuleInstance` est utilisé pour rassembler les informations sur les règles déclenchées dans un journal d'historique des instances de règles.

Les actions effectuées par une règle déclenchée sont collectées dans un journal d'historique des exécutions d'action. Les informations fournies incluent les exécutions d'action qui se sont terminées par une erreur ou par un avertissement par opposition à celles qui se sont terminées sans problème. Si au moins une action s'achève en générant une erreur, l'instance totale de la règle est consignée comme une erreur. Dans les informations suivies au niveau des exécutions d'action, les zones de règle sont également conservés et des requêtes peuvent être exécutées pour rechercher des exécutions d'action en fonction de ces zones (l'identificateur d'instance de règle est également disponible).

Définition d'événements personnalisés

Outre les types et les classes d'événements déjà définis (également appelés "fournisseurs") et répertoriés en détail à la section «Fournisseurs d'événements et définitions», à la page 707, Tivoli Workload Scheduler fournit un modèle générique de création d'événements intitulé `GenericEventPlugIn`, modifiable par les programmeurs ayant des compétences spécifiques dans certaines applications et en langage XML, afin de définir des types d'événements personnalisés utiles à l'organisation.

Les outils servant à définir des types d'événements personnalisés sont les suivants :

- Le fournisseur d'événements `GenericEventPlugIn` en langage XML

- La commande d'utilitaire evtdef, avec laquelle un programmeur peut télécharger le fournisseur d'événements `GenericEventPlugIn` sous forme de fichier local afin de définir des événements personnalisés
- Les fichiers de définition de schéma XML (fichiers XSD) nécessaires à la validation du fournisseur d'événements génériques modifiés. Ces fichiers contiennent également des directives en ligne pour faciliter le travail des programmeurs.
- La commande d'utilitaire sendevent, qui permet d'envoyer les événements personnalisés au serveur de traitement d'événements afin de déclencher des règles à partir de n'importe quel poste de travail simplement via le client de ligne de commande à distance de Tivoli Workload Scheduler.

La procédure de définition et d'utilisation des événements personnalisés est la suivante :

1. La commande evtdef permet aux programmeurs d'effectuer les tâches suivantes :
   a. Télécharger le fournisseur d'événements générique sous forme de fichier local.
   b. Suivre les définitions de schéma afin d'ajouter des types d'événements personnalisés et définir leurs propriétés et attributs dans le fichier via un éditeur XML.
   c. Charger le fichier local en tant que fournisseur d'événements génériques modifié contenant les nouvelles définitions de types d'événements personnalisées. Le fournisseur d'événements génériques modifié est sauvegardé sous forme de fichier XML sur le gestionnaire de domaine maître.
2. Le générateur de règles ou l'administrateur définit, via le composeur ou Dynamic Workload Console, les règles d'événement devant être déclenchées par ces événements personnalisés, en spécifiant :
   - Le fournisseur d'événements génériques en tant que fournisseur d'événements
   - Les types d'événements personnalisés en tant que types d'événements
   - Les propriétés (ou attributs) des types d'événements personnalisés défini(e)s pour les événements personnalisés dans le fournisseur d'événements génériques, avec les valeurs particulières qui déclencheront les règles.
3. Déployer les règles.
4. Lorsque l'occurrence d'un événement personnalisé survient, cet événement peut être envoyé vers le serveur de traitement d'événements selon l'une des méthodes suivantes :
   - Via la commande sendevent exécutée à partir d'un script ou d'une ligne de commande
   - Via une autre application, telle que Tivoli Enterprise Console ou Tivoli Monitoring

   Dès la réception de l'événement par le serveur de traitement d'événements, le déclenchement de la règle a lieu.

Chapitre 8. Définition d'objets dans la base de données

Tivoli Workload Scheduler exploite un ensemble de définitions d'objets qui servent à mapper votre environnement de planification dans la base de données. Les objets de planification sont :

- agendas
- domaines
- règles d'événement
- travaux
- flots de travaux
- invites
- paramètres
- ressources
- groupes de cycle d'exécution
- tables de variables
- applications de charge de travail
- postes de travail
- classes de postes de travail
- utilisateurs

Définition des objets de planification

Les objets de planification sont gérés par le programme de ligne de commande **composer** et stockés dans la base de données Tivoli Workload Scheduler. Le programme de ligne de commande **composer** peut être installé et utilisé sur toute machine connectée via TCP/IP au système sur lequel le gestionnaire de domaine maître est installé. Il ne nécessite pas l'installation d'un poste de travail Tivoli Workload Scheduler comme condition préalable. Il communique via HTTP/HTTPS avec le gestionnaire de domaine maître sur lequel la base de données DB2 est installée. La configuration de la communication HTTP/HTTPS et la vérification d'authentification sont gérées par l'infrastructure WebSphere Application Server. Le programme **composer** utilise des fichiers d'édition pour mettre à jour la base de données de planification. Le format du fichier d'édition utilisé pour définir un objet spécifique est décrit plus loin dans le présent chapitre. Par exemple, pour créer un objet, vous devez d'abord entrer sa définition dans un fichier d'édition, puis utiliser **composer** pour l'ajouter à la base de données en indiquant le fichier d'édition qui contient la définition. Le programme de ligne de commande **composer** vérifie la syntaxe dans le fichier d'édition, et si elle est correcte, il convertit la définition de l'objet en langage XML, puis envoie la demande au gestionnaire de domaine maître via HTTP/HTTPS.

Sur le gestionnaire de domaine maître, la définition XML est analysée, des contrôles de sémantique et d'intégrité sont effectués, puis la mise à jour est stockée dans la base de données.

Dans cette version du produit, toutes les entrées sont gérées individuellement. Le mécanisme de verrouillage des objets est une autre fonction disponible dans cette nouvelle version. Les objets de planification définis dans la base de données sont verrouillés lorsqu'un utilisateur les utilise afin d'empêcher les accès simultanés. Cela signifie que seul l'utilisateur qui verrouille l'objet possède un droit d'accès en `écriture`pour cet objet et que les autres utilisateurs ne possèdent un droit d'accès en `lecture uniquement`. Pour plus d'informations, voir «lock», à la page 341 et «unlock», à la page 357.

Vous pouvez utiliser des mots clés longs et courts lorsque vous émettez des commandes à partir de **composer**, comme le montre le tableau 20. Les deux premières colonnes sur la gauche répertorient les formats de mots-clés longs et courts actuellement supportés par Tivoli Workload Scheduler. La colonne la plus à droite répertorie les formats compatibles avec les versions antérieures à utiliser si votre réseau inclut des installations antérieures à la version 8.3.

Tableau 20. Liste des mots clés d'objet de planification pris en charge

Mots clés longs	Mots clés courts	Mots clés compatibles avec les installations antérieures à la version 8.3
agenda	cal	agendas
domain	dom	cpu
eventrule	erule | er	—
jobdefinition	jd	jobs
jobstream	js	sched
paramètre	parm	parms
prompt	prom	invites
resource	res	ressources
user	user	users
variabletable	vt	—
workstation	ws	cpu
workstationclass	wscl	cpu

Remarque : Le mot clé **cpu** est conservé pour la compatibilité des domaines, stations de travail et classes de station de travail avec les versions antérieures.

Le programme **composer** n'émet pas d'avertissement particulier lorsque des mots clés du langage de planification sont utilisés comme noms d'objet de planification. Toutefois, l'utilisation de ces mots clés peut entraîner des erreurs ; il est donc conseillé d'éviter d'utiliser les mots clés répertoriés dans le tableau 21 lorsque vous définissez des travaux et des flots de travaux :

Tableau 21. Liste des mots réservés lors de la définition de travaux et de flots de travaux

Liste des mots réservés :				
abendprompt	after	as	at	autodocoff
autodocon	canc	carryforward	confirmed	continue
dateval	day(s)	day_of_week	deadline	description
docommand	draft	end	every	everyday
except	extraneous	fdignore	fdnext	fdprev
filename	follows	freedays	from	go
hi	i18n_id	i18n_priority	interactive	isdefault
isuserjob	jobfilename	jobs	keyjob	keysched
limit	matching	members	needs	nextjob
notempty	number	on	onuntil	op
opens	order	previous	priority	prompt
qualifier	rccondsucc	reprise	relative	request

Tableau 21. Liste des mots réservés lors de la définition de travaux et de flots de travaux (suite)

Liste des mots réservés :				
rerun	runcycle	sa	sameday	schedtime
schedule	scriptname	stop	streamlogon	su
tasktype	timezone	to	token_in	until
validfrom	validto	vartable	vt	weekday(s)
workday(s)				

Evitez d'utiliser les mots clés répertoriés dans le tableau 22 lors de la définition de postes de travail, de classes de postes de travail et de domaines :

Tableau 22. Liste des mots réservés lors de la définition de postes de travail

Liste des mots réservés :				
access	AIX	agent_type	autolink	behindfirewall
command	cpuclass	cpuname	description	domain
enabled	end	extraneous	for	force
fta	fullstatus	host	hpux	ibm i
ignore	isdefault	linkto	maestro	manager
master	members	mpeix	mpev	mpexl
mpix	node	number	off	on
os	other	parent	posix	server
secureaddr	securitylevel	tcpaddr	timezone	type
tz	tzid	UNIX	using	vartable
wnt				

Evitez d'utiliser les mots clés répertoriés dans le tableau 23 lors de la définition d'utilisateurs :

Tableau 23. Liste des mots réservés lors de la définition d'utilisateurs

Liste des mots réservés :		
username	password	end

Utilisation des modèles de définition d'objet

Les modèles de définition d'objet de planification sont à votre disposition dans le répertoire *racine_TWS*\templates. Vous pouvez les utiliser comme point de départ pour la définition des objets de planification.

Notez que les dates des modèles sont présentées au format indiqué dans l'option locale `date format`.

Définition de poste de travail

Dans un réseau Tivoli Workload Scheduler, un poste de travail est un objet de planification qui exécute des travaux. Vous créez et gérez le poste de travail d'objet de planification dans la base de données Tivoli Workload Scheduler à l'aide d'une définition de poste de travail. Une définition de poste de travail est requise pour

tout objet qui exécute les travaux. En général, une définition de poste de travail permet de représenter un poste de travail physique. Cependant, en présence d'agents étendus, par exemple, elle représente une définition logique qui doit être hébergée par un poste de travail physique.

Vous pouvez faire figurer plusieurs définitions de poste de travail dans un même fichier texte, avec des définitions de classes de postes de travail et des définitions de domaines. Une définition de poste de travail a la syntaxe suivante :

Syntaxe

cpuname *poste_de_travail* [**description** *"description"*]
[**vartable** *nom_table*]
os *type_sys_expl*
[**node** *nom_hôte*] [**tcpaddr** *port*]
[**secureaddr** *port*][**timezone**|**tz** *fuseau_horaire*]
[**domain** *nom_domaine*]
[**for maestro** [**host** *host-workstation* [**access** *method* | **agentID** *agentID*]]
[**type fta** | **s-agent** | **x-agent** | **manager** | **broker** | **agent** | **rem-eng** | **pool** | **d-pool**]
[**ignore**]
[**autolink on** | **off**]
[**behindfirewall on** | **off**]
[**securitylevel enabled** | **on** | **force**]
[**fullstatus on** | **off**]
[**server** *id_serveur*]]
[**protocole** *http* | *htpps*]
[**membres** [*poste_de_travail*] [...]]
[**exigences** *définition_jsdl*]]
end

[**cpuname** ...]

[**cpuclass** ...]

[**domain** ...]

Arguments

Tableau 24. Paramètres des attributs pour la gestion des types de poste de travail

Attributs	Gestionnaire de domaine maître	Gestionnaire de domaine	Gestionnaire de domaine de sauvegarde
cpuname	Nom du poste de travail.		
description	Description du poste de travail entre guillemets. Cet attribut est facultatif.		
vartable	Nom d'une table de variables associée à ce poste de travail. Les variables utilisées avec le poste de travail sont définies dans cette table. Cet attribut est facultatif.		
os	Système d'exploitation installé sur le système. Spécifiez l'une des valeurs suivantes : UNIX WNT OTHER IBM_i		
node	Nom d'hôte ou adresse IP du système.		

Tableau 24. Paramètres des attributs pour la gestion des types de poste de travail (suite)

Attributs	Gestionnaire de domaine maître	Gestionnaire de domaine	Gestionnaire de domaine de sauvegarde
tcpaddr	Valeur affectée à *nm port* dans le fichier `localopts`. Pour plusieurs postes de travail sur un système, entrez un numéro de port non utilisé. La valeur par défaut est 31111.		
secureaddr	Valeur affectée à *nm ssl port* dans le fichier `localopts`. Spécifiez-la si **securitylevel** est défini sur `on`, `force` ou `enabled`.		
timezone | tz	Fuseau horaire du système. La valeur doit correspondre à celle définie sur le système d'exploitation.		
domaine	`MASTERDM`	Nom du domaine géré.	
hôte	Non applicable		
access	Non applicable		
type	`gestionnaire`		`fta`
ignore	Utilisez cet attribut si vous souhaitez que ce poste de travail ne s'affiche pas sur le prochain plan de production.		

autolink	Indique si un lien entre les postes de travail s'ouvre automatiquement au démarrage. Spécifiez l'une des valeurs suivantes :
	
	`ON`
	`OFF`
	
	Il s'agit d'un attribut facultatif. La valeur par défaut est `ON`.		

behindfirewall	Ce paramètre est ignoré.	Il indique si un pare-feu existe entre le poste de travail et le gestionnaire de domaine maître. Spécifiez l'une des valeurs suivantes :
		
		`ON`
		`OFF`
		
		La valeur par défaut est `OFF`.	

securitylevel	Type de l'authentification SSL à utiliser :
	
	`enabled`
	`on`
	`force`		

fullstatus	`ON`		
serveur	Non applicable		Ce paramètre est ignoré.
protocole	Non applicable		
membres	Non applicable		
exigences	Non applicable		

Le tableau 25, à la page 152 décrit les valeurs que vous avez définies pour chaque attribut des types de poste de travail cible. Vous trouverez ci-dessous un tableau avec des détails supplémentaires sur chaque attribut.

Tableau 25. Paramètres des attributs pour les types de poste de travail cible

Attribut	Agent tolérant aux pannes et agent standard	Poste de travail courtier de charge de travail	Agent étendu	Agent	Poste de travail de moteur distant	Pool	Pool dynamique
cpuname	Nom du poste de travail.						
description	Description du poste de travail entre guillemets. Cet attribut est facultatif.						
vartable	Nom d'une table de variables associée à ce poste de travail. Les variables utilisées avec le poste de travail sont définies dans cette table. Cet attribut est facultatif.						

os	Système d'exploitation installé sur le système. Spécifiez l'une des valeurs suivantes :
	
	`UNIX`
	`WNT`
	`OTHER`
	`IBM_i`
	
	Spécifiez `OTHER` pour les systèmes IBM i qui s'exécutent en tant qu'agents tolérants aux pannes limités.
		`OTHER`	Système d'exploitation installé sur la machine. Spécifiez l'une des valeurs suivantes :
			
			`UNIX`
			`WNT`
			`OTHER`
			`IBM_i`
				Ce paramètre de valeur est reconnu sur le système.	Système d'exploitation installé sur la machine. Spécifiez l'une des valeurs suivantes :
					
					`UNIX`
					`WNT`
					`ZOS`
						Système d'exploitation installé sur la machine. Spécifiez l'une des valeurs suivantes :
						
						`UNIX`
						`WNT`
						`OTHER`
						`IBM_i`	

node	Nom d'hôte ou adresse IP du système.		Nom d'hôte ou adresse IP du système. Spécifiez `NULL` lorsque **host** est défini sur `$MASTER`, ou lorsque vous définissez un agent étendu pour PeopleSoft, SAP ou Oracle.	Nom d'hôte ou adresse IP de l'agent.	Nom d'hôte ou adresse IP du moteur distant.	Non applicable	
tcpaddr	Valeur affectée à *nm port* dans le fichier `localopts`. Lorsque vous définissez plusieurs postes de travail sur un système, entrez un numéro de port non utilisé. La valeur par défaut est 31111.	Valeur affectée à *nm port* dans le fichier `localopts`. Lorsque vous définissez plusieurs postes de travail sur un système, entrez un numéro de port non utilisé. La valeur par défaut est 41114.	Voir les spécifications de la méthode d'accès sélectionnée.	Numéro de port permettant de communiquer avec l'agent lorsque le protocole est `http`.	Numéro de port permettant de communiquer avec le moteur distant lorsque le protocole est `http`.	Non applicable	
secureaddr	Valeur affectée à *nm ssl port* dans le fichier `localopts`. Spécifiez-la si **securitylevel** est défini sur `on`, `force` ou `enabled`.	Non applicable	Non applicable	Numéro de port permettant de communiquer avec l'agent lorsque le protocole est `https`.	Numéro de port permettant de communiquer avec le moteur distant lorsque le protocole est `https`.	Non applicable	

Tableau 25. Paramètres des attributs pour les types de poste de travail cible (suite)

Attribut	Agent tolérant aux pannes et agent standard	Poste de travail courtier de charge de travail	Agent étendu	Agent	Poste de travail de moteur distant	Pool	Pool dynamique
timezone | tz	Fuseau horaire du système. La valeur doit correspondre à celle définie sur le système d'exploitation.		Le fuseau horaire défini sur le poste de travail spécifié dans l'attribut **host**.	Le fuseau horaire défini sur l'agent.	Le fuseau horaire défini sur le moteur distant.	Le fuseau horaire défini sur les agents du pool.	Le fuseau horaire défini sur les agents du pool dynamique.
domaine	Spécifiez un domaine existant. La valeur par défaut pour les agents tolérants aux pannes est `MASTERDM`. Ce paramètre est obligatoire pour les agents standard.	Spécifiez un domaine existant. Ce paramètre est obligatoire.	Ce paramètre est nécessaire uniquement si la valeur affectée à **host** est `$MANAGER`.	Non applicable			
hôte	Non applicable		Poste de travail hôte. Il peut être défini sur `$MASTER` ou `$MANAGER`.	Le poste de travail courtier.			
access	Non applicable			Sélectionnez le nom du fichier de méthode d'accès approprié.	Non applicable		
agentID				Identificateur unique de l'agent			

type	`fta`
	`s-agent`
	
	La valeur par défaut est `fta`.
	
	Spécifiez `fta` pour les systèmes IBM i qui s'exécutent en tant qu'agents tolérants aux pannes limités.
		`broker`	`x-agent`	`agent`	`rem-eng`	`pool`	`d-pool`

ignore	Utilisez cet attribut si vous ne souhaitez pas que ce poste de travail apparaisse dans le plan de production suivant.						

autolink	Indique si un lien entre les postes de travail est automatiquement ouvert au démarrage. Spécifiez l'une des valeurs suivantes :
	
	`ON`
	`OFF`
	
	Il s'agit d'un attribut facultatif. La valeur par défaut est `ON`.
			`OFF`	Non applicable			

Tableau 25. Paramètres des attributs pour les types de poste de travail cible (suite)

Attribut	Agent tolérant aux pannes et agent standard	Poste de travail courtier de charge de travail	Agent étendu	Agent	Poste de travail de moteur distant	Pool	Pool dynamique

behindfirewall	Il indique si un pare-feu existe entre le poste de travail et le gestionnaire de domaine maître. Spécifiez l'une des valeurs suivantes :
	
	`ON`
	`OFF`
	
	La valeur par défaut est `OFF`.
			`OFF`	Non applicable			

securitylevel	Type de l'authentification SSL à utiliser :
	
	`enabled`
	`on`
	`force`
	
	Non applicable pour les systèmes IBM i qui s'exécutent en tant qu'agents tolérants aux pannes limités.
		Non applicable					

fullstatus	Indique si le poste de travail est mis à jour pour l'état de traitement des travaux dans son domaine et ses sous-domaines. Spécifiez l'une des valeurs suivantes :
	
	`ON`
	`OFF`
	
	Spécifiez `OFF` pour les agents standard.
		`OFF`		Non applicable			

serveur	0-9, A-Z. Lorsqu'il est spécifié, requiert la création de processus mailman dédiés sur le poste de travail parent.		Non applicable				

protocole	Non applicable			Spécifiez l'une des valeurs suivantes :
				
				`http`
				`https`
				
				Cet attribut est facultatif. S'il n'est pas spécifié, il est automatiquement déterminé à partir des paramètres spécifiés pour **tcpaddr** et **secureaddr**.
						Non applicable	

membres	Non applicable					Valeur obligatoire	Non applicable
exigences	Non applicable						Valeur obligatoire

La liste suivante fournit des détails supplémentaires sur les attributs de définition de poste de travail :

cpuname *poste_de_travail*
Indique le nom du poste de travail. Les noms de poste de travail doivent être uniques et ne peuvent pas être identiques à des noms de classes de postes de travail.

Il doit commencer par une lettre et peut contenir des caractères alphanumériques, des tirets et des traits de soulignement. Il peut comporter jusqu'à 16 caractères.

N'utilisez pas dans cette zone l'un des mots réservés spécifiés dans le tableau 21, à la page 148.

description *"description"*
Fournit une description du poste de travail. La description peut contenir jusqu'à 120 caractères alphanumériques. Ce texte doit être délimité par des guillemets.

vartable *nom_table*
Indique le nom de la table de variables que vous souhaitez associer au poste de travail. Les variables utilisées avec le poste de travail sont définies dans cette table.

Il doit commencer par une lettre et peut contenir des caractères alphanumériques, des tirets et des traits de soulignement. Il peut contenir jusqu'à 80 caractères.

os *type_système-exploitation*
Indique le système d'exploitation du poste de travail. Lorsqu'il est utilisé dans les définitions de poste de travail de moteur distant, il correspond au système d'exploitation du moteur distant Tivoli Workload Scheduler.

Les valeurs correctes sont les suivantes :

UNIX Pour les systèmes d'exploitation pris en charge sur les systèmes basés sur UNIX, y compris les systèmes LINUX.

WNT Pour les systèmes d'exploitation Windows pris en charge.

OTHER
Valeur obligatoire pour : les postes de travail Tivoli Dynamic Workload Broker et les systèmes IBM i qui s'exécutent en tant qu'agents tolérants aux pannes limités. Valeur facultative pour d'autres types de postes de travail.

ZOS Utilisé avec les postes de travail distants qui sont définis pour communiquer avec le moteur distant Tivoli Workload Scheduler for z/OS.

IBM_i Pour les systèmes d'exploitation IBM i pris en charge.

Remarque : Pour obtenir la liste des systèmes d'exploitation pris en charge, reportez-vous au document *IBM Tivoli Workload Scheduler System Requirements*, à l'adresse http://www-01.ibm.com/support/docview.wss?rs=672&uid=swg27041009.

node *nom_hôte*
Indiquez le nom d'hôte ou l'adresse TCP/IP du poste de travail. Les noms de domaines complets sont admis.

Pour les noms d'hôtes, les caractères alphanumériques sont autorisés, y compris le tiret (-). La longueur est limitée à 51 caractères.

Spécifiez `NULL` lorsque :

- Vous définissez un agent étendu pour PeopleSoft, SAP ou Oracle.
- **host** est défini sur `$MASTER`

Si vous définissez un poste de travail distant, spécifiez le nom d'hôte du système sur lequel le moteur distant est installé.

tcpaddr *port*
Indique le numéro de port TCP/IP **netman** que Tivoli Workload Scheduler utilise pour la communication entre les postes de travail.

Pour les postes de travail courtiers de charge de travail
Spécifiez la valeur de la propriété **TWS.Agent.Port** du fichier `TWSAgentConfig.properties`.

Pour les postes de travail distants utilisant le protocole HTTP pour communiquer avec le moteur distant
Spécifiez le numéro de port HTTP du moteur distant.

Pour les autres types de postes de travail
Spécifiez la valeur affectée à la variable *nm port* dans le fichier `localopts`.

La valeur par défaut de cette zone est `31111`. Indiquez une valeur comprise entre `1` et `65535`.

secureaddr
Définit le port utilisé pour détecter les connections SSL entrantes. Cette valeur est lue lorsque l'attribut **securitylevel** est défini.

Pour les postes de travail courtiers de charge de travail
Ignorez cet attribut.

Pour les postes de travail de moteur distant utilisant le protocole HTTPS pour communiquer avec le moteur distant
Spécifiez le numéro de port HTTPS du moteur distant.

Pour les autres types de postes de travail
Spécifiez la valeur affectée à la variable *nm ssl port* dans le fichier `localopts`. La valeur doit être différente de celle affectée à la variable *nm port* dans le fichier `localopts`.

Si **securitylevel** est spécifié, mais que cet attribut est manquant, la valeur par défaut de cette zone est `31113`. Indiquez une valeur comprise entre `1` et `65535`.Voir *IBM Tivoli Workload Scheduler - Guide d'administration* pour plus d'informations sur l'authentification SSL et les options locales définies dans le fichier de configuration *racine_TWS*`/localopts`.

timezone | **tz** *fuseau_horaire*
Indique le fuseau horaire du poste de travail. Pour garantir l'exactitude des heures de planification, ce fuseau horaire doit être identique à celui du système d'exploitation de l'ordinateur.

Lorsqu'il est utilisé dans les définitions de poste de travail de moteur distant, il correspond au fuseau horaire défini sur le moteur distant Tivoli Workload Scheduler.

Pour plus d'informations sur les noms de fuseau horaire valides, voir Chapitre 16, «Gestion des fuseaux horaires», à la page 653.

domain *nom_domaine*
Indique le nom du domaine Tivoli Workload Scheduler auquel appartient le poste de travail. La valeur par défaut pour un poste de travail tolérant aux pannes est `MASTERDM`.

Tivoli Workload Scheduler ignore le paramètre de domaine lorsqu'il est défini pour les agents étendus, excepté lorsque l'attribut **host** est défini par `$MASTER`.

Ce paramètre est obligatoire pour les postes de travail d'agent standard et de courtier de charge de travail dynamiques.

host *poste_de_travail_hôte*
Cet attribut est obligatoire pour les agents étendus et les postes de travail de moteur distant et spécifie :

Pour les postes de travail de moteur distant, les agents, les pools et les pools dynamiques :
Le poste de travail courtier hébergeant le poste de travail. Cette zone ne peut pas être mise à jour après la création du poste de travail de moteur distant.

Pour les agents étendus
Le poste de travail avec lequel l'agent étendu communique et sur lequel sa méthode d'accès est installée. Le poste de travail hébergeant ne peut pas être un autre agent étendu.

Si le poste de travail hébergeant est un gestionnaire de domaine pour lequel vous avez défini une sauvegarde, vous pouvez spécifier une des valeurs suivantes pour garantir que l'agent étendu n'est pas isolé si le poste de travail hébergeant devient indisponible :

$MASTER
Indiquer que le poste de travail hôte de l'agent étendu est le gestionnaire de domaine maître.

$MANAGER
Indiquer que le poste de travail hôte de l'agent étendu est le gestionnaire de domaine. Dans ce cas, vous devez spécifier le domaine dans lequel est situé l'agent.

Dans ce cas, vérifiez que les méthodes d'agent étendu sont également installées sur le poste de travail de sauvegarde. Vous pouvez activer et désactiver la résolution automatique de la clé `$MASTER` à l'aide de l'option *mm resolve master* dans le fichier `localopts`.

Pour plus d'informations sur les options disponibles dans le fichier `localopts`, voir *IBM Tivoli Workload Scheduler - Guide d'administration*.

access *méthode*
Indique une méthode d'accès pour des agents étendus et des agents réseau. Il doit s'agir du nom d'un fichier situé dans le répertoire *racine_TWS*`/methods` du poste de travail hébergeant.

Spécifiez `NULL` lorsque vous définissez un agent étendu pour PeopleSoft, SAP ou Oracle.

agentID *agentID*
Identificateur unique de l'agent.

type Indique le type de poste de travail. Si vous planifiez la modification des types de postes de travail, prenez en compte les règles suivantes :

- Vous pouvez modifier les postes de travail d'agent tolérant aux pannes, d'agent standard, d'agent étendu, de gestionnaire de domaine et de courtier de charge de travail dynamiques par tout type de poste de travail, à l'exception de l'agent dynamique, du pool, du pool dynamique et du moteur distant.
- Vous ne pouvez pas modifier le type de l'agent dynamique, du pool, du pool dynamique et du moteur distant.

Entrez l'une des valeurs suivantes :

fta Si vous définissez un *agent tolérant aux pannes*, il s'agit d'un poste de travail d'agent qui lance les travaux et résout les dépendances locales sans gestionnaire de domaine. Il s'agit de la valeur de la valeur par défaut de cet attribut.

Vous devez spécifier `fta` si vous souhaitez affecter au poste de travail le rôle de gestionnaire de domaine de sauvegarde ou de gestionnaire de domaine maître de sauvegarde.

Spécifiez `fta` pour les systèmes IBM i qui s'exécutent en tant qu'agents tolérants aux pannes limités.

s-agent
Si vous définissez un *agent standard*, il s'agit d'un poste de travail d'agent qui lance les travaux uniquement sous l'ordre de son gestionnaire de domaine.

x-agent
Si vous définissez un *agent étendu*, il s'agit d'un poste de travail d'agent qui lance les travaux uniquement sous l'ordre de son poste de travail hébergeant. Les agents étendus peuvent servir d'interface entre Tivoli Workload Scheduler et des systèmes ou des applications non Tivoli.

Pour plus d'informations, voir *IBM Tivoli Workload Scheduler for Applications*.

manager
Si vous définissez un *gestionnaire de domaine*, il s'agit d'un poste de travail qui gère un domaine. Lors de la définition de ce type de poste de travail, spécifiez :

Serveur
NULL

Domaine, zone
Nom du domaine géré par le poste de travail, s'il est différent du domaine `MASTERDM`.

Vous spécifiez qu'un poste de travail est un gestionnaire également dans la zone *manager* de la «Définition de domaine», à la page 167. Tivoli Workload Scheduler vérifie automatiquement que les valeurs spécifiées dans ces zones sont cohérentes.

broker
Si vous définissez un poste de travail *courtier de charge de travail dynamique*, il s'agit d'un poste de travail qui exécute à la fois les types de travaux existants et les types de travaux avec options avancées. Il s'agit du serveur courtier installé avec le gestionnaire

de domaine maître et le gestionnaire de domaine dynamique. Il héberge les postes de travail suivants :

- Agent étendu
- Moteur distant
- Pool
- Pool dynamique
- Agent. Cette définition comprend les agents suivants :
  - agent
  - agent Tivoli Workload Scheduler for z/OS
  - agent pour z/OS

  Pour plus d'informations sur l'agent et l'agent Tivoli Workload Scheduler for z/OS, voir *Scheduling Workload Dynamically*. Pour plus d'informations sur le agent pour z/OS, voir *Planification avec l'agent pour z/OS*.

agent Si vous définissez un *agent dynamique*, il s'agit d'un poste de travail qui gère une grande variété de types de travaux, par exemple, des travaux de base de données spécifiques ou de transfert de fichiers, en plus des types de travaux traditionnels. Il est hébergé par le poste de travail Workload Broker. La définition de poste de travail est automatiquement créée et enregistrée lorsque vous installez le composant agent dynamique. Dans sa définition, vous pouvez éditer uniquement les attributs suivants :

- Description
- Vartable

Remarque : Si l'option globale `enAddWorkstation` vaut `"yes"`, la définition de poste de travail d'agent dynamique est automatiquement ajoutée au plan après que le processus d'installation a créé le poste de travail d'agent dynamique dans la base de données.

rem-eng

Si vous définissez un *poste de travail de moteur distant*, il s'agit d'un poste de travail utilisé pour communiquer avec un moteur distant lors de la liaison d'un travail défini localement, appelé *travail reflet*, à un travail spécifique qui s'exécute sur le moteur distant, appelé *travail distant*. Lorsque les deux travaux sont liés, la transition d'état du travail reflet est mappée vers la transition d'état du travail distant. Ce mappage permet également de définir et de surveiller les dépendances des travaux locaux sur les travaux qui s'exécutent sur le moteur distant ; ces dépendances sont appelées *dépendances croisées*.

Pour plus d'informations sur les travaux reflet et les dépendances croisées, voir Chapitre 19, «Définition et gestion des dépendances croisées», à la page 681.

Lors de la définition de ce type de poste de travail, renseignez les zones suivantes :

os Le système d'exploitation du moteur distant.

hôte Le nom du poste de travail courtier hébergeant.

node Le nom d'hôte ou l'adresse IP du moteur distant.

Lorsque vous spécifiez le numéro de port à utiliser pour communiquer avec le moteur distant, utilisez **secureaddr** si le protocole utilisé est HTTPS ou **tcpaddr** si le protocole utilisé est HTTP. Il est recommandé de spécifier dans la zone **fuseau horaire** le fuseau horaire défini sur le moteur distant.

pool Si vous définissez un *pool*, il s'agit d'un ensemble d'agents dynamiques avec des caractéristiques matérielles ou logicielles identiques sur lequel vous soumettez les travaux. Ce poste de travail est hébergé par le poste de travail Workload Broker. Dans sa définition, vous pouvez éditer uniquement les attributs suivants :

- Description
- Vartable
- Members

d-pool Si vous définissez un *pool dynamique*, il s'agit d'un ensemble d'agents dynamiques qui est basé dynamiquement sur les exigences répertoriées dans le fichier JSDL spécifié dans l'attribut **resources**. Ce poste de travail est hébergé par le poste de travail Workload Broker. Dans sa définition, vous pouvez éditer uniquement les attributs suivants :

- Description
- Vartable
- Requirements

ignore Indique que la définition de poste de travail ne doit pas être ajoutée au plan de production. Si vous spécifiez ce paramètre, les flots de travaux planifiés pour s'exécuter sur ce poste de travail ne sont pas ajoutés au plan de production.

autolink
Indique si les postes de travail doivent être connectés au démarrage. Selon le type du poste de travail, lorsque vous définissez sa valeur sur `on` :

Sur un agent tolérant aux pannes ou un agent standard
Cela implique que le gestionnaire de domaine ouvre le lien à l'agent lorsque le gestionnaire de domaine est démarré.

Sur un gestionnaire de domaine
Cela implique que ses agents ouvre des liens au gestionnaire de domaine à leur démarrage.

Ce paramètre est particulièrement utile lorsqu'un nouveau plan de production est créé sur le gestionnaire de domaine maître : dans le cadre de la génération du plan de production, tous les postes de travail sont arrêtés puis redémarrés. Pour tous les agents pour lesquels l'option **autolink** est activée, le gestionnaire de domaine envoie automatiquement une copie du nouveau plan de production et démarre l'agent. Si l'option **autolink** est également activée pour le gestionnaire de domaine, l'agent, à son tour, se connecte au gestionnaire de domaine.

Si la valeur de l'option **autolink** est `off` pour un agent, vous pouvez ouvrir un lien à partir de son gestionnaire de domaine en exécutant la commande **conman link** sur le gestionnaire de domaine ou le gestionnaire de domaine maître de l'agent.

behindfirewall
Si cette option est définie sur `on`, cela signifie qu'il existe un pare-feu entre le poste de travail et le gestionnaire de domaine maître. Dans ce cas, seule

une connexion directe entre le poste de travail et son gestionnaire de domaine est autorisée et les commandes **start** *poste_de_travail*, **stop** *poste_de_travail* et **showjobs** sont envoyées selon la hiérarchie du domaine. Il n'est pas nécessaire de faire établir une connexion directe avec le poste de travail par le gestionnaire de domaine maître ou le gestionnaire de domaine.

Définissez cet attribut par `off` si vous spécifiez un poste de travail de type `broker`.

fullstatus
Spécifiez ce paramètre lorsque vous définissez un poste de travail d'agent tolérant aux pannes. Pour les gestionnaires de domaine, ce mot clé est automatiquement défini sur `on`. Spécifiez l'une des valeurs suivantes :

on Si vous souhaitez que le poste de travail d'agent tolérant aux pannes fonctionne en mode *statut intégral*. Cela signifie que le poste de travail est mis à jour avec l'état des travaux et flots de travaux s'exécutant sur tous les postes de travail de son domaine et dans les domaines subordonnés, mais pas dans les domaines homologues ou parents. La copie du plan de production sur l'agent est conservée avec le même niveau de détails que la copie du plan de production sur son gestionnaire de domaine.

off Si vous souhaitez que le poste de travail d'agent tolérant aux pannes soit informé de l'état des travaux et flots de travaux uniquement sur les postes de travail qui affectent ses propres travaux et flots de travaux. La valeur "on" peut améliorer les performances en réduisant l'activité du réseau.

securitylevel
Indique le type d'authentification SSL pour le poste de travail. Ne spécifiez pas cet attribut pour un poste de travail de type `broker`. Il peut prendre l'une des valeurs suivantes :

enabled
Le poste de travail utilise l'authentification uniquement si le poste de travail de son gestionnaire de domaine ou un autre agent tolérant aux pannes situé en-dessous dans la hiérarchie des domaines l'exige.

on Le poste de travail utilise l'authentification SSL lorsqu'il se connecte à son gestionnaire de domaine. Le gestionnaire de domaine utilise une authentification SSL lorsqu'il établit une connexion avec le gestionnaire de domaine parent. L'agent tolérant aux pannes refuse toute connexion entrante autre que SSL à partir de son gestionnaire de domaine.

force Le poste de travail utilise une authentification SSL pour toutes ses connexions et accepte les connexions de ses gestionnaires de domaine parent et subordonnés. Il refuse toute connexion entrante non-SSL.

Si cet attribut fait défaut, le poste de travail n'est pas configuré pour les connexions SSL et toute valeur de **secureaddr** est ignorée. Dans ce cas, assurez-vous que l'option locale *nm ssl port* est définie sur 0 pour garantir que le processus **netman** ne tente pas d'ouvrir le port spécifié dans **secureaddr**. Voir *IBM Tivoli Workload Scheduler - Guide d'administration* pour plus d'informations sur l'authentification SSL.

Le tableau suivant décrit le type de communication utilisé pour chaque type de configuration **securitylevel**.

Tableau 26. Type de communication en fonction de la valeur securitylevel

Valeur définie sur l'agent tolérant aux pannes (ou le gestionnaire de domaine)	Valeur définie sur son gestionnaire de domaine (ou sur son gestionnaire de domaine parent)	Type de connexion établie
Non spécifié	Non spécifié	TCP/IP
Activé	Non spécifié	TCP/IP
Actif	Non spécifié	Pas de connexion
Force	Non spécifié	Pas de connexion
Non spécifié	Actif	TCP/IP
Activé	Actif	TCP/IP
Actif	Actif	SSL
Force	Actif	SSL
Non spécifié	Activé	TCP/IP
Activé	Activé	TCP/IP
Actif	Activé	SSL
Force	Activé	SSL
Non spécifié	Force	Pas de connexion
Activé	Force	SSL
Actif	Force	SSL
Force	Force	SSL

La valeur de **securitylevel** n'est pas spécifiée pour les postes de travail courtiers de charge de travail dynamiques ou pour les postes de travail comportant l'agent Tivoli Workload Scheduler version 8.2 ou antérieure.

server *ID_serveur*

Utilisez l'attribut **server** dans la définition de poste de travail d'agent tolérant aux pannes pour réduire le temps requis pour initialiser des agents et améliorer l'opportunité des messages. Par défaut, les communications avec les agent tolérant aux pannes sont gérés par un processus **mailman** exécuté sur le gestionnaire de domaine. L'attribut **server** vous permet de démarrer le processus **mailman** sur le gestionnaire de domaine afin de traiter les communications avec ce poste de travail d'agent tolérant aux pannes uniquement.

Si vous définissez un agent tolérant aux pannes qui peut fonctionner en tant que gestionnaire de domaine de secours, l'*ID_serveur* est utilisé uniquement lorsque le poste de travail fonctionne en tant qu'agent tolérant aux pannes ; le paramètre est ignoré lorsque le poste de travail fonctionne en tant que gestionnaire de domaine de secours.

Dans *ID_serveur*, l'ID correspond à une seule lettre ou à un chiffre (A-Z et 0-9). Les ID sont uniques à chaque gestionnaire de domaine, de sorte qu'il est possible d'utiliser les mêmes ID dans d'autres domaines sans créer de conflit. Un *ID_serveur* spécifique peut être dédié à plusieurs postes de travail d'agent tolérant aux pannes.

Les valeurs recommandées sont les suivantes :

- Si vous avez besoin de plus de 36 ID serveur dans un domaine, pensez à diviser ce dernier en deux domaines ou plus.

- Si le même ID est utilisé pour plusieurs agents, un seul serveur est créé pour gérer leurs communications. Vous devez définir des serveurs supplémentaires afin d'éviter qu'un seul serveur gère plus de huit agents.

Si aucun *ID_serveur* n'est indiqué, les communications avec l'agent sont traitées par le processus **mailman** du gestionnaire de domaine.

protocole *http | https*
Indique le protocole à utiliser pour communiquer avec :

Le poste de travail courtier
Si le poste de travail est un poste de travail d'agent.

Le moteur distant
Si le poste de travail est un poste de travail de moteur distant.

members *[poste de travail] [...]*
Utilisez cette valeur pour un poste de travail de pool afin de spécifier les agents que vous souhaitez ajouter au pool.

requirements *définition_jsdl*
Utilisez cette valeur pour un poste de travail de pool dynamique afin de spécifier les exigences, au format .JSDL, que les agents doivent satisfaire pour appartenir automatiquement au pool dynamique. Vous utilisez la syntaxe suivante :

```
jsdl_definition:
<jsdl:resources>
<jsdl:logicalResource subType="MyResourceType"/>
</jsdl:resources>
```

Pour plus d'informations sur la syntaxe JSDL, voir *Scheduling Workload Dynamically*.

Remarque : Vous pouvez ajouter des définitions de poste de travail à la base de données à tout moment mais vous devez réexécuter **JnextPlan -for 0000** afin de pouvoir exécuter des travaux sur les postes de travail récemment créés. Chaque fois que vous exécutez **JnextPlan**, tous les postes de travail sont arrêtés et redémarrés.

Exemples

Dans l'exemple suivant, vous allez créer un gestionnaire de domaine maître baptisé `hdq-1` et un agent tolérant aux pannes baptisé `hdq-2` dans le gestionnaire de domaine. Notez que l'argument **domain** est facultatif dans cet exemple, car la valeur par défaut du domaine maître est **masterdm**.

```
cpuname hdq-1 description "Headquarters master DM"
     os unix
     tz America/Los_Angeles
     node sultan.ibm.com
     domain masterdm
     for maestro type manager
          autolink on
          fullstatus on
end

cpuname hdq-2
     os wnt
     tz America/Los_Angeles
     node opera.ibm.com
```

```
     domain masterdm
     for maestro type fta
          autolink on
end
```

L'exemple suivant illustre la création d'un domaine nommé `distr-a` avec un gestionnaire de domaine nommé `distr-a1` et un agent standard nommé `distr-a2` :

```
domain distr-a
     manager distr-a1
     parent masterdm
end

cpuname
distr-a1 description "District A domain mgr"
     os wnt
     tz America/New_York
     node pewter.ibm.com
     domain distr-a
     for maestro type manager
          autolink on
          fullstatus on
end

cpuname distr-a2
     os wnt
     node quatro.ibm.com
     tz America/New_York
     domain distr-a
     for maestro type s-agent
end
```

L'exemple suivant crée un poste de travail tolérant aux pannes avec l'authentification SSL. La définition de sécurité **securitylevel** indique que la connexion entre le poste de travail et son gestionnaire de domaine peut être uniquement de type SSL. Le port 32222 est réservé pour la détection des connexions SSL entrantes.

```
cpuname ENNETI3
     os WNT
     node apollo
     tcpaddr 30112
     secureaddr 32222
     for maestro
          autolink off
          fullstatus on
          securitylevel on
end
```

Dans l'exemple suivant, un poste de travail de courtier est créé. Ce poste de travail gère le cycle de vie des travaux de type Tivoli Workload Scheduler Workload Broker dans Tivoli Dynamic Workload Broker.

```
cpuname ITDWBAGENT
       vartable TABLE1
       os OTHER
      node itdwbtst11.ibm.com TCPADDR 41114
      timezone Europe/Rome
      domain MASTERDM
      for MAESTRO
          type BROKER
          autolink OFF
          behindfirewall OFF
          fullstatus OFF
end
```

L'exemple suivant crée un poste de travail de moteur distant à utiliser pour gérer les dépendances croisées et communiquer avec un moteur distant installé sur un système de nom d'hôte London-hdq utilisant le port HTTPS 31116 par défaut. Le poste de travail de moteur distant est hébergé par le poste de travail courtier ITDWBAGENT

```
cpuname REW_London
     description "Remote engine workstation to communicate with London-hdq"
     os WNT
     node London-hdq secureaddr 31116
     timezone Europe/London
     for maestro host ITDWBAGENT
        type rem-eng
        protocol HTTPS
end
```

L'exemple suivant montre comment créer un pool dynamique d'agents. Tous les agents du pool dynamique doivent disposer du système d'exploitation HP-UX ou Linux installé :

```
CPUNAME DPOOLUNIX
  DESCRIPTION "Sample Dynamic Pool Workstation"
  VARTABLE table1
  OS OTHER
  TIMEZONE Europe/Rome
  FOR MAESTRO HOST MAS86MAS_DWB
    TYPE D-POOL
  REQUIREMENTS
    <?xml version="1.0" encoding="UTF-8"?>
<jsdl:resourceRequirements
      xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl">
  <jsdl:resources>
    <jsdl:candidateOperatingSystems>
      <jsdl:operatingSystem type="HPUX"/>
      <jsdl:operatingSystem type="LINUX"/>
    </jsdl:candidateOperatingSystems>
  </jsdl:resources>
</jsdl:resourceRequirements>
FIN
```

L'exemple suivant montre comment créer un pool dynamique d'agents. Sur tous les agents dans le pool dynamique, le système d'exploitation Windows 2000 doit être installé :

```
CPUNAME DPOOLWIN
  DESCRIPTION "Sample Dynamic Pool Workstation"
  OS WNT
  TIMEZONE Europe/Rome
  FOR MAESTRO HOST MAS86MAS_DWB
    TYPE D-POOL
  REQUIREMENTS
    <?xml version="1.0" encoding="UTF-8"?>
<jsdl:resourceRequirements
      xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl">
  <jsdl:resources>
    <jsdl:candidateOperatingSystems>
      <jsdl:operatingSystem type="Windows 2000"/>
    </jsdl:candidateOperatingSystems>
  </jsdl:resources>
</jsdl:resourceRequirements>
FIN
```

L'exemple suivant montre comment créer un pool d'agents nommé POOLUNIX et contenant deux agents : NC121105 et NC117248 :

```
CPUNAME POOLUNIX
  DESCRIPTION "Sample Pool Workstation"
  OS OTHER
  TIMEZONE Europe/Rome
  FOR MAESTRO HOST MAS86MAS_DWB
    TYPE POOL
  MEMBERS
    NC121105
    NC117248
FIN
```

Voir aussi

Pour savoir comment effectuer la même tâche à partir de Dynamic Workload Console, voir :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Création de postes de travail distribués".

Définition de classe de postes de travail

Une classe de postes de travail correspond à un groupe de postes de travail pour lequel des travaux et des flots de travaux communs peuvent être écrits. Vous pouvez faire figurer plusieurs définitions de classes de travail dans un même fichier texte, avec des définitions de postes de travail et des définitions de domaines.

Lors de la définition des classes de poste de travail, assurez-vous que les postes de travail dans la classe prennent en charge les types de travail que vous prévoyez d'exécuter dessus. Les règles suivantes s'appliquent :

- Les types de travail avec options avancées s'exécutent uniquement sur les agents dynamiques, les pools et les pools dynamiques.
- Les travaux reflet s'exécutent uniquement sur les moteurs distants.

Chaque définition de classe de postes de travail utilise le format et les arguments suivants :

Syntaxe

cpuclass *classe_poste_de_travail* [**description** *"description"*]
  [**ignore**]
  members [*poste_de_travail* | **@**] [...]
  end

[**cpuname** ...]

[**cpuclass** ...]

[**domain** ...]

Arguments

cpuclass *classes_poste_de_travail*
: Nom de la classe de postes de travail. Il doit commencer par une lettre et peut contenir des caractères alphanumériques, des tirets et des traits de soulignement. Il peut comporter jusqu'à 16 caractères.

Remarque : Vous ne pouvez pas utiliser des noms identiques pour des postes de travail, des classes de postes de travail et des domaines.

description *"description"*
Fournit une description de la classe de postes de travail. La description peut contenir jusqu'à 120 caractères alphanumériques. Ce texte doit être délimité par des guillemets.

ignore Indique que Tivoli Workload Scheduler doit ignorer tous les postes de travail appartenant à cette classe lors de la génération du plan de production.

members *poste_de_travail*
Présente une liste de noms de poste de travail, séparés par des espaces, membres de la classe. Le caractère générique @ signifie que la classe de postes de travail inclut tous les postes de travail.

Exemples

Dans l'exemple suivant, nous définissons une classe de postes de travail appelée `backup` :

```
cpuclass backup
     members
          main
          site1
          site2
end
```

Dans l'exemple suivant, nous définissons une classe de postes de travail appelée `allcpus`, qui contient chaque poste de travail :

```
cpuclass allcpus
     members
          @
end
```

Voir aussi

Pour plus d'informations sur la façon d'effectuer la même tâche à partir de Dynamic Workload Console, voir :

la section "Conception de votre charge de travail" dans le document Dynamic Workload Console - Guide d'utilisation.

Définition de domaine

Un domaine correspond à un groupe de postes de travail comportant un ou plusieurs agents ainsi qu'un gestionnaire de domaine. Le gestionnaire de domaine se comporte de la même manière que le concentrateur de gestion pour les agents dans le domaine. Vous pouvez faire figurer plusieurs définitions de domaines dans un même fichier texte, avec des définitions de postes de travail et des définitions de classes de postes de travail. Chaque définition de domaine utilise le format et les arguments suivants :

Syntaxe

domain *nom_domaine*[**description** *"description"*]
  ***** **manager** *poste_de_travail*
    [**parent** *nom_domaine* **|** **ismaster**]
end

[**cpuname** ...]

[**cpuclass** ...]

[**domain** ...]

Arguments

domain *nom_domaine*
: Nom du domaine. Ce nom doit commencer par une lettre et peut contenir des caractères alphanumériques, des tirets et des traits de soulignement. Il peut comporter jusqu'à 16 caractères. Vous ne pouvez pas utiliser des noms identiques pour des postes de travail, des classes de postes de travail et des domaines.

description "*description***"**
: Fournit une description du domaine. Ce texte doit être délimité par des guillemets.

*** manager** *poste_de_travail*
: Il s'agit d'une zone commentée utilisée uniquement pour montrer, lors de l'affichage de la définition du domaine, le nom du poste de travail qui occupe le rôle de gestionnaire de domaine pour ce domaine. Assurez-vous que cette zone reste commentée. Elle est conservée à des fins de compatibilité avec les versions antérieures. Avec la Tivoli Workload Scheduler version 8.3, l'information indiquant si un poste de travail est un gestionnaire de domaine est spécifiée dans la zone **type** de «Définition de poste de travail», à la page 149.

parent *nom_domaine*
: Nom du domaine parent auquel le gestionnaire de domaine est connecté. Il s'agit par défaut du domaine maître, qui ne requiert pas de définition de domaine. Il est défini par les options globales **master** et **master domain**.

ismaster
: S'il est défini, indique que le domaine est le domaine supérieur du réseau Tivoli Workload Scheduler. S'il est défini, il ne peut pas être supprimé ultérieurement.

Exemples

Dans l'exemple suivant, nous définissons un domaine appelé `east` utilisant le domaine maître comme parent et deux domaines subordonnés baptisés `northeast` et `southeast` :

```
domain east description "The Eastern domain"
     * manager cyclops
end
domain northeast description "The Northeastern domain"
     * manager boxcar
     parent east
end
```

```
domain southeast description "The Southeastern domain"
     * manager sedan
     parent east
end
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle décrite dans :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Création d'un domaine".

Définition de travail

Un travail correspond à une commande, à un fichier exécutable ou à un programme planifié et exécuté par Tivoli Workload Scheduler. Vous pouvez écrire des définitions de travaux dans des fichiers d'édition, puis les ajouter à la base de données Tivoli Workload Scheduler à l'aide du programme Composer. Vous pouvez faire figurer plusieurs définitions de travaux dans un même fichier d'édition.

Chaque définition de travail utilise le format et les arguments suivants :

Syntaxe

$jobs
[*poste_de_travail***#**]*nom_travail*
    {**scriptname** *file_name* **streamlogon** *nom_utilisateur* |
      docommand *"commande"* **streamlogon** *nom_utilisateur* |
      task *définition_travail* [**streamlogon** *nom_utilisateur*]}
    [**description** *"description"*]
    [**tasktype** *type_tâche*]
    [**interactive**]
    [**rccondsucc** *"Condition de succès"*]
    [**recovery**
        {**stop** | **continue** | **rerun**}
        [**after** [*poste_de_travail***#**]*nom_travail*]
        [**abendprompt** *"texte"*]]

Dans la mesure où un travail proprement dit ne comporte pas de paramètres pour les dépendances, ces paramètres doivent être ajoutés au travail lorsqu'il est inclus dans une définition de flot de travaux.

Vous pouvez ajouter ou modifier des définitions de travail à partir des définitions de flot de travaux. Les modifications apportées aux définitions de travail dans les définitions de flot de travaux sont prises en compte dans les définitions de travail stockées dans la base de données. En d'autres termes, si vous modifiez la définition de travail de `job1` dans la définition de flot de travaux `js1` et si `job1` est également utilisé dans le flot de travaux `js2`, la définition de `job1` dans la définition de `js2` est modifiée en conséquence.

Remarque : Les mots clés saisis incorrectement dans les définitions de travail entraînent le stockage de définitions de travaux tronquées dans la base de données. En fait, le mot clé erroné est considéré comme externe à la définition de travail et il est interprété comme le nom de travail d'une définition de travail supplémentaire.

Généralement, cette mauvaise interprétation entraîne également une erreur de syntaxe ou une erreur de définition de travail inexistant pour la définition de travail supplémentaire.

Soyez vigilant dans le cas où un alias a été affecté à un travail. Vous pouvez décider d'utiliser un nom différent pour faire référence à une instance de travail particulière au sein d'un flot de travaux, mais le nom de l'alias ne doit pas entrer en conflit avec le nom d'un autre travail dans le même flot de travaux. Si une définition de travail est renommée, les travaux portant le même nom modifient leur nom conformément au nom de la définition de travail. Les exemples ci-dessous vous permettront de comprendre le comportement des travaux lorsque le nom de la définition de travail est modifié :

Tableau 27. Exemples : changement de nom de la définition de travail

Noms de définition de travail initiaux dans le flot de travaux	**Changement de nom de la définition de travail**	**Résultat**
SCHEDULE WKS#JS : FTA1#**A** FTA1#**B as C** END	Renommez le travail **A** en **D**	SCHEDULE WKS#JS : FTA1#**D** FTA1#**B as C** END
	Renommez le travail **B** en **D**	SCHEDULE WKS#JS : FTA1#**A** FTA1#**D as C** END
	Renommez le travail **A** en **C**	Une erreur se produit lors du changement de nom du travail **A** en **C** car le travail C existe déjà comme alias du travail **B**.

Pour savoir comment écrire des définitions de flot de travaux, voir «Définition de flot de travaux», à la page 236.

Arguments

*poste_de_travail***#**
: Indique le nom du poste de travail ou de la classe de postes de travail sur lequel le travail est exécuté. La valeur par défaut est le poste de travail spécifié pour *defaultws* lors du démarrage de la session de **composer**.

  Pour plus d'informations sur le démarrage d'une session de composer, voir «Exécution du programme Composer», à la page 303. Le signe dièse (**#**) est un délimiteur obligatoire. Si une classe de postes de travail est indiquée, celle-ci doit correspondre à la classe de postes de travail d'un flot de travaux contenant le travail.

  Si vous définissez un travail qui gère un travail Workload Broker, spécifiez le nom du poste de travail sur lequel le poste de travail Workload Broker est installé. Avec le poste de travail Workload Broker, Tivoli Workload Scheduler peut soumettre un travail dans l'environnement Tivoli Dynamic Workload Broker grâce à la soumission de travail dynamique.

nom_travail
: Indique le nom du travail. Il doit commencer par une lettre et peut

contenir des caractères alphanumériques, des tirets et des traits de soulignement. Il peut contenir jusqu'à 40 caractères.

scriptname *file_name*

Indique le nom du fichier exécuté par le travail. Utilisez le mot clé **scriptname** pour exécuter les travaux UNIX et Windows. Pour un fichier exécutable, entrez le nom du fichier et les éventuels arguments et options. La longueur du *file_name* additionnée à la longueur de la *Condition de succès* (du mot clé **rccondsucc**) ne doit pas dépasser 4095 caractères. Vous pouvez également utiliser des paramètres Tivoli Workload Scheduler.

Voir «Utilisation de variables et de paramètres dans les définitions de travail», à la page 210 pour plus d'informations.

Pour les travaux Windows, inclut les extensions de fichier. Les noms UNC sont admis. Ne spécifiez pas de fichiers résidant sur des unités mappées.

Si vous définissez un travail qui gère un travail Workload Broker, spécifiez le nom du travail Workload Broker. De plus, vous pouvez spécifier des variables et le type d'affinité existant entre le travail Tivoli Workload Scheduler et le travail Workload Broker à l'aide de la syntaxe indiquée dans la liste ci-dessous. Pour identifier un travail avec affinité à l'aide :

du nom de travail Tivoli Workload Scheduler
jobName [-var var1Name=var1Value,...,varNName=varNValue] [-twsAffinity jobname=twsJobName]

de l'ID du travail Tivoli Dynamic Workload Broker
jobName [-var var1Name=var1Value,...,varNName=varNValue] [-affinity jobid=jobid]

de l'alias du travail Tivoli Dynamic Workload Broker
jobName [-var var1Name=var1Value,...,varNName=varNValue] [-affinity alias=alias]

Pour plus d'informations, voir *IBM Tivoli Workload Scheduler : Planification dynamique de la charge de travail*.

Si le chemin d'accès du fichier ou le nom du fichier de l'argument **scriptname** contient des espaces, la chaîne toute entière doit être placée entre "\" et \" " comme illustré ci-dessous :

```
scriptname "\"C:\Program Files\tws\myscript.cmd\""
```

Si des caractères spéciaux sont inclus, autres que des barres obliques (/) et des barres obliques inversées (\), la chaîne toute entière doit être placée entre guillemets (").

Le travail `échoue` si le script spécifié dans l'option **scriptname** est introuvable ou ne possède pas de droit d'exécution. Il `s'interrompt` si le script qui est introuvable ou ne possède pas de droit d'exécution comprend des paramètres.

docommand *commande*

Indique une commande exécutée par le travail. Entrez une commande valide et les éventuels arguments et options placés entre guillemets ("). La longueur de la *commande* additionnée à la longueur de la *Condition de succès* (du mot clé **rccondsucc**) ne doit pas dépasser 4095 caractères. Vous pouvez également utiliser des paramètres Tivoli Workload Scheduler.

Le travail `s'interrompt` si le fichier spécifié avec l'option **docommand** est introuvable ou ne possède pas de droit d'exécution.

Voir «Utilisation de variables et de paramètres dans les définitions de travail», à la page 210 pour plus d'informations.

task *définition_travail*
Spécifie la syntaxe XML des types de travaux avec options avancées et des travaux reflet. Pour définir les types de travaux existants, utilisez la commande **docommand**. Ce mot-clé s'applique uniquement aux postes de travail du type suivant :

- agent
- pool
- d-pool
- rem-eng

La syntaxe du travail dépend du travail que vous définissez.

Pour obtenir la liste complète des types de travail pris en charge, voir «Création de types de travaux avec options avancées», à la page 518.

streamlogon *nom_utilisateur*

Nom d'utilisateur sous lequel le travail s'exécute. Cet attribut est obligatoire lorsque **scriptname** ou **docommand** est spécifié. Ce nom peut comporter jusqu'à 47 caractères. Si ce nom comporte des caractères spéciaux, il doit être placé entre guillemets ("). Indiquez un utilisateur qui peut se connecter au poste de travail sur lequel le travail s'exécute. Vous pouvez également utiliser des paramètres Tivoli Workload Scheduler.

Voir «Utilisation de variables et de paramètres dans les définitions de travail», à la page 210 pour plus d'informations.

Pour les travaux Windows, l'utilisateur doit également posséder une définition d'utilisateur.

Pour obtenir des informations relatives aux besoins de l'utilisateur, voir «Définition d'utilisateur», à la page 211.

Si vous définissez un travail qui gère un travail de courtier de charge de travail dynamique, spécifiez le nom de l'utilisateur que vous avez utilisé pour installer Tivoli Dynamic Workload Broker.

Le travail `échoue` si l'utilisateur spécifié dans l'option **streamologon** n'existe pas.

description "*description***"**
Fournit une description du travail. Ce texte doit être délimité par des guillemets.Le nombre maximum de caractères autorisés est 120.

tasktype *type_tâche*

Indique le type de travail. Il peut prendre l'une des valeurs suivantes :

UNIX Pour les travaux qui s'exécutent sur les plateformes UNIX.

WINDOWS
Pour les travaux qui s'exécutent sur les systèmes d'exploitation Windows.

OTHER
Pour les travaux qui s'exécutent sur des agents étendus. Voir *IBM Tivoli Workload Scheduler for Applications - Guide d'utilisation* pour plus d'informations sur les types de tâche personnalisées pour les applications prises en charge par le fournisseur.

COURTIER
Pour les travaux qui gèrent le cycle de vie d'un travail Tivoli Dynamic Workload Broker. Reportez-vous à *IBM Tivoli Workload Scheduler : Planification dynamique de la charge de travail* pour des informations sur l'utilisation de Tivoli Dynamic Workload Broker.

interactive

Si vous définissez un travail qui gère un travail Tivoli Dynamic Workload Broker, ignorez cet argument. Spécifie que le travail s'exécute de façon interactive sur votre bureau. Cette fonction est disponible uniquement dans les environnementsWindows.

rccondsucc "*Condition de succès***"**
Expression qui détermine le code retour (RC) requis pour considérer qu'un travail est réussi. La condition de réussite peut contenir au maximum 256 caractères. Cette expression peut être l'une des suivantes :

COMPLETE_IF_BIND_FAILS
Ce paramètre s'applique à un travail reflet uniquement. Lorsqu'il est indiqué, l'état du travail reflet est automatiquement défini par SUCC si la liaison au travail distant échoue.

Expression de comparaison
Indique les codes retour des travaux. La syntaxe est la suivante :

```
(RC opérateur opérande)
```

RC Mot clé RC.

opérateur
Opérateur de comparaison. Il peut prendre l'une des valeurs suivantes :

Tableau 28. Opérateurs de comparaison

Exemple	Opérateur	Description
RC<a	<	Inférieur à
RC<=a	<=	Inférieur ou égal à
RC>a	>	Supérieur à
RC>=a	>=	Supérieur ou égal à
RC=a	=	Egal à
RC!=a	!=	Différent de
RC<>a	<>	Différent de

opérande
Un entier entre -2147483647 et 2147483647.

Par exemple, vous pouvez définir un travail réussi comme un travail qui prend fin avec un code retour inférieur ou égal à 3. Exemple :

```
rccondsucc "(RC <= 3)"
```

expression booléenne
Indique une combinaison logique d'expressions de comparaison. La syntaxe est la suivante :

```
expression_de_comparaison opérateur expression_de_comparaison
```

expression_de_comparaison
Cette expression est analysée de gauche à droite. Vous pouvez utiliser des parenthèses pour attribuer une priorité à l'évaluation de l'expression.

opérateur
Opérateur logique. Il peut prendre l'une des valeurs suivantes :

Tableau 29. Opérateurs logiques

Exemple	Opérateur	Résultat
expr_a and expr_b	And	TRUE si expr_a ET expr_b sont TRUE.
expr_a or expr_b	Or	TRUE si expr_a OU expr_b est TRUE.
Not expr_a	Not	TRUE si expr_a n'est PAS TRUE.

Par exemple, vous pouvez définir un travail réussi comme un travail qui termine par un code retour inférieur ou égal à 3 ou par un code retour différent de 5 et inférieur à 10. Exemple :

```
rccondsucc "(RC≤3) OR ((RC≠5) AND (RC<10))"
```

recovery

Options de reprise du travail. La valeur par défaut est **stop** sans aucun travail de reprise et aucune invite de reprise. Entrez l'une des options de reprise suivantes : **stop**, **continue** ou **rerun**. Elle peut être suivie d'un travail de reprise et/ou d'une invite de reprise.

stop En cas de fin anormale du travail, le traitement ne se poursuit pas avec le travail suivant.

continue
En cas de fin anormale du travail, le traitement se poursuit avec le travail suivant. Le travail n'est pas répertorié comme interrompu dans les propriétés du flot de travail. Si aucun autre problème ne survient, le flot de travaux aboutit.

rerun En cas de fin anormale du travail, le traitement est réexécuté.

after [*poste_de_travail***#]***nom_travail*

Indique le nom d'un travail de reprise à lancer si le travail parent prend fin de façon anormale. Les travaux de reprise ne sont lancés qu'une seule fois pour chaque instance du travail parent qui prend fin de façon anormale.

Vous pouvez indiquer le poste du travail de reprise s'il est différent du poste du travail parent. Par défaut, il s'agit du poste du travail parent. Tous les travaux ne remplissent pas les conditions nécessaires pour permettre l'exécution de travaux de reprise sur un poste de travail différent. Suivez les instructions ci-après :

- Si l'un des postes de travail est un agent étendu, il doit être hébergé par un gestionnaire de domaine ou un agent tolérant aux pannes pour lequel la valeur du mode **fullstatus** a été définie sur **on**.
- Le poste de travail du travail de reprise peut se trouver dans le même domaine que le poste de travail du travail parent ou dans un domaine supérieur.

- Si le poste de travail du travail de reprise est un agent tolérant aux pannes, la valeur **fullstatus** de cet agent doit être définie sur **on**.

abendprompt "*texte***"**
Indique le texte d'une invite de reprise, figurant entre guillemets, qui doit être affiché dans le cas d'une fin anormale du travail. 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 processus. Si le texte commence par un point d'exclamation (!), l'invite s'affiche, mais il n'est pas enregistré dans le fichier journal. Vous pouvez également utiliser des paramètres Tivoli Workload Scheduler.

Pour plus d'informations, voir «Utilisation de variables et de paramètres dans les définitions de travail», à la page 210.

Le tableau 30 résume toutes les combinaisons possibles d'options de reprise et d'actions.

Ce tableau est basé sur les critères suivants d'un flot de travaux appelé `sked1` :

- Le flot de travaux `sked1` a deux travaux : `job1` et `job2`.
- S'il est sélectionné pour `job1`, le travail de reprise est `jobr`.
- `job2` dépend de `job1` et ne démarre pas avant que `job1` ne soit terminé.

Tableau 30. Options et actions de reprise

	Stop	**Continue**	**Rerun**
Invite de reprise : Non **Travail de reprise** : Non	Une intervention est requise.	Exécutez `job2`.	Réexécutez `job1`. En cas de fin anormale de `job1`, émettez une invite. Si la réponse est *oui*, répétez ce qui précède. Si `job1` réussit, exécutez `job2`.
Invite de reprise : Oui **Travail de reprise** : Non	Emettez une invite de reprise. Une intervention est requise.	Emettez une invite de reprise. Si la réponse est *oui*, exécutez `job2`.	Emettez une invite de reprise. Si la réponse est *oui*, exécutez de nouveau `job1`. En cas de fin anormale de `job1`, répétez ce qui précède. Si `job1` réussit, exécutez `job2`.
Invite de reprise : Non **Travail de reprise** : Oui	Exécutez `jobr`. Si celui-ci se termine de façon anormale, une intervention est requise. S'il s'exécute avec succès, exécutez `job2`.	Exécutez `jobr`. Exécutez `job2`.	Exécutez `jobr`. Si `jobr` se termine de façon anormale, une intervention est requise. Si `jobr` réussit, réexécutez `job1`. En cas de fin anormale de `job1`, émettez une invite. Si la réponse est *oui*, répétez ce qui précède. Si `job1` réussit, exécutez `job2`.

Tableau 30. Options et actions de reprise (suite)

	Stop	**Continue**	**Rerun**
Invite de reprise : Oui **Travail de reprise** : Oui	Emettez une invite de reprise. Si la réponse est *oui*, exécutez `jobr`. Si celui-ci se termine de façon anormale, une intervention est requise. S'il s'exécute avec succès, exécutez `job2`.	Emettez une invite de reprise. Si la réponse est *oui*, exécutez `jobr`. Exécutez `job2`.	Emettez une invite de reprise. Si la réponse est *oui*, exécutez `jobr`. Si `jobr` se termine de façon anormale, une intervention est requise. Si `jobr` réussit, réexécutez `job1`. En cas de fin anormale de `job1`, répétez ce qui précède. Si `job1` réussit, exécutez `job2`.

Remarques :

1. L'expression "une intervention est requise" signifie que `job2` n'est pas libéré de sa dépendance par rapport à `job1`, et que c'est à l'utilisateur de le faire.
2. L'option de reprise **continue** se substitue à l'état de fin anormale, ce qui peut faire apparaître le flot de travaux contenant le travail en échec comme ayant réussi. Cela empêche évite le report du flot de travaux au plan de production suivant.
3. Si vous sélectionnez l'option **rerun** sans indiquer d'invite de reprise, Tivoli Workload Scheduler génère sa propre invite.
4. Pour référencer un travail de reprise dans **conman**, utilisez le nom du travail d'origine (`job1` dans le scénario ci-dessus et non `jobr`). Un seul travail de reprise est exécuté par fin anormale.

Exemples

L'exemple suivant présente un fichier comportant deux définitions de travail :

```
$jobs
cpu1#gl1
     scriptname "/usr/acct/scripts/gl1"
     streamlogon acct
     description "general ledger job1"
bkup
     scriptname "/usr/mis/scripts/bkup"
     streamlogon "^mis^"
     recovery continue after recjob1
```

L'exemple ci-dessous illustre la définition du travail Tivoli Workload Scheduler TWSJOB qui gère le travail Workload Broker broker_1 s'exécutant sur le même agent Workload Broker que TWSJOB2 :

```
ITDWBAGENT#TWSJOB
SCRIPTNAME "broker_1 -var var1=name,var2=address
           -twsaffinity jobname=TWSJOB2"
STREAMLOGON brkuser
DESCRIPTION "Added by composer."
TASKTYPE BROKER
RECOVERY STOP
```

L'exemple suivant montre comment définir un travail qui est affecté à un pool dynamique d'agents UNIX et exécute le script `df` :

```
DPOOLUNIX#JOBDEF7
 TASK
    <?xml version="1.0" encoding="UTF-8"?>
    <jsdl:jobDefinition
          xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
          xmlns:jsdle="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdle">
    <jsdl:application name="executable">
    <jsdle:executable interactive="false">
    <jsdle:script>df</jsdle:script>
    </jsdle:executable>
    </jsdl:application>
    </jsdl:jobDefinition>
 DESCRIPTION "Added by composer."
 RECOVERY STOP
```

L'exemple suivant montre comment définir un travail qui est affecté à un pool dynamique d'agents Windows et exécute le script `dir` :

```
DPOOLWIN#JOBDEF6
 TASK
    <?xml version="1.0" encoding="UTF-8"?>
    <jsdl:jobDefinition
          xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
          xmlns:jsdle="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdle">
    <jsdl:application name="executable">
    <jsdle:executable interactive="false">
    <jsdle:script>dir</jsdle:script>
    </jsdle:executable>
    </jsdl:application>
    </jsdl:jobDefinition>
 DESCRIPTION "Added by composer."
 RECOVERY STOP
```

L'exemple suivant montre comment définir un travail qui est affecté aux agent NC115084 et exécute le script `dir` :

```
NC115084#JOBDEF3
 TASK
    <?xml version="1.0" encoding="UTF-8"?>
    <jsdl:jobDefinition
          xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
          xmlns:jsdle="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdle">
    <jsdl:application name="executable">
    <jsdle:executable interactive="false">
    <jsdle:script>dir</jsdle:script>
    </jsdle:executable>
    </jsdl:application>
    </jsdl:jobDefinition>
 DESCRIPTION "Added by composer."
 RECOVERY STOP
```

L'exemple suivant montre comment définir un travail qui est affecté à un pool d'agents UNIX et exécute le script défini dans la balise `script` :

```
POOLUNIX#JOBDEF5
 TASK
    <?xml version="1.0" encoding="UTF-8"?>
    <jsdl:jobDefinition
          xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
          xmlns:jsdle="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdle">
    <jsdl:application name="executable">
    <jsdle:executable interactive="false">
    <jsdle:script>#!/bin/sh
sleep 60
dir</jsdle:script>
    </jsdle:executable>
```

```
   </jsdl:application>
   </jsdl:jobDefinition>
 DESCRIPTION "Added by composer."
 RECOVERY STOP
```

L'exemple suivant montre comment définir un travail qui est affecté à un pool d'agents Windows et exécute le script défini dans la balise `script` :

```
POOLWIN#JOBDEF4
 TASK
   <?xml version="1.0" encoding="UTF-8"?>
   <jsdl:jobDefinition
         xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
         xmlns:jsdle="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdle">
   <jsdl:application name="executable">
   <jsdle:executable interactive="false">
   <jsdle:script>ping -n 120 localhost</jsdle:script>
   </jsdle:executable>
   </jsdl:application>
   </jsdl:jobDefinition>
 DESCRIPTION "Added by composer."
 RECOVERY STOP
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle décrite dans :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Création de définitions de travail".

Pour plus d'informations sur la création et l'édition d'objets de planification, voir :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Conception de votre charge de travail".

Définition de travail - Travaux reflet

Les travaux reflet sont définis à l'aide de la syntaxe XML. Les attributs de clé permettant d'identifier l'instance de travail distant et les critères de correspondance dépendent du type de moteur distant sur lequel l'instance de travail distant est définie. Les zones en gras sont celles utilisées pour identifier l'instance de travail distant.

Etant donné que les moteurs z/OS prennent uniquement en charge les critères de correspondance précédents les plus proches, le modèle XML permettant de définir un travail reflet z/OS est le suivant :

```
$JOBS
WORKSTATION#ZSHADOW_CLOS_PRES
 TASK
   <?xml version="1.0" encoding="UTF-8"?>
   <jsdl:jobDefinition
      xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
      xmlns:zshadow="http://www.ibm.com/xmlns/prod/scheduling/1.0/zshadow">
   <jsdl:application name="zShadowJob">
   <zshadow:ZShadowJob>
   <zshadow:JobStream>JobStream</zshadow:JobStream>
   <zshadow:JobNumber>JobNumber</zshadow:JobNumber>
   <zshadow:matching>
   <zshadow:previous/>
   </zshadow:matching>
   </zshadow:ZShadowJob>
```

```
   </jsdl:application>
   </jsdl:jobDefinition>
DESCRIPTION "Sample Job Definition"
RECOVERY STOP
```

Remarque : Assurez-vous d'entrer des paramètres valides dans les zones **JobStream** et **JobNumber**.

Les travaux reflet distribués, en revanche, prennent en charge les quatre critères de correspondance disponibles pour les dépendances de prédécesseur/successeur externes. Vous trouverez ci-dessous les modèles XML que vous pouvez utiliser pour définir les travaux reflet distribués :

Critères de correspondance : précédents les plus proches

Modèle XML :

```
$JOBS
WORKSTATION#DSHADOW_CLOS_PRES
 TASK
    <?xml version="1.0" encoding="UTF-8"?>
    <jsdl:jobDefinition
       xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
       xmlns:dshadow="http://www.ibm.com/xmlns/prod/scheduling/1.0/dshadow">
    <jsdl:application name="distributedShadowJob">
    <dshadow:DistributedShadowJob>
    <dshadow:JobStream>JobStream</dshadow:JobStream>
    <dshadow:Workstation>Workstation</dshadow:Workstation>
    <dshadow:Job>Job</dshadow:Job>
    <dshadow:matching>
    <dshadow:previous/>
    </dshadow:matching>
    </dshadow:DistributedShadowJob>
    </jsdl:application>
    </jsdl:jobDefinition>
 DESCRIPTION "Sample Job Definition"
 RECOVERY STOP
```

Critères de correspondance : dans un intervalle absolu

Modèle XML :

```
$JOBS
WORKSTATION#DSHADOW_ABSOLUTE
 TASK
    <?xml version="1.0" encoding="UTF-8"?>
    <jsdl:jobDefinition
          xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
          xmlns:dshadow="http://www.ibm.com/xmlns/prod/scheduling/1.0/dshadow">
    <jsdl:application name="distributedShadowJob">
    <dshadow:DistributedShadowJob>
    <dshadow:JobStream>JobStream</dshadow:JobStream>
    <dshadow:Workstation>Workstation</dshadow:Workstation>
    <dshadow:Job>Job</dshadow:Job>
    <dshadow:matching>
    <dshadow:absolute from="0600 -4" to="1100 +3"/>
    </dshadow:matching>
    </dshadow:DistributedShadowJob>
    </jsdl:application>
    </jsdl:jobDefinition>
 DESCRIPTION "Sample Job Definition"
 RECOVERY STOP
```

Critères de correspondance : dans un intervalle relatif

```
$JOBS
WORKSTATION#DSHADOW_RELATIVE
 TASK
    <?xml version="1.0" encoding="UTF-8"?>
    <jsdl:jobDefinition
          xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
          xmlns:dshadow="http://www.ibm.com/xmlns/prod/scheduling/1.0/dshadow">
    <jsdl:application name="distributedShadowJob">
    <dshadow:DistributedShadowJob>
    <dshadow:JobStream>JobStream</dshadow:JobStream>
    <dshadow:Workstation>Workstation</dshadow:Workstation>
    <dshadow:Job>Job</dshadow:Job>
    <dshadow:matching>
```

```
   <dshadow:relative from="-400" to="+500" />
   </dshadow:matching>
   </dshadow:DistributedShadowJob>
   </jsdl:application>
   </jsdl:jobDefinition>
 DESCRIPTION "Sample Job Definition"
 RECOVERY STOP
```

Critères de correspondance : même date planifiée

Modèle XML :

```
$JOBS
WORKSTATION#DSHADOW_SAMEDAY
 TASK
   <?xml version="1.0" encoding="UTF-8"?>
   <jsdl:jobDefinition
         xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
         xmlns:dshadow="http://www.ibm.com/xmlns/prod/scheduling/1.0/dshadow">
   <jsdl:application name="distributedShadowJob">
   <dshadow:DistributedShadowJob>
   <dshadow:JobStream>JobStream</dshadow:JobStream>
   <dshadow:Workstation>Workstation</dshadow:Workstation>
   <dshadow:Job>Job</dshadow:Job>
   <dshadow:matching>
   <dshadow:sameDay/>
   </dshadow:matching>
   </dshadow:DistributedShadowJob>
   </jsdl:application>
   </jsdl:jobDefinition>
 DESCRIPTION "Sample Job Definition"
 RECOVERY STOP
```

Pour plus d'informations sur les critères de correspondance, voir «Gestion des dépendances externes de prédécesseur/successeur des travaux et des flots de travaux», à la page 65.

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle décrite dans :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Création de définitions de travail".

Pour plus d'informations sur la création et l'édition d'objets de planification, voir :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Conception de votre charge de travail".

Définition de travail - Travaux de services Web

Cette section décrit les attributs obligatoires et facultatifs pour les travaux de services Web. Les travaux de services Web ne fonctionnent qu'avec les paramètres d'entrée et de sortie qui utilisent des types de données simples. Chaque définition de travail utilise le format et les arguments suivants :

Tableau 31. Attributs obligatoires et facultatifs pour la définition d'un travail de services Web

Attribut	Description/valeur	Obligatoire
nom d'application	ws	✓
opération	Le nom de la commande de services Web que vous appelez.	✓
wsdlURL	Nom de l'URL de service Web.	✓
arguments	Les paramètres requis par la commande de services Web que vous appelez (le nombre de valeurs dépend de la commande).	

Tableau 31. Attributs obligatoires et facultatifs pour la définition d'un travail de services Web (suite)

Attribut	Description/valeur	Obligatoire

droits d'accès	Le nom et le mot de passe de l'utilisateur qui exécute ce travail. Comme alternative au codage en dur des valeurs réelles, vous pouvez paramétrer de l'une des manières suivantes :
	• Entrez un *nom_utilisateur* spécifié dans la base de données avec la définition de l'utilisateur (applicable à tous les systèmes d'exploitation sur ce type de travail) et saisissez l'instruction suivante :
	`<jsdl:password>${password:nom_utilisateur}</jsdl:password>`
	Le mot de passe est récupéré dans la définition de l'utilisateur *nom_utilisateur* dans la base de données et résolu au cours de l'exécution. Pour plus d'informations, voir «Utilisation des définitions d'utilisateur sur des types de travaux avec des options avancées», à la page 214.
	Vous pouvez également spécifier l'utilisateur d'un poste de travail différent et utiliser la syntaxe suivante pour le mot de passe :
	`<jsdl:password>${password:poste_travail#nom_utilisateur}</jsdl:password>`
	• Saisissez un utilisateur et un mot de passe définis avec la commande `param` localement sur l'agent dynamique qui va exécuter le travail (si le travail consiste à être soumis à un pool ou à un pool dynamique, la définition doit être présente sur tous les agents du pool). A condition que vous ayez défini le nom d'utilisateur avec la variable *utilisateur* et un mot de passe, les instructions de données d'identification se présenteront ainsi :
	`<jsdl:userName>${agent:utilisateur}</jsdl:userName>`
	`<jsdl:password>${agent:password.utilisateur}</jsdl:password>`
	Les variables de l'utilisateur et du mot de passe sont résolues sur l'agent au moment de l'exécution. Pour plus d'informations, voir «Définition des variables et mots de passe pour la résolution locale sur des agents dynamiques», à la page 521.
	Si vous utilisez une connexion HTTPS, assurez-vous que les certificats de sécurité sont configurés pour le gestionnaire de travaux sur le poste de travail sur lequel le travail doit s'exécuter.	

Le résultat de la commande est enregistré dans le journal de travail.

L'exemple suivant présente une tâche qui exécute une commande de services Web `getSum`. La définition de tâche fournit dans la section `arguments` les deux valeurs qui doivent être ajoutées.

```
$JOBS
AGENT#WEB_SERVICE
TASK
<?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
xmlns:jsdlws="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdlws" description="Calls a web
service to perform a sum of two numbers" name="SumNumber">
<jsdl:annotation>annotation</jsdl:annotation>
<jsdl:variables>
<jsdl:stringVariable description="URL of Web Service"
name="wsdlURL">http://np515516.cyber.com:9080/
```

```
Sum/services/Sum/wsdl/Sum.wsdl</jsdl:stringVariable>
<jsdl:stringVariable description="Operation to Invoke"
name="Operation">getSum</jsdl:stringVariable>
</jsdl:variables>
<jsdl:application name="ws">
<jsdlws:ws>
<jsdlws:wsToInvoke operation="${Operation}" wsdlURL="${wsdlURL}">
<jsdlws:arguments>
<jsdlws:value>1</jsdlws:value>
<jsdlws:value>2</jsdlws:value>
</jsdlws:arguments>
<jsdlws:credentials>
<jsdl:userName>administrator</jsdl:userName>
<jsdl:password>mdp</jsdl:password>
</jsdlws:credentials>
</jsdlws:wsToInvoke>
</jsdlws:ws>
</jsdl:application>
<jsdl:resources>
<jsdl:candidateHosts>
<jsdl:hostName>${host}</jsdl:hostName>
</jsdl:candidateHosts>
</jsdl:resources>
</jsdl:jobDefinition>
```

L'exemple suivant s'applique si vous utilisez une connexion HTTPS avec l'agent exécutant la tâche de services Web. Il montre comment configurer la clé `JVMOptions` dans le fichier `jobManager.ini` de l'agent afin de pointer vers les certificats de sécurité.

```
JVMOptions=-Djavax.net.ssl.keyStore=/images/ITAuser/TWA/TWS/JavaExt/cfg/agentKeystore.jks
-Djavax.net.ssl.keyStorePassword=tdwb8nxt
-Djavax.net.ssl.trustStore=/images/ITAuser/TWA/TWS/JavaExt/cfg/agentKeystore.jks
-Djavax.net.ssl.trustStorePassword=tdwb8nxt
-Djavax.net.ssl.trustStoreType=JKS
```

Par défaut, le délai d'attente des services Web est de 90 secondes. Si le service Web ne s'est pas bien exécuté dans cette période, le travail Tivoli Workload Scheduler associé se termine par une erreur. Pour éviter l'échec des travaux lorsque l'exécution des services Web prend plus de 90 secondes, vous pouvez personnaliser le délai d'attente.

Le délai d'attente est spécifié dans le fichier `WSJobExecutor.properties` situé dans le répertoire `TWA_HOME/TWS/JavaExt/cfg`. Ce délai est structuré comme suit :

```
# Web Service timeout
# Specify the timeout in seconds
# Default Value is 90 seconds
TIMEOUT=90
```

Par exemple, pour un service Web dont l'exécution prend 5 minutes, personnalisez le fichier de configuration en indiquant : `TIMEOUT=300`.

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle décrite dans :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Création de définitions de travail".

Pour plus d'informations sur la création et l'édition d'objets de planification, voir :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Conception de votre charge de travail".

Définition de travail - Travaux de transfert de fichier

Cette section décrit les attributs obligatoires et les attributs facultatifs des travaux de transfert de fichier. Seuls les transferts de fichier unique sont autorisés. Chaque définition de travail utilise le format et les arguments suivants :

Tableau 32. Attributs obligatoires et facultatifs pour la définition d'un travail de transfert de fichier

Attribut	Description/valeur	Obligatoire
nom d'application	filetransfer	✓
Type de transfert de fichier (importation ou téléchargement)	Enserrez tous les attributs de transfert de fichier de balises `jsdlfiletransfer:uploadInfo` ou `jsdlfiletransfer:downloadInfo` comme indiqué dans l'exemple.	✓
serveur	L'adresse du serveur de transfert de fichier (et du numéro de port, s'il est différent du port standard, lorsque vous choisissez FTP comme protocole).	✓
localfile et remotefile	• Si vous effectuez une importation, `localfile` est le nom et le chemin qualifié complet du fichier à importer, tandis que `remotefile` est le nom et le chemin qualifié complet du fichier à créer sur la cible distante. • Si vous effectuez un téléchargement, `localfile` est le nom et le chemin qualifié complet du fichier à créer sur la cible locale, tandis que `remotefile` est le nom et le chemin qualifié complet du fichier à télécharger. • Les caractères génériques ne sont pas pris en charge.	✓

localCredentials et remoteCredentials	Noms et mots de passe des utilisateurs autorisés sur les systèmes locaux et distants. Comme alternative au codage en dur des valeurs réelles, vous pouvez paramétrer de l'une des manières suivantes :
	• Entrez un *nom_utilisateur* spécifié dans la base de données avec la définition de l'utilisateur (applicable à tous les systèmes d'exploitation sur ce type de travail) et saisissez l'instruction suivante :
	`<jsdl:password>${password:nom_utilisateur}</jsdl:password>`
	Le mot de passe est récupéré dans la définition de l'utilisateur *nom_utilisateur* dans la base de données et résolu au cours de l'exécution. Pour plus d'informations, voir «Utilisation des définitions d'utilisateur sur des types de travaux avec des options avancées», à la page 214.
	Vous pouvez également spécifier l'utilisateur d'un poste de travail différent et utiliser la syntaxe suivante pour le mot de passe :
	`<jsdl:password>${password:poste_travail#nom_utilisateur}</jsdl:password>`
	• Saisissez un utilisateur et un mot de passe définis avec la commande `param` localement sur l'agent dynamique qui va exécuter le travail (si le travail consiste à être soumis à un pool ou à un pool dynamique, la définition doit être présente sur tous les agents du pool). A condition que vous ayez défini le nom d'utilisateur avec la variable *utilisateur* et un mot de passe, les instructions de données d'identification se présenteront ainsi :
	`<jsdl:userName>${agent:utilisateur}</jsdl:userName>`
	`<jsdl:password>${agent:password.utilisateur}</jsdl:password>`
	Les variables de l'utilisateur et du mot de passe sont résolues sur l'agent au moment de l'exécution. Pour plus d'informations, voir «Définition des variables et mots de passe pour la résolution locale sur des agents dynamiques», à la page 521.
		✓

Tableau 32. Attributs obligatoires et facultatifs pour la définition d'un travail de transfert de fichier (suite)

Attribut	Description/valeur	Obligatoire
protocole	peut être : **WINDOWS** Protocole de partage de fichiers de Microsoft. Si vous ne spécifiez pas de protocole et que SSH ne fonctionne pas, le système suppose WINDOWS. Spécifiez le répertoire partagé dans le mot clé `remotefile`, sans indiquer de chemin où est imbriqué le répertoire partagé. Indiquez l'adresse du poste de travail hébergeant le répertoire partagé dans le mot clé `server`. Le répertoire de travail par défaut est : `C:\Program Files\IBM\TWA\TWS\ITA`. **SSH** Un protocole réseau qui fournit des fonctions d'accès au fichier, de transfert de fichier et de gestion de fichier sur un flot de données fiable. Si vous ne précisez pas de protocole, SSH est supposé par défaut. **FTP** Un protocole réseau standard utilisé pour échanger des fichiers sur un réseau basé sur TCP/IP, tel qu'Internet. **FTPS** Extension du protocole FTP qui ajoute la prise en charge du protocole cryptographique TLS. Plus précisément, le transfert de fichier s'effectue implicitement à l'aide du protocole TLS pour les sessions FTP, fournissant une clé de sécurité privée pour la connexion de données. La version 1 du protocole TLS est prise en charge. La configuration de réutilisation de session SSL n'est pas prise en charge. **FTPES** Extension du protocole FTP qui ajoute la prise en charge du protocole cryptographique TLS. Plus précisément, le transfert de fichier s'effectue explicitement à l'aide du protocole TLS pour les sessions FTP fournissant une clé de sécurité privée pour la connexion de données. La version 1 du protocole TLS est prise en charge. La configuration de réutilisation de session SSL n'est pas prise en charge.	
transferMode	Peut être `binary` (valeur par défaut) ou `ascii`.	

remoteCodepage	Page de codes utilisée sur le poste de travail distant. Si vous souhaitez utiliser une page de codes personnalisée, définissez le paramètre `remoteCodepage` comme suit :
	`<jsdlfiletransfer:remoteCodepage>USER:`*`CUSTOM_CP`*`</jsdlfiletransfer:remoteCodepage>`
	Où *CUSTOM_CP* est la page de codes définie par l'utilisateur.
	Par exemple, pour utiliser la page de codes personnalisée `tcpip.ftpd.ftpxlbin.frence3`, définissez le paramètre `remoteCodepage` comme suit :
	`<jsdlfiletransfer:remoteCodepage>USER:tcpip.ftpd.ftpxlbin.frence3 </jsdlfiletransfer:remoteCodepage>`
		Obligatoire si vous spécifiez localCodepage

localCodepage	Page de codes utilisée sur le poste de travail local.	Obligatoire si vous spécifiez remoteCodepage
Délai d'attente	Indique le nombre de secondes à utiliser pour l'opération de transfert de fichier. La valeur par défaut est 60 secondes.	

Tableau 32. Attributs obligatoires et facultatifs pour la définition d'un travail de transfert de fichier (suite)

Attribut	Description/valeur	Obligatoire

portsRange	Lorsque le mode actif est activé, la section `portsRange` restreint les numéros de port envoyés par la commande FTP PORT. Cette option accepte des règles de pare-feu très strictes. La section `portsRange` définit la plage de ports à utiliser côté client des connexions de données TCP. Si vous ne spécifiez pas la section `portsRange`, le système d'exploitation détermine les numéros de port à utiliser.
	
	Le paramètre `portsRange` requiert les paramètres suivants :
	
	min *(port_min)*
	La valeur minimale du port à utiliser côté client des connexions de données TCP. Les valeurs admises sont comprises entre 0 et 65535. Par exemple, si vous définissez cette valeur à 1035, Tivoli Workload Scheduler restreint les numéros de port au port 1035 et supérieur
	
	max *(port_max)*
	La valeur maximale du port à utiliser côté client des connexions de données TCP. Les valeurs admises sont comprises entre 0 et 65535. Par exemple, si vous définissez cette valeur à 1038, Tivoli Workload Scheduler restreint les numéros de port au port 1038 et inférieur
	
	Par exemple, pour limiter la plage de ports à utiliser côté client entre le port 1035 et le port 1040, indiquez :
	`<jsdlfiletransfer:portsRange>`
	`<jsdlfiletransfer:min>1035</jsdlfiletransfer:min>`
	`<jsdlfiletransfer:max>1040</jsdlfiletransfer:max>`
	`</jsdlfiletransfer:portsRange>`	

passiveMode	Indique si le client est passif ou actif dans l'établissement des connexions pour les transferts de données. Si vous définissez cette option sur `NO`, le serveur établit la connexion de données avec le client (mode actif). Si vous définissez cette option sur `YES`, le client établit la connexion de données avec le serveur (mode passif). La valeur par défaut est `NO`.	

Le fichier xml suivant présente la section “application” JSDL d'un exemple de définition de travail pour un type de travail de transfert de fichier :

```
<jsdl:application name="filetransfer">
<jsdlfiletransfer:filetransfer>
<jsdlfiletransfer:downloadInfo>
<jsdlfiletransfer:server>FTP_SERVER</jsdlfiletransfer:server>
<jsdlfiletransfer:localfile>LOCAL_FILE</jsdlfiletransfer:localfile>
<jsdlfiletransfer:remotefile>REMOTE_FILE</jsdlfiletransfer:remotefile>
<jsdlfiletransfer:remoteCredentials>
<jsdl:userName>USERNAME</jsdl:userName>
<jsdl:password>PASSWORD</jsdl:password>
</jsdlfiletransfer:remoteCredentials>
<jsdlfiletransfer:protocol>PROTOCOL</jsdlfiletransfer:protocol>
<jsdlfiletransfer:transferMode>ASCII_BINARY</jsdlfiletransfer:transferMode>
<jsdlfiletransfer:codepageConversion>
<jsdlfiletransfer:remoteCodepage>RM_CP</jsdlfiletransfer:remoteCodepage>
<jsdlfiletransfer:localCodepage>LC_CP</jsdlfiletransfer:localCodepage>
</jsdlfiletransfer:codepageConversion>
<jsdlfiletransfer:timeout>CONNECTION_TIMEOUT</jsdlfiletransfer:timeout>
<jsdlfiletransfer:portsRange>
<jsdlfiletransfer:min>MIN_PORT</jsdlfiletransfer:min>
<jsdlfiletransfer:max>MAX_PORT</jsdlfiletransfer:max>
</jsdlfiletransfer:portsRange>
<jsdlfiletransfer:passiveMode>YES_NO</jsdlfiletransfer:passiveMode>
</jsdlfiletransfer:downloadInfo>
</jsdlfiletransfer:filetransfer>
</jsdl:application>
```

L'exemple suivant montre une tâche généralisée qui télécharge un fichier à partir d'un poste de travail distant dont l'adresse est 10.0.0.8 :

```
$JOBS
AGENT#FILE_TRANSFER
TASK
<?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
 xmlns:jsdlfiletransfer="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdlfiletransfer"
name="FILETRANSFER">
<jsdl:application name="filetransfer">
<jsdlfiletransfer:filetransfer>
<jsdlfiletransfer:downloadInfo>
<jsdlfiletransfer:server>10.0.0.8</jsdlfiletransfer:server>
<jsdlfiletransfer:localfile>c:\MyTextFile.txt</jsdlfiletransfer:localfile>
<jsdlfiletransfer:remotefile>./MyRemoteFile.txt</jsdlfiletransfer:remotefile>
<jsdlfiletransfer:localCredentials>
<jsdl:userName>${agent:locluser}</jsdl:userName>
<jsdl:password>${agent:password.${agent:locluser}}</jsdl:password>[1]
</jsdlfiletransfer:localCredentials>
<jsdlfiletransfer:remoteCredentials>
<jsdl:userName>remuser</jsdl:userName>
<jsdl:password>${password:remuser}</jsdl:password>[2]
</jsdlfiletransfer:remoteCredentials>
<jsdlfiletransfer:protocol>FTP</jsdlfiletransfer:protocol>
<jsdlfiletransfer:transferMode>ascii</jsdlfiletransfer:transferMode>
<jsdlfiletransfer:codepageConversion>
<jsdlfiletransfer:remoteCodepage>IBM-280</jsdlfiletransfer:remoteCodepage>
<jsdlfiletransfer:localCodepage>ISO8859-1</jsdlfiletransfer:localCodepage>
</jsdlfiletransfer:codepageConversion>
</jsdlfiletransfer:downloadInfo>
</jsdlfiletransfer:filetransfer>
</jsdl:application>
</jsdl:jobDefinition>
```

Remarque : dans les sections de droit d'accès,

1. le nom d'utilisateur local a été défini sur l'agent qui exécute le travail avec une variable `locluser` par le biais de la commande `param`. La valeur définie pour `locluser` sera récupérée au moment de l'exécution auprès du fichier de variables situé sur l'agent. De même, le mot de passe pour la valeur représentée par `locluser` a été défini sur l'agent à l'aide de la commande `param` et sera résolu au moment de l'exécution à partir du fichier de variables local.
2. Le nom d'utilisateur distant a été défini avec la commande `composer user` et est stocké dans la base de données avec son mot de passe sous le nom d'utilisateur `remuser`. Le mot de passe pour `remuser` sera extrait de la base de données au moment de l'exécution.

L'exemple suivant présente une définition de travail à utiliser pour télécharger un fichier texte à l'aide du protocole FTPES avec mode actif, délai d'attente de 10 secondes et plage de ports à utiliser comprise entre between 1035 et 1038 :

```
<?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
xmlns:jsdlfiletransfer="http://www.ibm.com/xmlns/prod/scheduling/1.0/
jsdlfiletransfer" name="FTPES_DOWNLOAD_TEXT">
<jsdl:application name="filetransfer">
<jsdlfiletransfer:filetransfer>
<jsdlfiletransfer:downloadInfo>
<jsdlfiletransfer:server>myServerFtp</jsdlfiletransfer:server>
<jsdlfiletransfer:localfile>d:\MyLocalFile.txt</jsdlfiletransfer:localfile>
<jsdlfiletransfer:remotefile>/tmp/MyRemoteFile.txt</jsdlfiletransfer:remotefile>
<jsdlfiletransfer:remoteCredentials>
<jsdl:userName>myUser</jsdl:userName>
<jsdl:password>myPassword</jsdl:password>
</jsdlfiletransfer:remoteCredentials>
<jsdlfiletransfer:protocol>FTPES</jsdlfiletransfer:protocol>
<jsdlfiletransfer:transferMode>ascii</jsdlfiletransfer:transferMode>
<jsdlfiletransfer:timeout>10</jsdlfiletransfer:timeout>
<jsdlfiletransfer:portsRange>
<jsdlfiletransfer:min>1035</jsdlfiletransfer:min>
<jsdlfiletransfer:max>1038</jsdlfiletransfer:max>
</jsdlfiletransfer:portsRange>
```

```
</jsdlfiletransfer:downloadInfo>
</jsdlfiletransfer:filetransfer>
</jsdl:application>
</jsdl:jobDefinition>
```

L'exemple suivant présente une définition de travail à utiliser pour transférer un fichier texte à l'aide du protocole SSH :

```
<?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
xmlns:jsdlfiletransfer="http://www.ibm.com/xmlns/prod/scheduling/1.0/
jsdlfiletransfer" name="SSH_UPLOAD">
<jsdl:application name="filetransfer">
<jsdlfiletransfer:filetransfer>
<jsdlfiletransfer:uploadInfo>
<jsdlfiletransfer:server>myServer</jsdlfiletransfer:server>
<jsdlfiletransfer:localfile>d:\MyLocalFile.txt</jsdlfiletransfer:localfile>
<jsdlfiletransfer:remotefile>/tmp/MyRemoteFile.txt</jsdlfiletransfer:remotefile>
<jsdlfiletransfer:remoteCredentials>
<jsdl:userName>myUser</jsdl:userName>
<jsdl:password>myPassword</jsdl:password>
</jsdlfiletransfer:remoteCredentials>
<jsdlfiletransfer:protocol>SSH</jsdlfiletransfer:protocol>
<jsdlfiletransfer:transferMode>ascii</jsdlfiletransfer:transferMode>
</jsdlfiletransfer:uploadInfo>
</jsdlfiletransfer:filetransfer>
</jsdl:application>
</jsdl:jobDefinition>
```

L'exemple suivant présente une définition de travail à utiliser pour transférer un fichier texte à l'aide du protocole WINDOWS :

```
<?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
xmlns:jsdlfiletransfer="http://www.ibm.com/xmlns/prod/scheduling/1.0/
jsdlfiletransfer" name="WINDOWS_DOWNLOAD">
<jsdl:application name="filetransfer">
<jsdlfiletransfer:filetransfer>
<jsdlfiletransfer:downloadInfo>
<jsdlfiletransfer:server>myServer</jsdlfiletransfer:server>
<jsdlfiletransfer:localfile>d:\MyLocalFile.txt</jsdlfiletransfer:localfile>
<jsdlfiletransfer:remotefile>mySharedFolder\MyRemoteFile.txt
</jsdlfiletransfer:remotefile>
<jsdlfiletransfer:remoteCredentials>
<jsdl:userName>myUser</jsdl:userName>
<jsdl:password>myPassword</jsdl:password>
</jsdlfiletransfer:remoteCredentials>
<jsdlfiletransfer:protocol>WINDOWS</jsdlfiletransfer:protocol>
<jsdlfiletransfer:transferMode>ascii</jsdlfiletransfer:transferMode>
</jsdlfiletransfer:downloadInfo>
</jsdlfiletransfer:filetransfer>
</jsdl:application>
</jsdl:jobDefinition>
```

Le type de travail de transfert de fichier fournit les codes retour suivants. Les codes retour sont disponibles uniquement pour les travaux terminés.

RC=0 Transfert de fichier terminé correctement.

RC=-1 Transfert de fichier non effectué. Le travail a échoué avec le code d'erreur suivant :

```
AWKFTE007E
```

Explication : une erreur s'est produite pendant l'opération de transfert de fichier.

Raisons possibles : fichier distant introuvable ou droits d'accès refusés.

RC=-2 Transfert de fichier non effectué. Le travail a échoué avec le code d'erreur suivant :

```
AWKFTE020E
```

Explication : uniquement pour les protocoles SSH ou WINDOWS. Une erreur a été renvoyée pendant la tentative de conversion de la page de codes.

Raisons possibles : pour les protocoles SSH ou WINDOWS, la page de codes est automatiquement détectée et convertie. Dans ce cas, il existe une erreur dans la page de codes du fichier à transférer qui ne respecte pas la page de codes du système local.

RC=-3 Transfert de fichier non effectué. Le travail a échoué avec le code d'erreur suivant :

```
WKFTE015E
```

Explication : une erreur s'est produite pendant l'opération de transfert de fichier.

Raisons possibles : fichier local introuvable.

RC=-4 Transfert de fichier effectué avec la page de codes par défaut. Le travail a échoué avec le code d'erreur suivant :

```
AWKFTE023E
```

Explication : la conversion de la page de codes spécifiée n'a pas été effectuée. Le transfert de fichier a été effectué avec la page de codes par défaut.

Raisons possible : la page de codes spécifiée est indisponible.

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle décrite dans :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Création de définitions de travail".

Pour plus d'informations sur la création et l'édition d'objets de planification, voir :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Conception de votre charge de travail".

Définition de travail - Travaux de type TPM (Provisioning)

Vous devez disposer d'IBM SmartCloud Provisioning version 2.1 pour pouvoir utiliser le type de travail TPM (Provisioning) dans Tivoli Workload Scheduler.

Pour lire un scénario commun basé sur la réalité qui atteint les objectifs métier, y compris l'implémentation d'un travail de type TPM (Provisioning), voir http://pic.dhe.ibm.com/infocenter/tivihelp/v47r1/index.jsp?topic=/com.ibm.tivoli.itws.doc_9.2/scen/awsbsscp_rc.html.

Avant de définir un travail TPM (Provisioning), vous devez avoir stocké le certificat de serveur HTTP TPM (Provisioning) et l'avoir défini en ayant suivi la procédure décrite dans "Etapes prérequises pour créer des travaux d'application des accès" dans le manuel *Tivoli Workload Automation : Dynamic Workload Console - Guide d'utilisation*.

La présente section décrit les attributs obligatoires et facultatifs des travaux de type TPM (Provisioning). Chaque définition de travail utilise le format et les arguments suivants :

Tableau 33. Attributs obligatoires et facultatifs pour la définition d'un travail de type TPM (Provisioning)

Attribut	Description/valeur	Obligatoire
application name	application des accès	✓
actionType	Valeurs valides : **deploy** Déployez une image virtuelle dans le groupe de clouds et créez une nouvelle instance de système virtuel qui contient le nombre d'instances d'images virtuelles que vous spécifiez. **manage** Démarrez, arrêtez ou supprimez des ressources virtuelles (par exemple, des machines virtuelles ou des images virtuelles d'un environnement complexe).	✓
cloudGroupId	Identificateur unique du groupe Cloud à partir duquel vous choisissez l'instance virtuelle à déployer.	✓
instanceId	Identificateur unique de l'instance virtuelle existante sur laquelle vous agissez. Une instance virtuelle peut être une ou plusieurs machines virtuelles. Si vous voulez agir uniquement sur une seule machine virtuelle qui appartient à l'instance actuelle, indiquez son ID dans l'attribut **virtualMachineId**.	✓
virtualMachineId	Identificateur unique de la machine virtuelle existante sur laquelle vous agissez.	
VirtualImageId	Identificateur unique de l'image virtuelle existante qui est utilisée comme modèle pour une nouvelle image virtuelle que vous déployez.	✓
instanceNameDeploy	Nom unique de l'instance virtuelle.	✓
numberOfVirtualMachines	Nombre de machines virtuelles que vous déployez.	✓
description	Chaîne de texte qui décrit l'instance de système virtuel.	
tags	Matrice de balises définies par l'utilisateur. Ces données utilisateur indiquées ici peuvent être extraites à l'intérieur de la machine virtuelle. Elles peuvent être utilisées pour configurer une machine virtuelle au premier amorçage ou pour exécuter un script d'amorçage enregistré dans la machine virtuelle.	
Size	Taille de l'instance. Il s'agit de la taille de chaque machine virtuelle unique appartenant à l'instance. Valeurs valides : • xsmall : très petite • small : petite • medium : moyenne • large : grande • xlarge : très grande	✓
winPassword	Mot de passe administrateur permettant d'accéder aux système Windows déployés.	
unixSSHPublicKey	Applicable uniquement à UNIX. La clé SSH doit être fournie par l'administrateur TPM (Provisioning).	

Tableau 33. Attributs obligatoires et facultatifs pour la définition d'un travail de type TPM (Provisioning) (suite)

Attribut	Description/valeur	Obligatoire

userName
 	
password
	Données d'identification associées au serveur TPM (Provisioning). Comme alternative au codage en dur des valeurs réelles, vous pouvez paramétrer de l'une des manières suivantes :
	• Entrez un *nom_utilisateur* spécifié dans la base de données avec la définition de l'utilisateur (applicable à tous les systèmes d'exploitation sur ce type de travail) et saisissez l'instruction suivante :
	`<jsdl:password>${password:nom_utilisateur}</jsdl:password>`
	Le mot de passe est récupéré dans la définition de l'utilisateur *nom_utilisateur* dans la base de données et résolu au cours de l'exécution. Pour plus d'informations, voir «Utilisation des définitions d'utilisateur sur des types de travaux avec des options avancées», à la page 214.
	Vous pouvez également spécifier l'utilisateur d'un poste de travail différent et utiliser la syntaxe suivante pour le mot de passe :
	`<jsdl:password>${password:poste_travail#nom_utilisateur}`
	`</jsdl:password>`
	• Saisissez un utilisateur et un mot de passe définis avec la commande `param` localement sur l'agent dynamique qui va exécuter le travail (si le travail consiste à être soumis à un pool ou à un pool dynamique, la définition doit être présente sur tous les agents du pool). A condition que vous ayez défini le nom d'utilisateur avec la variable *utilisateur* et un mot de passe, les instructions de données d'identification se présenteront ainsi :
	`<jsdl:userName>${agent:user}</jsdl:userName>`
	`<jsdl:password>${agent:password.utilisateur}</jsdl:password>`
	Les variables de l'utilisateur et du mot de passe sont résolues sur l'agent au moment de l'exécution. Pour plus d'informations, voir «Définition des variables et mots de passe pour la résolution locale sur des agents dynamiques», à la page 521.
		✓

hostname	Nom d'hôte du serveur TPM (Provisioning).	✓
port	Numéro de port du serveur d'TPM (Provisioning). La valeur par défaut est `443`.	✓
manageType	Valeurs valides : • **actionStart** : démarre l'instance du système virtuel. • **actionStop** : arrête l'instance du système virtuel. • **actionDelete** : supprime l'instance du système virtuel.	✓

L'exemple suivant présente une définition de travail à utiliser pour déployer une machine virtuelle dans l'environnement TPM (Provisioning) :

```
<?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl" xmlns:
   jsdlprovisioning="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdlprovisioning" name="PROVISIONING">
  <jsdl:application name="provisioning">
    <jsdlprovisioning:provisioning>
   <jsdlprovisioning:ProvisioningParameters>
    <jsdlprovisioning:actions>
     <jsdlprovisioning:actionType>
      <jsdlprovisioning:deploy>
       <jsdlprovisioning:groupCloud>
        <jsdlprovisioning:cloudGroupId>8</jsdlprovisioning:cloudGroupId>
       </jsdlprovisioning:groupCloud>
       <jsdlprovisioning:groupVirtualImage>
        <jsdlprovisioning:virtualImageId>52</jsdlprovisioning:virtualImageId>
       </jsdlprovisioning:groupVirtualImage>
       <jsdlprovisioning:instanceNameDeploy>TESTPROP</jsdlprovisioning:instanceNameDeploy>
       <jsdlprovisioning:numberOfVirtualMachines>1</jsdlprovisioning:numberOfVirtualMachines>
       <jsdlprovisioning:description/>
       <jsdlprovisioning:tags/>
       <jsdlprovisioning:size>xsmall</jsdlprovisioning:size>
       <jsdlprovisioning:winPassword/>
       <jsdlprovisioning:unixSSHPublicKey/>
      </jsdlprovisioning:deploy>
     </jsdlprovisioning:actionType>
    </jsdlprovisioning:actions>
```

```
    <jsdlprovisioning:connectionInfo>
     <jsdlprovisioning:credentials>
      <jsdl:userName>cbadmin</jsdl:userName>
      <jsdl:password>{aes}2WfJH/3aOxyX2f+QXeW+1YnrN2tM4z338QMYlYVgpOA=</jsdl:password>
     </jsdlprovisioning:credentials>
     <jsdlprovisioning:server>
      <jsdlprovisioning:hostname>9.168.58.192</jsdlprovisioning:hostname>
      <jsdlprovisioning:port>443</jsdlprovisioning:port>
     </jsdlprovisioning:server>
    </jsdlprovisioning:connectionInfo>
   </jsdlprovisioning:ProvisioningParameters>
  </jsdlprovisioning:provisioning>
  </jsdl:application>
</jsdl:jobDefinition>
```

L'exemple suivant présente une définition de travail à utiliser pour arrêter une machine virtuelle :

```
<?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl" xmlns:
   jsdlprovisioning="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdlprovisioning" name="PROVISIONING">
  <jsdl:application name="provisioning">
    <jsdlprovisioning:provisioning>
   <jsdlprovisioning:ProvisioningParameters>
    <jsdlprovisioning:actions>
     <jsdlprovisioning:actionType>
      <jsdlprovisioning:manage>
       <jsdlprovisioning:manageType>actionStop</jsdlprovisioning:manageType>
       <jsdlprovisioning:instanceId>102</jsdlprovisioning:instanceId>
       <jsdlprovisioning:virtualMachineId/>
      </jsdlprovisioning:manage>
     </jsdlprovisioning:actionType>
    </jsdlprovisioning:actions>
    <jsdlprovisioning:connectionInfo>
     <jsdlprovisioning:credentials>
      <jsdl:userName>cbadmin</jsdl:userName>
      <jsdl:password>{aes}2WfJH/3aOxyX2f+QXeW+1YnrN2tM4z338QMYlYVgpOA=</jsdl:password>
     </jsdlprovisioning:credentials>
     <jsdlprovisioning:server>
      <jsdlprovisioning:hostname>9.168.58.192</jsdlprovisioning:hostname>
      <jsdlprovisioning:port>443</jsdlprovisioning:port>
     </jsdlprovisioning:server>
    </jsdlprovisioning:connectionInfo>
   </jsdlprovisioning:ProvisioningParameters>
  </jsdlprovisioning:provisioning>
  </jsdl:application>
</jsdl:jobDefinition>
```

Vous planifiez les travaux de type TPM (Provisioning) de Tivoli Workload Scheduler en les définissant dans les flots de travaux. Ajoutez le travail à un flot de travaux avec tous les arguments de planification nécessaires et soumettez-le. Les travaux peuvent être soumis dans un environnement réparti à l'aide de Dynamic Workload Console ou de la ligne de commande **conman**. Les travaux peuvent être soumis dans un environnement z/OS à l'aide de Dynamic Workload Console ou de l'application ISPF. Après la soumission, lorsque le travail est en cours d'exécution (statut **EXEC**), vous pouvez arrêter le travail à l'aide de la commande **kill** si nécessaire. Toutefois, cette action est efficace sur le travail Tivoli Workload Scheduler, mais n'affecte pas la commande lancée à distance. Après une action **kill**, Tivoli Workload Scheduler collecte le journal des travaux lorsque l'agent redémarre et affecte le statut **Error** ou **ABEND** au travail Tivoli Workload Scheduler, quel que soit le statut du travail TPM (Provisioning)

Si l'agent Tivoli Workload Scheduler devient indisponible lorsque vous soumettez le travail TPM (Provisioning) de Tivoli Workload Scheduler ou lorsque le travail est en cours d'exécution, si l'agent redevient indisponible, Tivoli Workload Scheduler lance la surveillance du travail là où il s'était arrêté.

Pour plus d'informations sur TPM (Provisioning), voir le centre de documentation d'IBM SmartCloud Provisioning

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle décrite dans :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Création de définitions de travail".

Pour plus d'informations sur la création et l'édition d'objets de planification, voir :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Conception de votre charge de travail".

Définition de travail - Travaux J2EE

Cette section décrit les attributs obligatoires et facultatifs pour les travaux de services Web. Chaque définition de travail utilise le format et les arguments suivants :

Tableau 34. Attributs obligatoires et facultatifs pour la définition d'un travail J2EE.

Attribut	Description/valeur	Obligatoire
nom d'application	j2ee	✔
opération jms	L'opération à effectuer. Les valeurs prises en charge sont : • **send**. Il s'agit de la valeur par défaut. • **receive**. Si vous indiquez **receive**, vous pouvez facultativement définir une valeur pour l'attribut **timeout**.	
délai d'attente	Le délai, exprimé en secondes, dans lequel la tâche doit s'effectuer. Si vous n'indiquez pas de délai ou que vous le définissez sur 0, la tâche se poursuit indéfiniment.	
connectionURL	L'URL de WebSphere Application Server.	
connFactory	Un objet administré qu'un client utilise pour créer une connexion au fournisseur JMS. Pour spécifier la fabrique de connexions, vous pouvez utiliser une expression de variable contenant une ou plusieurs références de variable, par exemple ${var}, ou un caractère, ou une chaîne.	✔
destination	Un objet administré qui encapsule l'identité d'une destination de message, qui est l'emplacement où les messages sont fournis et consommés. Pour spécifier la destination, vous pouvez utiliser une expression de variable pouvant contenir une ou plusieurs références de variable, telles que ${var}, tout caractère et toute chaîne.	✔
message	Le message à envoyer.	✔ **Remarque :** Cet attribut est obligatoire uniquement pour la tâche d'envoi.

Tableau 34. Attributs obligatoires et facultatifs pour la définition d'un travail J2EE. (suite)

Attribut	Description/valeur	Obligatoire

Droits d'accès	Spécifiez le nom d'utilisateur et le mot de passe à utiliser lors de l'exécution de l'application J2EE. Utilisez cette zone si la sécurité globale est activée sur WebSphere Application Server. L'utilisateur doit être défini sur WebSphere Application Server. Pour spécifier ces données d'identification, vous pouvez utiliser une expression de variable pouvant contenir une ou plusieurs références de variable telles que ${var}, que vous pouvez associer, la cas échéant, à tout caractère ou à une chaîne simple. De plus, vous pouvez paramétrer de l'une des manières suivantes :
	• Entrez un *nom d'utilisateur* spécifié dans la base de données avec la définition de l'utilisateur *nom d'utilisateur* (applicable à tous les systèmes d'exploitation sur ce type de travail) et saisissez l'instruction suivante :
	`<jsdl:password>${password:nom_utilisateur}</jsdl:password>`
	Le mot de passe est récupéré dans la définition de l'utilisateur *nom_utilisateur* dans la base de données et résolu au cours de l'exécution. Pour plus d'informations, voir «Utilisation des définitions d'utilisateur sur des types de travaux avec des options avancées», à la page 214.
	Vous pouvez également spécifier l'utilisateur d'un poste de travail différent et utiliser la syntaxe suivante pour le mot de passe :
	`<jsdl:password>${password:poste_travail#nom_utilisateur}</jsdl:password>`
	• Saisissez un utilisateur et un mot de passe définis avec la commande `param` localement sur l'agent dynamique qui va exécuter le travail (si le travail consiste à être soumis à un pool ou à un pool dynamique, la définition doit être présente sur tous les agents du pool). A condition que vous ayez défini le nom d'utilisateur avec la variable *utilisateur* et un mot de passe, les instructions de données d'identification se présenteront ainsi :
	`<jsdl:userName>${agent:utilisateur}</jsdl:userName>`
	`<jsdl:password>${agent:password.utilisateur}</jsdl:password>`
	Les variables de l'utilisateur et du mot de passe sont résolues sur l'agent au moment de l'exécution. Pour plus d'informations, voir «Définition des variables et mots de passe pour la résolution locale sur des agents dynamiques», à la page 521.	

L'exemple suivant présente une tâche send qui envoie un message dans la file d'attente MyQueue :

```
$JOBS
AGENT#JOB_NAME_JMS_SEND
TASK
<?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
 xmlns:jsdlj="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdlj" name="JMS_JOB_SEND">
<jsdl:application name="j2ee">
<jsdlj:j2ee>
<jsdlj:jms operation="send">
<jsdlj:connectionURL>corbaloc:iiop:washost.mydomain.com:2809</jsdlj:connectionURL>
<jsdlj:connFactory>jms/MyCF</jsdlj:connFactory>
<jsdlj:destination>jms/MyQueue</jsdlj:destination>
<jsdlj:message>Submission of jms job: SEND MESSAGE</jsdlj:message>
</jsdlj:jms>
<jsdlj:credentials>
<jsdlj:userName>jtwoeeuser</jsdlj:userName>
<jsdlj:password>${password:jtwoeeuser}</jsdlj:password>[1]
</jsdlj:credentials>
</jsdlj:j2ee>
</jsdl:application>
</jsdl:jobDefinition>
```

Remarque : (1) l'utilisateur `jtwoeeuser` a été défini dans la base de données Tivoli Workload Scheduler à l'aide de la commande de définition d'utilisateur *username*. Le mot de passe associé, spécifié par la chaîne `${password:jtwoeeuser}` dans la tâche, sera extrait de la base de données au cours de l'exécution.

L'exemple suivant présente une tâche qui lit des messages à partir de la file d'attente MyQueue :

```
$JOBS
AGENT#JOB_NAME_JMS_RECEIVE
TASK
<?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
 xmlns:jsdlj="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdlj" name="JMS_JOB_RECEIVE">
<jsdl:application name="j2ee">
<jsdlj:j2ee>
<jsdlj:jms operation="receive" timeout="180">
<jsdlj:connFactory>jms/MyCF</jsdlj:connFactory>
<jsdlj:destination>jms/MyQueue</jsdlj:destination>
</jsdlj:jms>
<jsdlj:credentials>
<jsdlj:userName>userName</jsdlj:userName>
<jsdlj:password>password</jsdlj:password>
</jsdlj:credentials>
</jsdlj:j2ee>
</jsdl:application>
</jsdl:jobDefinition>
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle décrite dans :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Création de définitions de travail".

Pour plus d'informations sur la création et l'édition d'objets de planification, voir :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Conception de votre charge de travail".

Définition de travail - Travaux de base de données

Cette section décrit les attributs obligatoires et facultatifs des travaux de base de données. Chaque définition de travail utilise le format et les arguments suivants :

Tableau 35. Attributs obligatoires et facultatifs pour la définition d'un travail de base de données.

Attribut	Description/valeur	Obligatoire
application name	Base de données	✓
sgbd	Type de base de données dans lequel vous souhaitez exécuter le travail. Les valeurs prises en charge sont : **db2** Pour les bases de données DB2 **mssql** Pour les bases de données Microsoft SQL Server **oracle** Pour les bases de données Oracle	✓
server	Nom d'hôte du serveur sur lequel la base de données est installée.	✓
port	Numéro de port pour le travail de base de données.	✓
database	Nom de la base de données.	✓
Nom de classe du pilote JDBC	Nom de classe du pilote JDBC	Obligatoire si vous spécifiez une base de données personnalisée.
Chaîne de connexion JDBC	Chaîne qui est utilisée pour la connexion à la base de données, contenant l'URL de base de données, le nom d'utilisateur et le mot de passe	Obligatoire si vous spécifiez une base de données personnalisée.

Tableau 35. Attributs obligatoires et facultatifs pour la définition d'un travail de base de données. (suite)

Attribut	Description/valeur	Obligatoire
JDBC jar class path	Chemin d'accès aux fichiers JAR du client de base de données. Cette valeur remplace la valeur indiquée dans le fichier de configuration DatabaseJobExecutor.properties, s'il existe. Si vous sélectionnez la base de données Microsoft SQL Server, la version 4 des pilotes JDBC est requise.	
dbStatement	Nom du travail de base de données à exécuter.	Obligatoire si vous spécifiez une instruction SQL

storedProcedure name	Nom de la procédure stockée dans des bases de données DB2, Oracle ou MSSQL.
	
	La procédure ne peut pas être stockée sur DB2 si la base de données contient déjà une ou plusieurs procédures stockées avec le même nom et le même schéma. Par exemple, si la base de données comporte plusieurs procédures stockées appelées `TEST.STORE_PROC1`, avec différents paramètres comme dans ce qui suit :
	`TEST.STORE_PROC1(VARCHAR,?)`
	`TEST.STORE_PROC1(VARCHAR,VARCHAR,?)`
	`TEST.STORE_PROC1(VARCHAR,?,?)`
	`TEST.STORE_PROC1(VARCHAR,VARCHAR,?,?)`
	
	le travail de base de données ne peut pas être créé et le message suivant est renvoyé : `AWKDBE033E Le nom de procédure stockée fourni correspond à plus d'une définition de procédure stockée dans la base de données. Afin de lever toute ambiguïté, indiquez aussi le schéma.`
		Obligatoire si vous spécifiez une procédure stockée.

ParameterTableValue key	Nom et valeurs de la procédure exprimés conformément à la syntaxe suivante :
	Type de variable de la procédure stockée
	Les valeurs prises en charge sont :
	• IN
	• OUT
	• INOUT
	Nom de la variable
	Nom de la variable tel que défini dans la procédure stockée.
	Type de variable
	Type de la variable. Les types SQL pris en charge sont les suivants :
	• DATE
	• DECIMAL
	• ENTIER
	• VARCHAR
	0...n Position de chaque variable telle que définie dans la procédure stockée.
	Par exemple :
	`Nom Valeur`
	`IN VARIN DATE 0 2012-01-01`
	`OUT VAROUT DATE 1 ?`
		Obligatoire si vous spécifiez une procédure stockée.

ParameterTableValue content	Valeur de variable. Pour les variables de sortie, la valeur doit être : **?**. Pour entrer une variable de date, utilisez le format suivant : **aaaa-mm-jj**. Si aucune valeur n'est indiquée pour un paramètre, la valeur est considérée comme NULL dans la base de données.	Obligatoire si vous spécifiez une procédure stockée.

Tableau 35. Attributs obligatoires et facultatifs pour la définition d'un travail de base de données. (suite)

Attribut	Description/valeur	Obligatoire

credentials	Nom d'utilisateur et mot de passe d'accès à la base de données (les utilisateurs de domaine ne sont pas pris en charge). Comme alternative au codage en dur des valeurs réelles, vous pouvez paramétrer de l'une des manières suivantes :
	• Entrez un *nom_utilisateur* spécifié dans la base de données avec la définition de l'utilisateur (applicable à tous les systèmes d'exploitation sur ce type de travail) et saisissez l'instruction suivante :
	`<jsdl:password>${password:nom_utilisateur}</jsdl:password>`
	Le mot de passe est récupéré dans la définition de l'utilisateur *nom_utilisateur* dans la base de données et résolu au cours de l'exécution. Pour plus d'informations, voir «Utilisation des définitions d'utilisateur sur des types de travaux avec des options avancées», à la page 214.
	Vous pouvez également spécifier l'utilisateur d'un poste de travail différent et utiliser la syntaxe suivante pour le mot de passe :
	`<jsdl:password>${password:poste_travail#nom_utilisateur}</jsdl:password>`
	• Saisissez un utilisateur et un mot de passe définis avec la commande `param` localement sur l'agent dynamique qui va exécuter le travail (si le travail consiste à être soumis à un pool ou à un pool dynamique, la définition doit être présente sur tous les agents du pool). A condition que vous ayez défini le nom d'utilisateur avec la variable *utilisateur* et un mot de passe, les instructions de données d'identification se présenteront ainsi :
	`<jsdl:userName>${agent:utilisateur}</jsdl:userName>`
	`<jsdl:password>${agent:password.utilisateur}</jsdl:password>`
	Les variables de l'utilisateur et du mot de passe sont résolues sur l'agent au moment de l'exécution. Pour plus d'informations, voir «Définition des variables et mots de passe pour la résolution locale sur des agents dynamiques», à la page 521.	

L'exemple suivant présente un travail qui exécute une requête sur une base de données DB2 :

```
$JOBS
AGENT#DATABASE
TASK
<?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
 xmlns:jsdldatabase="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdldatabase" name="database">
  <jsdl:application name="database">
    <jsdldatabase:database>
            <jsdldatabase:sqlActionInfo>
                <jsdldatabase:dbms>db2</jsdldatabase:dbms>
                <jsdldatabase:server>localhost</jsdldatabase:server>
                <jsdldatabase:port>50000</jsdldatabase:port>
                <jsdldatabase:database>TWS32</jsdldatabase:database>
                <jsdldatabase:statements>
                    <jsdldatabase:dbStatement>SELECT * FROM DWB.ARE_ABSTRACT_
RESOURCES</jsdldatabase:dbStatement>
                </jsdldatabase:statements>
                <jsdldatabase:credentials>
                    <jsdl:userName>${agent:dbvars..dbtwouser}</jsdl:userName>
                    <jsdl:password>${agent:password.${agent:dbvars..dbtwouser}}</jsdl:password>[1]
                </jsdldatabase:credentials>
            </jsdldatabase:sqlActionInfo>
        </jsdldatabase:database>
  </jsdl:application>
</jsdl:jobDefinition>
DESCRIPTION "Defined using composer."
RECOVERY STOP
```

Remarque : (1) le nom d'utilisateur a été défini sur l'agent qui exécute le travail avec une variable `dbtwouser` par le biais de la commande `param`. La valeur définie pour `dbtwouser` sera récupérée au moment de l'exécution auprès du fichier de variables `dbvars` situé sur l'agent. De même, le mot de passe pour la valeur représentée par `dbtwouser` a été défini sur l'agent à l'aide de la commande `param` et sera résolu au cours de l'exécution à partir du même fichier de variables.

Définition de travail - Travaux MSSQL

Cette section décrit les attributs obligatoires et facultatifs des travaux MSSQL. Chaque définition de travail utilise le format et les arguments suivants :

Tableau 36. Attributs obligatoires et facultatifs pour la définition d'un travail MSSQL.

Attribut	Description/valeur	Obligatoire
application name	base de données	✓
dbms	Type de base de données dans lequel vous souhaitez exécuter le travail. Etant donné que ce travail est spécifique pour la base de données Microsoft SQL Server, la seule valeur prise en charge est mssql.	✓
JDBC jar class path	Chemin d'accès aux fichiers JAR du client de base de données. Cette valeur remplace la valeur indiquée dans le fichier de configuration DatabaseJobExecutor.properties, s'il existe. La version 4 des pilotes JDBC est requise.	
server	Nom d'hôte du serveur sur lequel la base de données est installée.	✓
port	Numéro de port pour le travail de base de données.	✓
database	Nom de la base de données.	✓
dbStatement	Instruction SQL. Pour séparer les instructions, utilisez une ligne vide.	✓

credentials	Nom d'utilisateur et mot de passe d'accès à la base de données (les utilisateurs de domaine ne sont pas pris en charge). Comme alternative au codage en dur des valeurs réelles, vous pouvez paramétrer de l'une des manières suivantes :
	• Entrez un *nom_utilisateur* spécifié dans la base de données avec la définition de l'utilisateur *nom d'utilisateur* (applicable à tous les systèmes d'exploitation sur ce type de travail) et saisissez l'instruction suivante :
	`<jsdl:password>${password:nom_utilisateur}</jsdl:password>`
	Le mot de passe est récupéré dans la définition de l'utilisateur *nom_utilisateur* dans la base de données et résolu au cours de l'exécution. Pour plus d'informations, voir «Utilisation des définitions d'utilisateur sur des types de travaux avec des options avancées», à la page 214.
	• Saisissez un utilisateur et un mot de passe définis avec la commande `param` localement sur l'agent dynamique qui va exécuter le travail (si le travail consiste à être soumis à un pool ou à un pool dynamique, la définition doit être présente sur tous les agents du pool). A condition que vous ayez défini le nom d'utilisateur avec la variable *utilisateur* et un mot de passe, les instructions de données d'identification se présenteront ainsi :
	`<jsdl:userName>${agent:utilisateur}</jsdl:userName>`
	`<jsdl:password>${agent:password.utilisateur}</jsdl:password>`
	Les variables de l'utilisateur et du mot de passe sont résolues sur l'agent au moment de l'exécution. Pour plus d'informations, voir «Définition des variables et mots de passe pour la résolution locale sur des agents dynamiques», à la page 521.	

L'exemple suivant présente un travail qui exécute un travail sur une base de données MSSQL :

```
$JOBS
AGENT#MSSQLJOB
TASK
<?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
 xmlns:jsdldatabase="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdldatabase" name="database">
  <jsdl:application name="mssqljob">
    <jsdldatabase:database>
            <jsdldatabase:sqlActionInfo>
                <jsdldatabase:dbms>mssql</jsdldatabase:dbms>
                <jsdldatabase:server>localhost</jsdldatabase:server>
                <jsdldatabase:port>111</jsdldatabase:port>
                <jsdldatabase:database>MYDATABASE</jsdldatabase:database>
                <jsdldatabase:statements>
                    <jsdldatabase:dbStatement type="job">sada</jsdldatabase:dbStatement>
                </jsdldatabase:statements>
                <jsdldatabase:credentials>
```

```
                    <jsdl:userName>mssqluser</jsdl:userName>
                    <jsdl:password>${agent:password.mssqluser}</jsdl:password>[1]
                </jsdldatabase:credentials>
            </jsdldatabase:sqlActionInfo>
        </jsdldatabase:database>
  </jsdl:application>
</jsdl:jobDefinition>
DESCRIPTION "Defined using composer."
RECOVERY STOP
```

Remarque : (1) Le mot de passe de l'utilisateur `mssqluser` a été défini à l'aide de la commande d'utilitaire `param` dans le fichier de variables sur l'agent qui doit exécuter le travail. Il sera résolu au cours de l'exécution avec la valeur définie.

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle décrite dans :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Création de définitions de travail".

Pour plus d'informations sur la création et l'édition d'objets de planification, voir :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Conception de votre charge de travail".

Définition de travail - Travaux Java

Cette section décrit les attributs obligatoires et facultatifs des travaux Java. Chaque définition de travail utilise le format et les arguments suivants :

Tableau 37. Attributs obligatoires et facultatifs pour la définition d'un travail Java.

Attribut	Description/valeur	Obligatoire
nom d'application	java	✓
jarPath	Répertoire dans lequel les fichiers jar sont stockés. Sont inclus tous les fichiers jar stockés dans le répertoire et tous les sous-répertoires spécifiés.	
className	Nom de la classe que le travail doit exécuter.	✓
clé de paramètre	Les paramètres à utiliser lors de l'exécution de la classe Java.	

Pour plus d'informations sur le développement d'un travail Java, voir *Tivoli Workload Automation : Guide du développeur - Extension de Tivoli Workload Automation*.

L'exemple suivante présente un travail qui exécute une classe de nom `com.ibm.test.Test` et un paramètre `failExecution` :

```
$JOBS
AGENT#JAVA
TASK
<?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
xmlns:jsdljava="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdljava" name="java">
  <jsdl:application name="java">
    <jsdljava:java>
            <jsdljava:javaParms>
                <jsdljava:jarPath>C:\JavaExecutors</jsdljava:jarPath>
                <jsdljava:className>com.ibm.test.Test</jsdljava:className>
                <jsdljava:parameters>
                    <jsdljava:parameter key="input">failExecution</jsdljava:parameter>
                </jsdljava:parameters>
            </jsdljava:javaParms>
```

```
       </jsdljava:java>
  </jsdl:application>
</jsdl:jobDefinition> DESCRIPTION "Defined using composer."
 RECOVERY STOP
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle décrite dans :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Création de définitions de travail".

Pour plus d'informations sur la création et l'édition d'objets de planification, voir :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Conception de votre charge de travail".

Définition de travail - Travaux exécutables

Cette section décrit les attributs obligatoires et facultatifs des travaux exécutables. Chaque définition de travail utilise le format et les arguments suivants :

Tableau 38. Attributs obligatoires et facultatifs pour la définition d'un travail exécutable.

Attribut	Description/valeur	Obligatoire
nom d'application	exécutable	✓
interactive	Indique si le travail requiert une intervention de l'utilisateur. Cette option s'applique uniquement aux travaux qui s'exécutent sur les systèmes d'exploitation Windows.	✓
value	Indiquez le nom et la valeur d'un ou plusieurs arguments.	
script	Entrez un script à exécuter par le travail. Le script est créé et exécuté lors de l'exécution du travail. Vous pouvez indiquer les arguments dans cette balise ou les entrer dans la balise **value** et les appeler dans le script.	✓
suffix	Indiquez l'extension de nom de fichier du script à exécuter par le travail. Cette option s'applique uniquement aux travaux qui s'exécutent sur les systèmes d'exploitation Windows. N'incluez pas "." au début du nom d'extension.	

L'exemple suivant présente un travail qui exécute une commande ping sur deux sites Web. L'adresse des sites Web est définie dans la balise **value** et est appelée dans la balise **script**. Ce travail possède une relation d'affinité avec le travail `affine_test`, ce qui signifie qu'il s'exécute sur le même poste de travail qu'`affine_test` :

```
$JOBS
AGENT#EXECUTABLE
TASK
<?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
xmlns:jsdle="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdle" name="executable">
  <jsdl:application name="executable">
    <jsdle:executable interactive="false" workingDirectory="c:\">
            <jsdle:arguments>
                <jsdle:value>www.mysite.com</jsdle:value>
                <jsdle:value>www.yoursite.com</jsdle:value>
```

```
            </jsdle:arguments>
            <jsdle:script>ping %1 ping %2</jsdle:script>
        </jsdle:executable>
  </jsdl:application>
</jsdl:jobDefinition>
DESCRIPTION "Defined using composer."
TWSAFFINITY "affine_test"
RECOVERY STOP
```

L'exemple suivant présente un travail qui exécute un script `vbs` sur les systèmes d'exploitation Windows. L'extension du nom de fichier est définie dans l'attribut `suffix` de la balise `script` :

```
WIN_WKS1#VBS_NAT1
 TASK
    <?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
xmlns:jsdle="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdle" name="executable">
  <jsdl:application name="executable">
    <jsdle:executable interactive="true" workingDirectory="c:\tws">
            <jsdle:script suffix="vbs">Wscript.Echo "ciao"</jsdle:script>
        </jsdle:executable>
  </jsdl:application>
</jsdl:jobDefinition>
 RECOVERY STOP
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle décrite dans :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Création de définitions de travail".

Pour plus d'informations sur la création et l'édition d'objets de planification, voir :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Conception de votre charge de travail".

Définition de travail - Travaux de type commande distante

Pour lire un scénario commun basé sur la réalité qui atteint les objectifs métier, y compris l'implémentation d'un travail de type commande distante, voir http://pic.dhe.ibm.com/infocenter/tivihelp/v47r1/index.jsp?topic=/com.ibm.tivoli.itws.doc_8.6.0.2/scen/awsbsscp_rc.html.

La présente section décrit les attributs obligatoires et facultatifs des travaux de type commande distante. Chaque définition de travail utilise le format et les arguments suivants :

Tableau 39. Attributs obligatoires et facultatifs pour la définition d'un travail de type commande distante

Attribut	Description/valeur	Obligatoire
application name	remotecommand	✓

Tableau 39. Attributs obligatoires et facultatifs pour la définition d'un travail de type commande distante (suite)

Attribut	Description/valeur	Obligatoire

userName	Nom d'utilisateur autorisé à établir une connexion sur l'ordinateur distant à l'aide du protocole défini. Comme alternative au codage en dur des valeurs réelles, vous pouvez paramétrer de l'une des manières suivantes :
	• Entrez un *nom_utilisateur* spécifié dans la base de données avec la définition de l'utilisateur (applicable à tous les systèmes d'exploitation sur ce type de travail) et saisissez l'instruction suivante :
	`<jsdl:password>${password:nom_utilisateur}</jsdl:password>`
	Le mot de passe est récupéré dans la définition de l'utilisateur *nom_utilisateur* dans la base de données et résolu au cours de l'exécution. Pour plus d'informations, voir «Utilisation des définitions d'utilisateur sur des types de travaux avec des options avancées», à la page 214.
	Vous pouvez également spécifier l'utilisateur d'un poste de travail différent et utiliser la syntaxe suivante pour le mot de passe :
	`<jsdl:password>${password:poste_travail#nom_utilisateur}`
	`</jsdl:password>`
	• Saisissez un utilisateur et un mot de passe définis avec la commande `param` localement sur l'agent dynamique qui va exécuter le travail (si le travail consiste à être soumis à un pool ou à un pool dynamique, la définition doit être présente sur tous les agents du pool). A condition que vous ayez défini le nom d'utilisateur avec la variable *utilisateur* et un mot de passe, les instructions de données d'identification se présenteront ainsi :
	`<jsdl:userName>${agent:utilisateur}</jsdl:userName>`
	`<jsdl:password>${agent:password.utilisateur}</jsdl:password>`
		✓

password	Mot de passe de l'utilisateur autorisé. Le mot de passe est chiffré lorsque le travail est créé. Voir la description de userName pour plus de détails.	
serverName	Nom d'hôte de l'ordinateur où s'exécute l'instance commande distante.	✓
port	Numéro de port de l'ordinateur distant où la commande s'exécute.	✓
protocol	Valeurs possibles : **AUTO** Le protocole est automatiquement sélectionné à partir des protocoles existants : SSH, Windows, RSH et REXEC. Le produit tente d'utiliser d'abord le protocole SSH. Si ce protocole échoue, il utilise le protocole Windows. Lors de l'utilisation de Secure Shell (SSH), le chemin d'accès doit être au format SSH. Dans ce cas, le serveur ssh Cygwin est monté sur /home/Administrator. **SSH** Un protocole réseau qui fournit des fonctions d'accès aux fichiers, de transfert et de gestion de fichiers sur n'importe quel flux de données. **WINDOWS** Protocole de partage de fichiers de Microsoft. Utilisez la syntaxe samba pour spécifier le chemin d'accès. Partagez le dossier contenant les fichiers que vous souhaitez transférer. **RSH** Le protocole RSH (Remote Shell Protocol) est un protocole qui permet à un utilisateur d'exécuter des commandes sur un système distant sans devoir se connecter au système. **REXEC** Le serveur REXEC (Remote Execution) est une application TCP/IP (Transmission Control Protocol/Internet Protocol) qui permet à un client de soumettre des commandes système à un système distant. Le protocole REXEC (Remote Execution Protocol) autorise le traitement de ces commandes ou programmes sur n'importe quel hôte du réseau. L'hôte local reçoit ensuite les résultats du traitement des commandes.	
keystoreFilePath	Chemin d'accès complet du fichier de clés contenant la clé privée utilisée pour établir la connexion. Un fichier de clés est une base de données de clés. Les clés privées présentes dans un fichier de clés sont associées à une chaîne de certificats qui authentifie la clé publique correspondante sur le serveur distant. Un fichier de clés contient également les certificats provenant des entités de confiance. Concerne le protocole SSH uniquement.	

Tableau 39. Attributs obligatoires et facultatifs pour la définition d'un travail de type commande distante (suite)

Attribut	Description/valeur	Obligatoire
keystorePassword	Mot de passe qui protège la clé privée et qui est requis pour établir la connexion. Cet attribut est obligatoire uniquement si vous indiquez un chemin d'accès au fichier de clés. Si le chemin d'accès au fichier de clés et la combinaison fichier de clés/mot de passe ne parviennent pas à établir une connexion, une tentative est effectuée à l'aide du nom d'utilisateur et du mot de passe associés à l'utilisateur autorisé à établir une connexion sur l'ordinateur distant.	✓
command	Entrez la commande à soumettre sur l'ordinateur distant.	✓
environment	Sortie standard et fichiers d'erreur standard pour la commande distante. Ces fichiers se trouvent sur l'agent, et pas localement sur les postes de travail où s'exécute la commande distante. Vérifiez que vous disposez des droits en écriture sur les répertoires indiqués, faute de quoi aucun fichier ne sera créé. **Sortie standard** Chemin et nom de fichier dans lequel la sortie standard de la commande doit être sauvegardée. Indiquez un nom de chemin absolu ou un nom de chemin relatif pour accéder au répertoire de travail. Le fichier est remplacé chaque fois que la commande génère une nouvelle sortie. **Erreur standard** Chemin et nom de fichier dans lequel l'erreur standard de la commande doit être sauvegardée. Indiquez un nom de chemin absolu ou un nom de chemin relatif pour accéder au répertoire de travail. Le fichier est remplacé chaque fois que la commande génère une nouvelle erreur.	

L'exemple ci-dessous illustre la section JSDL “application” d'un modèle de définition de travail pour un travail de type commande distante :

```
$JOBS
NC112016#REMCMD
 TASK
    <?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
   xmlns:jsdlremotecommand="http://www.ibm.com/xmlns/prod/scheduling/1.0/
    jsdlremotecommand" name="REMOTECOMMAND">
  <jsdl:application name="remotecommand">
    <jsdlremotecommand:remotecommand>
   <jsdlremotecommand:RemoteCommandParameters>
    <jsdlremotecommand:taskPanel>
     <jsdlremotecommand:command>ping -c 10 localhost </jsdlremotecommand:command>
    </jsdlremotecommand:taskPanel>
    <jsdlremotecommand:environmentPanel>
     <jsdlremotecommand:standardOutput>stdout</jsdlremotecommand:standardOutput>
     <jsdlremotecommand:standardError>stderr</jsdlremotecommand:standardError>
    </jsdlremotecommand:environmentPanel>
    <jsdlremotecommand:serverPanel>
     <jsdlremotecommand:serverInfo>
      <jsdlremotecommand:serverName>9.168.112.16</jsdlremotecommand:serverName>
      <jsdlremotecommand:port>23</jsdlremotecommand:port>
      <jsdlremotecommand:protocol>ssh</jsdlremotecommand:protocol>
     </jsdlremotecommand:serverInfo>
     <jsdlremotecommand:credentials>
      <jsdl:userName>userName</jsdl:userName>
      <jsdl:password>{aes}mv0GJqOHWo8lbuhcpFaluL9RkGQKrYvTiAUpKTMgp90=</jsdl:password>
     </jsdlremotecommand:credentials>
     <jsdlremotecommand:certificates>
      <jsdlremotecommand:keystoreFilePath>/var/keyStoreFile</jsdlremotecommand:
                 keystoreFilePath>
      <jsdlremotecommand:keystorePassword>pwd</jsdlremotecommand:keystorePassword>
     </jsdlremotecommand:certificates>
    </jsdlremotecommand:serverPanel>
   </jsdlremotecommand:RemoteCommandParameters>
```

```
  </jsdlremotecommand:remotecommand>
  </jsdl:application>
</jsdl:jobDefinition>
 RECOVERY STOP
```

Vous planifiez les travaux de type commande distante de Tivoli Workload Scheduler en les définissant dans les flots de travaux. Ajoutez le travail à un flot de travaux avec tous les arguments de planification nécessaires et soumettez-le. Les travaux peuvent être soumis dans un environnement réparti à l'aide de Dynamic Workload Console ou de la ligne de commande **conman**. Les travaux peuvent être soumis dans un environnement z/OS à l'aide de Dynamic Workload Console. Après la soumission, lorsque le travail est en cours d'exécution (statut **EXEC**), vous ne pouvez pas arrêter le travail à l'aide de la commande **kill**. L'action **kill** est ignorée par **conman** et le travail poursuit son exécution.

Sur les systèmes répartis, si l'agent Tivoli Workload Scheduler devient indisponible lorsque vous soumettez le travail Tivoli Workload Scheduler de type commande distante ou lorsque le travail est en cours d'exécution, Tivoli Workload Scheduler affecte le statut Error ou ABEND au travail lorsque l'agent redémarre.

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle décrite dans :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Création de définitions de travail".

Pour plus d'informations sur la création et l'édition d'objets de planification, voir :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Conception de votre charge de travail".

Définition de travail - Travaux de méthode d'accès

Cette section décrit les attributs obligatoires et facultatifs des travaux de la méthode d'accès. Chaque définition de travail utilise le format et les arguments suivants :

Tableau 40. Attributs obligatoires et facultatifs pour la définition d'un travail de la méthode d'accès.

Attribut	Description/valeur	Obligatoire
nom d'application	xajob	✓
accessMethod	Nom de la méthode d'accès utilisée pour communiquer avec le système externe afin de lancer le travail et de renvoyer l'état de ce travail.	✓
target	Nom d'un fichier d'options.	
taskString	Commande devant être interprétée par la méthode sélectionnée. La longueur de ligne maximale est de 8 Ko.	✓

Tableau 40. Attributs obligatoires et facultatifs pour la définition d'un travail de la méthode d'accès. (suite)

Attribut	Description/valeur	Obligatoire

droits d'accès	Le nom et le mot de passe de l'utilisateur qui exécute ce travail. Comme alternative au codage en dur des valeurs réelles, vous pouvez paramétrer de l'une des manières suivantes :
	• Entrez un *nom_utilisateur* spécifié dans la base de données avec la définition de l'utilisateur (applicable à tous les systèmes d'exploitation sur ce type de travail) et saisissez l'instruction suivante :
	`<jsdl:password>${password:nom_utilisateur}</jsdl:password>`
	Le mot de passe est récupéré dans la définition de l'utilisateur *nom_utilisateur* dans la base de données et résolu au cours de l'exécution. Pour plus d'informations, voir «Utilisation des définitions d'utilisateur sur des types de travaux avec des options avancées», à la page 214.
	Vous pouvez également spécifier l'utilisateur d'un poste de travail différent et utiliser la syntaxe suivante pour le mot de passe :
	`<jsdl:password>${password:poste_travail#nom_utilisateur}</jsdl:password>`
	• Saisissez un utilisateur et un mot de passe définis avec la commande `param` localement sur l'agent dynamique qui va exécuter le travail (si le travail consiste à être soumis à un pool ou à un pool dynamique, la définition doit être présente sur tous les agents du pool). Si vous avez défini le nom d'utilisateur avec la variable *user* et un mot de passe, les instructions correspondantes de données d'identification se présenteront ainsi :
	`<jsdl:userName>${agent:utilisateur}</jsdl:userName>`
	`<jsdl:password>${agent:password.utilisateur}</jsdl:password>`
	Les variables de l'utilisateur et du mot de passe sont résolues sur l'agent au moment de l'exécution. Pour plus d'informations, voir «Définition des variables et mots de passe pour la résolution locale sur des agents dynamiques», à la page 521.	

L'exemple suivant présente un travail qui crée un fichier dans un dossier /methods à l'aide d'un travail de la méthode d'accès par défaut :

```
$JOBS
AGENT#XA_JOB
TASK
<?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl" xmlns:jsdlxa=
"http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdlxa" name="xajob">
  <jsdl:application name="xajob">
    <jsdlxa:xajob accessMethod="unixlocl" target="optionFile">
       <jsdlxa:taskString>touch file</jsdlxa:taskString>
       <jsdlxa:credentials>
         <jsdlxa:userName>TestUser</jsdlxa:userName>
         <jsdlxa:password>{aes}IEr/DES8wRzQEij1ySQBfUR587QBxM0iwfQ1EWJaDds=</jsdlxa:password>
       </jsdlxa:credentials>
     </jsdlxa:xajob>
  </jsdl:application>
</jsdl:jobDefinition>
DESCRIPTION "Defined using composer."
RECOVERY STOP
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle décrite dans :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Création de définitions de travail".

Pour plus d'informations sur la création et l'édition d'objets de planification, voir :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Conception de votre charge de travail".

Définition de travail - Travaux z/OS

Cette section décrit les attributs obligatoires et facultatifs des travaux z/OS. Un travail z/OS exécute la commande que vous spécifiez dans l'onglet JCL d'un système JCL. Ce type de travail ne s'exécute que sur Tivoli Workload Scheduler distributed - Agent for z/OS. Chaque définition de travail utilise le format et les arguments suivants :

Tableau 41. Attributs obligatoires et facultatifs pour la définition d'un travail z/OS.

Attribut	Description/valeur	Obligatoire
application name	jcl	✓
byDefinition	Type de soumission de travail. Il s'agit de l'unique type de soumission pris en charge.	
jclDefinition	Opération à effectuer sur le système JCL.	✓

L'exemple suivant présente un travail qui renvoie l'état du système JCL :

```
ZOSAGENT#JCLDEF
 TASK
<?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:jsdl=="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
 xmlnss:jsdljcl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdljcl">
<jsdl:application name="jcl">
<jsdljcl:jcl>
<jsdljcl:JCLParameters>
<jsdljcl:jcl>
<jsdljcl:byRefOrByDef>
<jsdljcl:byDefinition>
<jsdljcl:jclDefinition>//NORMAL JOB ,'TWS JOB',CLASS=A,MSGCLASS=A,>
// MSGLEVEL=(1,1)
//*
//STEP1 EXEC PGM=IEFBR14</jsdljcl:jclDefinition>
</jsdljcl:byDefinition>
</jsdljcl:byRefOrByDef>
</jsdljcl:jcl>
</jsdljcl:JCLParameters>
<jsdljcl:JOBParameters>
<jsdljcl:jobStreamName>${tws.jobstream.name}jsdljcl:jobStreamName>${tws.jobstream.name}>
<jsdljcl:inputArrival>${tws.job.ia}jsdljcl:inputArrival>${tws.job.ia}>
</jsdljcl:JOBParameters>
</jsdljcl:jcl>
</jsdl:application>
</jsdl:jobDefinition>
 DESCRIPTION "Sample JCL Job Definition"
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle décrite dans :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Création de définitions de travail".

Pour plus d'informations sur la création et l'édition d'objets de planification, voir :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Conception de votre charge de travail".

Définition de travail - Travaux IBM i

Cette section décrit les attributs obligatoires et facultatifs des travaux IBM i. Un travail IBM i exécute la commande que vous spécifiez dans l'onglet IBM i sur un

système IBM i (anciennement connu comme AS/400 et i5 OS). Chaque définition de travail utilise le format et les arguments suivants :

Tableau 42. Attributs obligatoires et facultatifs pour la définition d'un travail IBM i.

Attribut	Description/valeur	Obligatoire
nom d'application	ibmi	✓
commande	La commande à exécuter sur le système IBM i.	✓

L'exemple suivant présente un travail qui renvoie l'état du système IBM i :

```
$JOBS
AGENT#IBM_I
TASK
   <?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
 xmlns:jsdlibmi="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdlibmi" name="ibmi">
  <jsdl:application name="ibmi">
    <jsdlibmi:ibmi>
            <jsdlibmi:IBMIParameters>
                <jsdlibmi:Task>
                    <jsdlibmi:command>wrksyssts</jsdlibmi:command>
                </jsdlibmi:Task>
            </jsdlibmi:IBMIParameters>
        </jsdlibmi:ibmi>
  </jsdl:application>
</jsdl:jobDefinition>
RECOVERY STOP
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle décrite dans :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Création de définitions de travail".

Pour plus d'informations sur la création et l'édition d'objets de planification, voir :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Conception de votre charge de travail".

Définition de travail - Automatisation OSLC

Utilisez le travail d'automatisation OSLC pour appeler n'importe quel fournisseur OSLC qui implémente la spécification d'automatisation OSLC. Pour plus d'informations, voir http://open-services.net/wiki/automation/OSLC-Automation-Specification-Version-2.0/.

Avant de pouvoir définir les définitions de travail d'automatisation OSLC, vous devez exécuter certaines étapes prérequises, comme expliqué dans "Etapes prérequises pour créer des travaux d'automatisation OSLC et d'application des accès OSLC" dans le manuel *Dynamic Workload Console - Guide d'utilisation*.

Cette section décrit les attributs obligatoires et facultatifs des travaux d'automatisation OSLC. Chaque définition de travail utilise le format et les arguments suivants :

Tableau 43. Attributs obligatoires et facultatifs pour la définition d'un travail d'automatisation OSLC.

Attribut	Description/valeur	Obligatoire
URI de Registry Services	Adresse de Registry Services (par exemple, `https://myhost.mydomain:16311/oslc/pr`).	
Nom d'utilisateur de Registry Services	Utilisateur se connectant à Registry Services.	
Mot de passe de Registry Services	Mot de passe associé à l'utilisateur se connectant à Registry Services.	
URI du fournisseur de services	Adresse du fournisseur de services.	✓
Nom d'utilisateur du fournisseur de services	Utilisateur se connectant au fournisseur de services.	
Mot de passe d'accès au fournisseur de services	Mot de passe associé à l'utilisateur se connectant au fournisseur de services.	
Demande	Représentation RDF de la demande d'automatisation.	✓

L'exemple suivant affiche un travail qui planifie un flot de travaux Tivoli Workload Scheduler :

```
$JOBS
WKS#AUTOMATION
 TASK
    <?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
     xmlns:jsdloslcautomation="
http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdloslcautomation" name="OSLCAUTOMATION">
  <jsdl:application name="oslcautomation">
    <jsdloslcautomation:oslcautomation>
             <jsdloslcautomation:OSLCAutomationParameters>
                   <jsdloslcautomation:AutomationRequest>
                         <jsdloslcautomation:automationRequestGroup>
                               <jsdloslcautomation:automationRequestBody>

<!-- ajoutez la représentation rdf de la ressource -->

</jsdloslcautomation:automationRequestBody>
                            </jsdloslcautomation:automationRequestGroup>
                       </jsdloslcautomation:AutomationRequest>
                       <jsdloslcautomation:ConnectionInfo>
                           <jsdloslcautomation:ServiceProviderCatalogGroup>
                                 <jsdloslcautomation:catalogURI>
                                  https://registryserviceshost.domain:16311/oslc/pr>
                                 </jsdloslcautomation:catalogURI>
                                 <jsdloslcautomation:username>registryUser</jsdloslcautomation:
                                    username>
                                 <jsdloslcautomation:password>registryPassword</jsdloslcautomation:
                                    password>
                           </jsdloslcautomation:ServiceProviderCatalogGroup>
                           <jsdloslcautomation:ServiceProviderGroup>
                                 <jsdloslcautomation:serviceProviderURI>
https://serviceprovideraddress.domain:16310/oslc/providers/1360665198982</jsdloslcautomation:
                                    serviceProviderURI>
                                  <jsdloslcautomation:usernameSP>
                                   serviceProviderUser
                                  </jsdloslcautomation:usernameSP>
                                  <jsdloslcautomation:passwordSP>
                                   serviceProviderPassword
                                  </jsdloslcautomation:passwordSP>
                           </jsdloslcautomation:ServiceProviderGroup>
                        </jsdloslcautomation:ConnectionInfo>
```

```
            </jsdloslcautomation:OSLCAutomationParameters>
          </jsdloslcautomation:oslcautomation>
  </jsdl:application>
</jsdl:jobDefinition>
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle décrite dans :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Création de définitions de travail".

Pour plus d'informations sur la création et l'édition d'objets de planification, voir :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Conception de votre charge de travail".

Définition de travail - Application des accès OSLC

Utilisez le travail d'application des accès OSLC pour appeler n'importe quel fournisseur OSLC qui implémente la spécification d'application des accès OSLC.

Avant de pouvoir définir les définitions de travail d'application des accès OSLC, vous devez exécuter certaines étapes prérequises, comme expliqué dans "Etapes prérequises pour créer des travaux d'automatisation OSLC et d'application des accès OSLC" dans le manuel *Dynamic Workload Console - Guide d'utilisation*.

La présente section décrit les attributs obligatoires et facultatifs des travaux d'application des accès OSLC. Chaque définition de travail utilise le format et les arguments suivants :

Tableau 44. Attributs obligatoires et facultatifs pour la définition d'un travail d'application des accès OSLC.

Attribut	Description/valeur	Obligatoire
URI Registry Services	Adresse de Registry Services (par exemple, `https://myhost.mydomain:16311/odlc/pr`).	✓
Nom d'utilisateur de Registry Services	Utilisateur se connectant à Registry Services.	✓
Mot de passe de Registry Services	Mot de passe associé à l'utilisateur se connectant à Registry Services.	✓
URI du fournisseur de services	Adresse du fournisseur de services.	✓
Nom d'utilisateur du fournisseur de services	Utilisateur se connectant au fournisseur de services.	✓
Mot de passe d'accès au fournisseur de services	Mot de passe associé à l'utilisateur se connectant au fournisseur de services.	✓
Instance	Représentation RDF de l'instance à déployer.	✓

L'exemple suivant présente un travail qui planifie l'application des accès d'un modèle de système :

```
WKS#PROVSAMPLETASK
<?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
 xmlns:jsdloslcprovisioning="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdloslcprovisioning"
 name="OSLCPROVISIONING">
 <jsdl:application name="oslcprovisioning">
   <jsdloslcprovisioning:oslcprovisioning>
    <jsdloslcprovisioning:OSLCProvisioningParameters>
     <jsdloslcprovisioning:actionPanel>
      <jsdloslcprovisioning:instanceFromTemplate>
       <jsdloslcprovisioning:instance>
       <!-- Définition RDF de l'instance>
&lt;?ml version=&quot;1.0&quot; encoding=&quot;UTF-8&quot;?>
&lt;rf:RDF
     xmlns:rdf=&quot;http://www.w3.org/1999/02/22-rdf-syntax-ns#&quot;
     xmlns:oslc=&quot;http://open-services.net/ns/core#&quot;
     xmlns:sco=&quot;http://jazz.net/ns/ism/provisioning/sco#&quot;
     xmlns:oslc_auto=&quot;http://open-services.net/ns/auto#&quot;
     xmlns:dcterms=&quot;http://purl.org/dc/terms/&quot;
     xmlns:rdfs=&quot;http://www.w3.org/2000/01/rdf-schema#&quot; >
    &lt;rf:Description rdf:about=&quot;
http://myServiceProvider.domain:31115/CLIModelWeb/OSLC/BatchApplicationInstance/BatchApplication/
  0f01af24-72e0-3c4b-b95c-18f908c76898&quot;>
     &lt;olc_auto:parameterDefinition rdf:nodeID=&quot;A1&quot;/>
     &lt;olc_auto:parameterDefinition rdf:nodeID=&quot;A2&quot;/>
     &lt;rf:type rdf:resource=&quot;http://jazz.net/ns/ism/provisioning/sco#Entity&quot;/>
     &lt;olc_auto:parameterDefinition rdf:nodeID=&quot;A4"/>
     <dcterms:identifier>0f01af24-72e0-3c4b-b95c-18f908c76898</dcterms:identifier>
     <oslc_auto:parameterDefinition rdf:nodeID="A5"/>
     <dcterms:title>InstanceName</dcterms:title>
     <oslc_auto:parameterDefinition rdf:nodeID="A0"/>
   </rdf:Description>
   <rdf:Description rdf:nodeID="A5">
     <oslc:name>XML</oslc:name>
     <oslc:value>&lt;?xml version=&quot1.0&quot encoding=&quotUTF-8&quot;?&gt;
        &lt;model:TWSBatchApplicationInstance xmlns:model=&quot;
http://www.ibm.com/xmlns/prod/scheduling/1.0/Model&quot&gt;
        &lt;model:Name&gt;InstanceName&lt;/model:Name&gt;
             <!--Here starts the definition of jobs, job streams, etc contained in this instance>
             ...
             <!--Here starts the definition >
 &lt;/model:TWSBatchApplicationInstance&gt;&lt;/slc:value>
  &lt;olc:defaultValue>&lt;/slc:defaultValue>
     &lt;rdf:type rdf:resource=&quot;http://open-services.net/ns/core#Property&quot;/>
   </rdf:Description>
   <!--Mapping of the instance>
    &lt;rdf:Description rdf:nodeID=&quot;A0&quot;>
    &lt;oslc:name>JOB_SAMPLE&lt;/oslc:name>
        &lt;oslc:value>JOB_TARGET&lt;/oslc:value>
    &lt;oslc:defaultValue>JOB_TARGET&lt;/oslc:defaultValue>
    &lt;rdf:type rdf:resource=&quot;http://open-services.net/ns/core#Property&quot;/>
  &lt;/rdf:Description>
  <!--Here continues the definition of jobs, job streams, etc contained in this instance>
   ...
  <!--Here ends the mapping>
   &lt;rdf:Description rdf:nodeID=&quot;A7&quot;>
     &lt;sco:node rdf:resource=&quot;
http://thinklinux:31115/CLIModelWeb/OSLC/BatchApplicationInstance/BatchApplication/
      0f01af24-72e0-3c4b-b95c-18f908c76898&quot;/>
     &lt;rdf:type rdf:resource=&quot;http://jazz.net/ns/ism/provisioning/sco#Template&quot;/>
    &lt;/rdf:Description>
   &lt;rdf:Description rdf:nodeID=&quot;A1&quot;>
     &lt;oslc:name>ICON&lt;/oslc:name>
     &lt;oslc:value>../js/images/default.png&lt;/oslc:value>
     &lt;oslc:defaultValue>../js/images/default.png&lt;/oslc:defaultValue>
    &lt;rdf:type rdf:resource=&quot;http://open-services.net/ns/core#Property&quot;/>
   &lt;/rdf:Description> &lt;/rdf:RDF></jsdloslcprovisioning:instance>
      </jsdloslcprovisioning:instanceFromTemplate>
     </jsdloslcprovisioning:actionPanel>
     <jsdloslcprovisioning:connectionInfo>
      <jsdloslcprovisioning:serviceProviderCatalog>
       <jsdloslcprovisioning:catalogURI>
            https://myregistry.domain:16311/oslc/pr</jsdloslcprovisioning:catalogURI>

       <jsdloslcprovisioning:username>registryServicesUser</jsdloslcprovisioning:username>

      <jsdloslcprovisioning:password>registryServicesPassword</jsdloslcprovisioning:password>
                                            </jsdloslcprovisioning:serviceProviderCatalog>
                                             <jsdloslcprovisioning:serviceProvider>
                                             <jsdloslcprovisioning:serviceProviderURI>
```

```
https://myregistry.domain:16311/oslc/providers/1380617052297
                                                    </jsdloslcprovisioning:serviceProviderURI>

       <jsdloslcprovisioning:usernameSP>myServiceProviderUser</jsdloslcprovisioning:usernameSP>
       <jsdloslcprovisioning:passwordSP>myServiceProviderPassword</jsdloslcprovisioning:passwordSP>
                                                 </jsdloslcprovisioning:serviceProvider>
                                        </jsdloslcprovisioning:connectionInfo>
                                    </jsdloslcprovisioning:OSLCProvisioningParameters>
                                </jsdloslcprovisioning:oslcprovisioning>
  </jsdl:application>
 </jsdl:jobDefinition>
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle décrite dans :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Création de définitions de travail".

Pour plus d'informations sur la création et l'édition d'objets de planification, voir :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Conception de votre charge de travail".

Utilisation de variables et de paramètres dans les définitions de travail

Une variable est un objet de planification qui fait partie d'une table de variables et qui est défini dans la base de données Tivoli Workload Scheduler. Elle peut être utilisée par tous les agents du domaine tant que les utilisateurs peuvent accéder au fichier de sécurité.

Un paramètre est défini et utilisé en local sur un agent (avec la commande d'utilitaire `parms`).

Les utilisations et restrictions suivantes s'appliquent aux variables et aux paramètres des définitions de travail :

- Des variables et des paramètres sont admis dans les valeurs des mots clé **streamlogon**, **scriptname**, **docommand** et **abendprompt**.
- Une variable ou un paramètre peut correspondre à une chaîne entière ou partielle.
- Plusieurs variables et paramètres sont admis dans une seule zone.
- Les noms de variables doivent figurer entre carets (^) et la chaîne entière entre guillemets. Assurez-vous que les caractères caret ne sont pas précédés d'une barre oblique inversée dans la chaîne. Si nécessaire, incluez la barre oblique inversée dans la définition de la variable ou du paramètre.
- Sous UNIX, les noms de paramètres doivent figurer entre des apostrophes (') et la chaîne entière entre guillemets.
- Voir «Définition des variables et des paramètres», à la page 217 pour plus d'informations et d'exemples.

Dans l'exemple ci-dessous, une variable nommée **mis** est utilisée dans la valeur **streamlogon** :

```
$jobs
cpu1#bkup
     scriptname "/usr/mis/scripts/bkup"
     streamlogon "^mis^"
     recovery continue after recjob1
```

Pour d'autres exemples, voir «Définition des variables et des paramètres», à la page 217.

Définition d'utilisateur

Les noms d'utilisateur utilisés comme valeur **streamlogon** dans les définitions de travail Windows doivent posséder des définitions utilisateur. Cette règle n'est pas obligatoire pour les utilisateurs qui exécutent des travaux sur d'autres systèmes d'exploitation. Si vous utilisez destypes de travaux avec options avancées, vous pouvez exploiter ces valeurs quel que soit le système d'exploitation. Pour plus d'informations, voir «Utilisation des définitions d'utilisateur sur des types de travaux avec des options avancées», à la page 214.

Remarque : Si l'option globale `enAddUser` vaut `"yes"`, la définition d'utilisateur est automatiquement ajoutée au plan après la création ou la modification de la définition d'utilisateur dans la base de données.

Syntaxe

Chaque définition d'utilisateur utilise le format et les arguments suivants :

```
username[poste_de_travail#][domaine\]nom_utilisateur[@domaine_internet]
password "mot_de_passe"
end
```

Arguments

username

[*poste_de_travail***#]***nom_utilisateur*

poste de travail
Indique le poste de travail sur lequel l'utilisateur peut exécuter des travaux. Le signe dièse est obligatoire. La valeur par défaut est le caractère blanc qui signifie que tous les postes de travail sont concernés.

nom_utilisateur
Indique le nom de l'utilisateur Windows. La zone *nom_utilisateur* peut comporter jusqu'à 47 caractères.

[*poste_de_travail***#]***domaine**nom_utilisateur*

poste de travail
Indique le poste de travail sur lequel l'utilisateur peut exécuter des travaux. Le signe dièse est obligatoire. La valeur par défaut est le caractère blanc qui signifie que tous les postes de travail sont concernés.

*domaine**nom_utilisateur*

Indique le nom de l'utilisateur du domaine Windows. La zone *domaine**nom_utilisateur* peut comporter jusqu'à 47 caractères.

[*poste_de_travail***#]***nom_utilisateur***@***domaine_internet*

poste de travail
Indique le poste de travail sur lequel l'utilisateur peut exécuter des travaux. Le signe dièse est obligatoire. La valeur par défaut est le caractère blanc qui signifie que tous les postes de travail sont concernés.

nom_utilisateur@domaine_interne

Indique le nom de l'utilisateur au format UPN (nom principal de l'utilisateur). Le format UPN est le nom d'un utilisateur système dans un format d'adresse électronique. Le nom d'utilisateur est suivi du symbole @, lui-même suivi du nom du domaine Internet auquel l'utilisateur est associé. La zone *nom_utilisateur@domaine_internet* peut comporter jusqu'à 47 caractères.

Remarque :

Si vous définissez un utilisateur pour les **systèmes d'exploitation Windows** :

- Les majuscules et les minuscules sont différenciées dans les noms d'utilisateur. En outre, l'utilisateur doit être autorisé à se connecter au poste de travail sur lequel Tivoli Workload Scheduler lance les travaux et disposer de l'autorisation d'accès requise pour **se connecter par lot**.
- Si le nom d'utilisateur n'est pas unique, il est considéré comme étant respectivement un utilisateur local, un utilisateur de domaine ou un utilisateur de domaine sécurisé.

password

Indique le mot de passe de l'utilisateur. Le mot de passe peut comporter jusqu'à 31 caractères et doit figurer entre guillemets. Pour indiquer un mot de passe vide, utilisez deux guillemets consécutifs qui ne soient pas séparés par des espaces, **""**. Une fois une définition d'utilisateur enregistrée, il est impossible de lire le mot de passe. Les utilisateurs disposant des droits d'accès appropriés peuvent modifier ou supprimer un utilisateur, mais les informations relatives aux mots de passe ne sont jamais affichées.

Exemples

L'exemple suivant permet de définir quatre utilisateurs :

```
username joe
     password "okidoki"
end
#
username server#jane
     password "okitay"
end
#
username dom1\jane
     password "righto"
end
#
username jack
     password ""
end
#
username administrator@twsbvt.com
     password "internetpwd"
end
#
username serverA#dom1\jack
     password "righto"
end
#
```

```
username serverB#user1@twsbvt.com
     password "internetpwd"
end
#
```

Commentaires

Les mots de passe extraits avec la commande `composer extract` ont un usage limité. Lorsque vous exécutez la commande `composer extract` sur une définition d'utilisateur, le mot de passe est masqué avec le mot clé réservé "**********". Si vous essayez d'exécuter les commandes composer `import`, `replace` ou `modify` sur un mot de passe utilisateur extrait, le remplacement du mot de passe n'a aucun effet et l'ancien mot de passe est conservé. De plus, si vous tentez d'utiliser les commandes composer `create`, `new` ou `add` sur un utilisateur où le mot de passe correspond au mot clé réservé "**********", l'erreur suivante est renvoyée :

```
AWSJCL521E Le mot de passe spécifié pour l'utilisateur Windows "USER_NAME" n'est
pas conforme aux exigences de règles de sécurité du mot de passe.
```

Notez que le mot-clé réservé est une chaîne de 10 astérisques (*). Vous ne pouvez pas entrer une séquence de dix astérisques comme mot de passe mais vous pouvez la définir avec n'importe quel autre nombre d'astérisques.

Pour corriger ce problème, assurez-vous d'exécuter `composer extract` avec l'option ;password.

Voir aussi

Pour plus d'informations sur la façon d'effectuer la même tâche à partir de Dynamic Workload Console, voir :

la section "Conception de votre charge de travail" dans le document Dynamic Workload Console - Guide d'utilisation.

Utilisation des définitions d'utilisateur et streamlogon Tivoli Workload Scheduler

Sur les postes de travail Windows, vous devez spécifier les définitions d'utilisateur à l'aide de **composer** sous la forme **[*poste_de_travail*#]*nom_utilisateur***. L'instance **[*poste_de_travail*#]*nom_utilisateur*** identifie de façon unique l'utilisateur Windows dans l'environnement Tivoli Workload Scheduler. Le nom du poste de travail est facultatif ; si vous ne l'indiquez pas, cela signifie que l'utilisateur nommé *nom_utilisateur* est défini sur tous les postes de travail Windows du réseau Tivoli Workload Scheduler. Si l'utilisateur nommé *nom_utilisateur* est défini uniquement sur certains postes de travail Windows du réseau Tivoli Workload Scheduler, vous devez créer, afin d'éviter les incohérences, une définition d'utilisateur au format [*poste_de_travail*#]*nom_utilisateur* pour chaque poste de travail Windows où l'utilisateur *nom_utilisateur* est défini.

Si vous planifiez un travail sur un agent, sur un pool ou un pool dynamique, le travail s'exécute avec l'utilisateur défini sur le pool ou le pool dynamique. Cependant, l'utilisateur doit exister sur tous les postes de travail du pool ou du pool dynamique où vous planifiez d'exécuter le travail.

Si vous définissez un travail à l'aide du programme **composer**, vous devez spécifier à la fois un poste de travail et lui affecter un nom de connexion correct. Le nom de connexion est simplement un nom d'utilisateur valide pour Windows, sans le nom du poste de travail. Par exemple, dans la définition de travail suivante :

```
$JOB
poste_de_travail#job01 docommand "dir"
streamlogon nom_utilisateur
```

la valeur de l'option `streamlogon` est *nom_utilisateur* et non *poste_de_travail#nom_utilisateur*.

Si vous utilisez la commande **altpass**, rappelez-vous que vous devez toutefois utiliser la définition d'utilisateur sous la forme

poste_de_travail#nom_utilisateur

Dans cette commande, vous pouvez omettre le nom du poste de travail seulement si vous modifiez son mot de passe sur la machine sur laquelle vous exécutez la commande.

Utilisateur de domaine sécurisé

Si Tivoli Workload Scheduler doit lancer des travaux pour un utilisateur de domaine sécurisé, respectez les instructions suivantes lors de la définition des comptes utilisateur. En supposant que Tivoli Workload Scheduler est installé dans le domaine `Domaine1` pour le compte utilisateur `maestro` et que le compte utilisateur `sue` dans le domaine `Domaine2` doit lancer un travail, les conditions suivantes doivent être remplies :

- Une sécurisation réciproque doit subsister entre `Domaine1` et `Domaine2`.
- Dans `Domaine1`, sur les ordinateurs sur lesquels des travaux sont lancés, `sue` doit posséder le droit de se **connecter par lot**.
- Dans `Domaine1`, `maestro` doit correspondre à un utilisateur de domaine.
- Dans `Domaine2`, `maestro` doit détenir les droits pour **accéder à l'ordinateur à partir du réseau**.

Utilisation des définitions d'utilisateur sur des types de travaux avec des options avancées

Avec les types de travaux avec options avancées, quel que soit le système d'exploitation de l'agent dynamique qui exécutera le travail, vous pouvez fournir le nom d'utilisateur d'une définition d'utilisateur dans la section des droits d'accès du travail et résoudre la zone mot de passe lors de l'exécution avec la valeur de mot de passe stockée dans la base de données.

Par exemple, lorsque vous définissez le travail avec Dynamic Workload Console, entrez le nom d'utilisateur d'une définition d'utilisateur et cliquez sur les points de suspension (...) situés en regard de la zone du mot de passe pour afficher le widget `Type de mot de passe` :

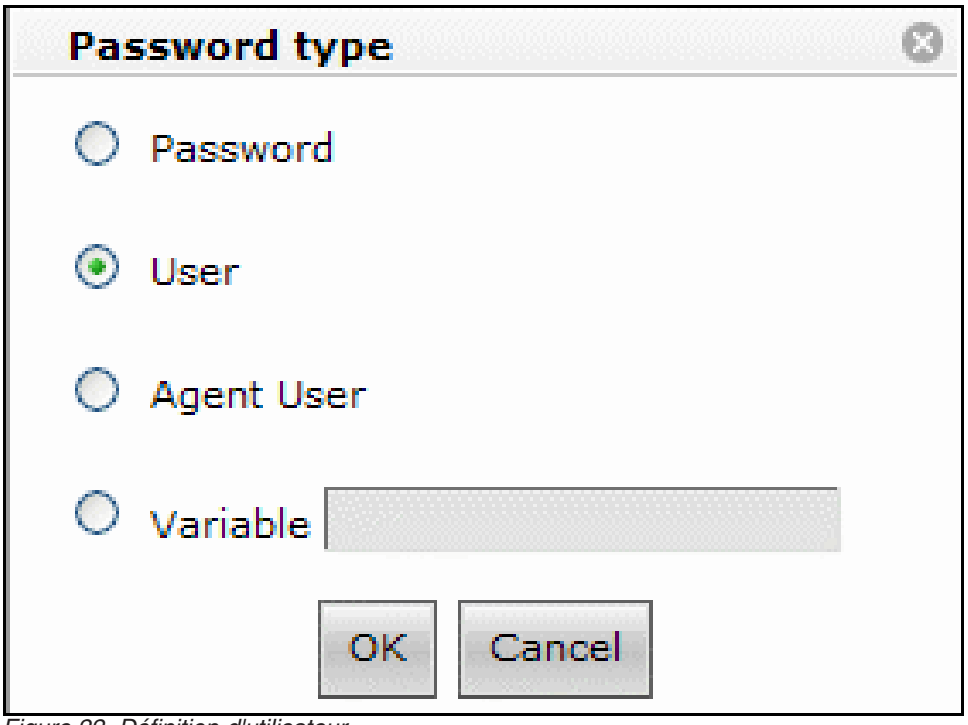


Figure 22. Définition d'utilisateur

où vous sélectionnez `Utilisateur` comme indiqué. Vous pouvez de la même façon coder cette option dans la section tâche (JSDL) de la définition de travail dans le composeur. Pour plus d'informations, consultez les sections connexes.

Pour pouvoir utiliser cette option lorsque vous définissez un travail, vous devez être autorisé dans le fichier de sécurité avec le mot-clé `use` pour le type d'objet `userobj` :

```
userobj  access=use
```

Tivoli Workload Scheduler suit cette séquence lorsqu'il est appelé pour résoudre le nom d'utilisateur et le mot de passe lors de l'exécution :

- Si le poste de travail n'est pas spécifié (par exemple, `${password:myuser}`) :
  1. Recherche `myuser` sur le poste de travail qui exécute le travail qui applique une politique sensible à la casse.
  2. Recherche `myuser` sur le poste de travail qui exécute le travail qui applique une politique insensible à la casse.
  3. Recherche `myuser` sans poste de travail associé qui applique une politique sensible à la casse.
  4. Recherche `myuser` sans poste de travail associé qui applique une politique insensible à la casse.
- Si le poste de travail est spécifié (par exemple, `${password:agent#myuser}`) :
  1. Recherche `myuser` sur l'`agent` du poste de travail qui applique une politique sensible à la casse.
  2. Recherche `myuser` sur l'`agent` du poste de travail qui applique une politique insensible à la casse.
  3. Recherche `myuser` sans poste de travail associé qui applique une politique sensible à la casse.
  4. Recherche `myuser` sans poste de travail associé qui applique une politique insensible à la casse.

- Si le poste de travail est spécifié mais vide (par exemple, `${password:#myuser}`) :
  1. Recherche `myuser` sans poste de travail associé qui applique une politique sensible à la casse.
  2. Recherche `myuser` sans poste de travail associé qui applique une politique insensible à la casse.

Avertissement : Les définitions d'utilisateur manquent d'intégrité référentielle. Cela implique que, si une définition d'utilisateur référencée dans la section des données d'identification d'un type de travail avec des options avancées est modifiée ou supprimée, aucun message d'avertissement ou d'erreur n'est renvoyé jusqu'à l'exécution du travail.

Définition d'agenda

Un agenda est une liste de dates qui définissent si un flot de travaux s'exécute et le moment où il s'exécute. Chaque définition d'agenda utilise le format et les arguments suivants :

Syntaxe

$calendar
nom_agenda [*"description"*]
date [...]

[*nom_agenda* ...]

Arguments

nom_agenda
Nom de l'agenda. Ce nom peut comporter jusqu'à huit caractères alphanumériques, incluant les tirets (-) et les traits de soulignement (_), et doit commencer par une lettre.

"*description***"**
Fournit une description de l'agenda. La description peut contenir jusqu'à 120 caractères alphanumériques. Elle doit figurer entre guillemets. Son nom peut comporter des caractères alphanumériques à condition de commencer par une lettre. Les caractères suivants sont admis : virgule (,), point (.), tiret (-), plus (+), apostrophe (') et égal (=). Il ne peut pas comporter les caractères suivants : guillemet (") (autres que ceux encadrant la description), deux-points (:), point-virgule(;) et la perluète (&).

date **[...]**
Indique une ou plusieurs dates, séparées par des espaces. Le format est *mm/jj/aa*.

Exemples

L'exemple ci-après permet de définir trois agendas intitulés `monthend`, `paydays` et `holidays` :

```
$calendar
monthend "Month end dates 1st half 2005"
     01/31/2005 02/28/2005 03/31/2005 04/30/2005 05/31/2005 06/30/2005
paydays
     01/15/2005 02/15/2005
```

```
     03/15/2005 04/15/2005
     05/14/2005 06/15/2005
holidays
     01/01/2005 02/15/2005 05/31/2005
```

Voir aussi

Pour plus d'informations sur la façon d'effectuer la même tâche à partir de Dynamic Workload Console, voir :

la section "Conception de votre charge de travail" dans le document Dynamic Workload Console - Guide d'utilisation.

Définition des variables et des paramètres

Les variables et les paramètres sont des objets auxquels vous attribuez différentes valeurs.

Les variables et les paramètres sont utiles lorsque des valeurs changent en fonction des flots de travaux et des travaux. Les définitions de flot de travaux, de travail et d'invite qui les utilisent sont mises à jour automatiquement soit au début du cycle de production, soit lors de l'exécution du travail en fonction du format utilisé lorsque la variable est spécifiée.

Utilisez les variables et les paramètres en remplacement des valeurs répétitives lors de la définition d'invites, de travaux et de flots de travaux. Par exemple, si vous utilisez des variables pour les ouvertures de session utilisateur et les noms de fichiers script dans les définitions de travaux, ainsi que pour les dépendances de fichier et d'invite, vous utilisez des valeurs permettant une gestion centralisée dans la base de données du gestionnaire maître.

Lorsque les variables sont des objet de planification définis dans la base de données Tivoli Workload Scheduler et qu'elles peuvent être utilisées par tout utilisateur autorisé du domaine, les paramètres sont définis et utilisés en local sur des agents individuels.

Les sections suivantes décrivent en détail les variables et les paramètres.

Variables

Les variables sont définies en tant qu'objets de planification dans la base de données. Les variables peuvent être définies de manière individuelle avec la commande suivante :

$parm
[*nom_table.*]*nom_variable "valeur_variable"*
...

Où :

nom_table
: Nom de la table de variables contenant la nouvelle variable. Elle doit déjà être définie. Si vous n'indiquez pas de nom de table de variables, la variable est ajoutée à la table par défaut.

nom_variable
Nom de la variable. Ce nom peut comporter jusqu'à 16 caractères alphanumériques, incluant les tirets (-) et les traits de soulignement (_), et doit commencer par une lettre.

valeur Valeur affectée à la variable. La valeur peut comporter jusqu'à 72 caractères alphanumériques. N'incluez pas les noms d'autres variables.

Toutefois, le meilleur moyen de définir une variables est d'utiliser une «Définition de table de variables», à la page 222. Dans tous les cas, toutes les variables sont placées dans une table de variables. Si vous définissez une variable sans préciser le nom de la table de variables, elle est incluse dans la table de variables par défaut.

Vous pouvez utiliser des variables dans les définitions de travail et de flot de travaux. Ils sont résolus, c'est-à-dire, remplacés par la valeur qui leur a été affectée, lorsque le plan de production est généré ou étendu, ou que vous soumettez l'exécution d'un travail ou d'un flot de travaux. Le format utilisé pour spécifier une variable détermine également le moment où la variable est résolue avec une valeur. Les formats suivants peuvent être utilisés lors de la spécification d'une variable :

^*nom_variable***^**
Spécifiez la variable à ce format si vous souhaitez qu'elle soit résolue lorsque le plan est généré ou étendu.

${*nom_variable***}**
Spécifiez la variable à ce format si vous souhaitez qu'elle soit résolue ou substituée lorsque l'exécution du travail ou du flot de travaux est soumise. Une option dans la définition de travail indiquant de résoudre les variables au moment de l'exécution du travail doit également être spécifiée. Si cette variable n'est présente que dans la table des variables par défaut, elle ne peut pas être résolue. Pour obtenir un exemple d'application de ce type de variable, voir la section «Exemples», à la page 220.

Pour plus de détails sur la résolution des variables, voir «Résolution de variable», à la page 125.

Les noms de variables spécifiés dans ces définitions sont d'abord résolus pour les définitions de tables de variables, puis pour les paramètres locaux si les variables sont introuvables.

Lorsque vous spécifiez une variable, placez la chaîne contenant la variable entre guillemets.

Si la variable contient une partie d'un chemin, vérifiez que les accents circonflexes ne sont pas immédiatement précédés d'une barre oblique inversée (\). En effet, la séquence \^ pourrait être interprétée comme une séquence d'échappement et serait alors résolue par le programme d'analyse syntaxique comme un accent circonflexe. Si nécessaire, placez la barre oblique inversée dans la définition de la variable entre carets afin d'éviter une interprétation incorrecte. Par exemple, le tableau ci-dessous illustre le meilleur moyen de définir et d'utiliser une variable nommée `MYDIR` dans la table de variables par défaut :

Tableau 45. Traitement d'une barre oblique inversée dans la substitution de variable

Mauvais moyen	Bon moyen

1. Définissez la variable `MYDIR` de la manière suivante :
`$PARM`
`MYDIR "scripts"`
2. Utilisez-la de cette manière :
`job01 scriptname`
`"c:\operid\^MYDIR^\test.cmd"`
3. Utilisez-la de cette manière :
`job01 scriptname`
`"c:\operid\${MYDIR}\test.cmd"`
	1. Définissez la variable `MYDIR` de la manière suivante :
	`$PARM`
	`MYDIR "\scripts"`
	2. Utilisez-la de cette manière :
	`job01 scriptname`
	`"c:\operid^MYDIR^\test.cmd"`
	3. Utilisez-la de cette manière :
	`job01 scriptname`
	`"c:\operid{MYDIR}\test.cmd"`

Cela est vrai pour toutes les commandes de ligne de commande, les interfaces graphiques et les API grâce auxquelles vous utilisez la substitution de variable.

Paramètres

Les paramètres locaux sont définis dans une base de données locale sur le poste de travail sur lequel est exécuté le travail qui les utilise. Pour les définir, n'utilisez pas cette commande **composer** mais la commande «parms», à la page 568.

Les paramètres locaux peuvent être utilisés dans les :
- JCL
- ouvertures de session
- dépendances d'invite
- dépendances de fichier
- invites de reprise

Un paramètre local est défini dans ces mots clés ou dans le script de travail appelé à l'aide de la syntaxe suivante :

```
'bin\parms PARAMETERNAME'
```

Les paramètres locaux sont résolus à l'aide des définitions stockées dans la base de données PARMS locale comme suit :

- Au moment de l'exécution, sur le poste de travail où le traitement du travail a lieu.
- Au moment de la soumission, sur le poste de travail sur lequel le travail ou le flot de travaux est soumis à partir de la ligne de commande **conman**. Le tableau 46 récapitule les commandes **submit** dans lesquelles vous pouvez utiliser les paramètres.

Tableau 46. Mots clés auxquels peuvent être associés des paramètres locaux dans les commandes **submit**

Mot clé	submit docommand (commande sbd)	submit file (commande sbf)	submit job (commande sbj)	submit job stream (commande sbs)
abendprompt	✓	✓	✓	
scriptname		✓		
docommand	✓			
logon	✓	✓		
opens	✓	✓	✓	✓

Tableau 46. Mots clés auxquels peuvent être associés des paramètres locaux dans les commandes **submit** *(suite)*

Mot clé	submit docommand (commande sbd)	submit file (commande sbf)	submit job (commande sbj)	submit job stream (commande sbs)
prompt	✓	✓	✓	✓

Pour plus d'informations sur la soumission de travaux et de flots de travaux en production à partir de la ligne de commande **conman**, voir Chapitre 11, «Gestion des objets dans le plan - conman», à la page 373.

Sous UNIX, lorsque vous définissez un travail ou un flot de travaux dans la base de données, vous devez placer la chaîne

chemin/parms *nom_paramètre*

entre les caractères ' ' pour que le paramètre soit résolu au moment de l'exécution sur le poste de travail, même si un paramètre ayant le même nom est défini en tant que paramètre global dans la base de données Tivoli Workload Scheduler. Par exemple, si vous ajoutez la définition de travail suivante à la base de données :

```
$jobs
myjob
  docommand "ls ^MYDIR^"
  streamlogon "^MYUSER^"
```

et que deux paramètres nommés MYDIR et MYUSER sont définis dans la base de données, les deux paramètres sont résolus, lors de la création ou de l'extension du plan de production, à l'aide des définitions contenues dans la base de données et leurs valeurs correspondantes sont contenues dans le fichier Symphony. Si vous définissez myjob dans la base de données comme suit :

```
$jobs
myjob
  docommand "ls 'bin/parms MYDIR'"
  streamlogon "'bin/parms MYUSER'"
```

la seule action effectuée lors de la création ou de l'extension du plan de production sur les deux paramètres dans la définition de myjob est la suppression des caractères ' '. Les paramètres sont contenus dans le fichier Symphony, non résolus, puis ils sont résolus au moment de l'exécution en local sur le poste de travail cible à l'aide de la valeur stockée dans la base de données PARMS.

Exemples

Deux paramètres, glpah et gllogon, sont définis de la manière suivante :

```
$parm
glpath   "/glfiles/daily"
gllogon   "gluser"
```

Dans un flot de travaux intitulé glsched, les paramètres glpath et gllogon sont utilisés dans un travail intitulé gljob2 :

```
schedule glsched on weekdays
:
gljob2
     scriptname "/usr/gl^glpath^"
     streamlogon "^gllogon^"
     opens "^glpath^/datafile"
     prompt ":^glpath^ started by ^gllogon^"
end
```

Voici un exemple de variable utilisée avec le mot clé **docommand** :

```
docommand "ls ^MY_HOME^"
```

L'exemple ci-dessous décrit comment la spécification de variables dans différents formats permet aux variables d'avoir plusieurs valeurs, étant donné qu'elles sont résolues à différents moments. Il explique également comment les variables peuvent être transmises d'un travail à un autre dans un flot de travaux. La variable *SWITCH_VAR* est définie dans la table de variable STATETABLE avec la valeur initiale par défaut `on`. Le travail UPDATE1 est chargé de modifier la valeur de la variable *SWITCH_VAR* dans la table de variable STATETABLE en `off`. Le flot de travaux PROCJS contient deux travaux identiques, PROC1 et PROC2, dans lesquels la variable *SWITCH_VAR* a été spécifiée dans deux formats différents. Le premier place la variable entre carets (^) (^*nom_var*^) et le second utilise le format ${*nom_var*} :

```
<jsdle:script>echo ^SWITCH_VAR^:${SWITCH_VAR}</jsdle:script>
```

L'ordre dans lequel ces travaux s'exécutent est le suivant :

```
SCHEDULE NC117126#PROCJS
VARTABLE STATETABLE
:
NC117126_1# PROC1

NC117126_1# PROC2
 FOLLOWS UPDATE1

NC117126_1# UPDATE1
 FOLLOWS  PROC1
END
```

Lorsque le flot de travaux est ajouté au plan, *SWITCH_VAR*, défini dans PROC1 et PROC2, prend immédiatement la valeur par défaut `on` attribuée dans la table de variable. Lorsque le flot de travaux est soumis pour être exécuté, le premier travail à être soumis est PROC1 et la variable définie comme *SWITCH_VAR* est résolue en `on` de sorte que les variables du travail PROC1 soient résolues comme suit :

```
<jsdle:script>echo on:onjsdle:script>echo on:on</jsdle:script>
```

UPDATE1 paramètre ensuite la valeur de *SWITCH_VAR* dans la table de variable sur `off` de sorte que, lorsque PROC2 s'exécute, les variables soient résolues comme suit :

```
<jsdle:script>echo on:onjsdle:script>echo on:off</jsdle:script>
```

La variable spécifiée comme ^*SWITCH_VAR*^ dans le travail conserve la valeur `on` car les variables à ce format sont résolues lorsque le flot de travaux est ajouté au plan et ne sont pas actualisées lorsque l'exécution du travail est soumise. La variable spécifiée au format *${SWITCH_VAR}*, précédemment définie sur `on` est désormais mise à jour avec la nouvelle valeur dans la table de variable `off`.

Création d'une définition de variable à l'aide de Dynamic Workload Console

Pour créer une définition de variable dans Dynamic Workload Console, vous devez l'ajouter dans une définition de table de variable :

1. Cliquez sur **Tivoli Workload Scheduler→Workload (Charge de travail)→Design (Conception)→Create Workload Definitions (Créer des définitions de charge de travail)**
2. Sélectionnez un nom de moteur et cliquez sur **Go (Aller)**

3. Ouvrez en mode édition une table de variables existante à partir du panneau Ouverture rapide, ou créez une nouvelle table de variables comme décrit dans «Définition de table de variables»
4. Dans le panneau Propriétés - Table de variables, cliquez sur l'onglet Variables et ajoutez de nouvelles définitions de variables en cliquant sur l'icône "+" (Ajouter) et en indiquant les noms et valeurs des variables.

Pour plus d'informations, voir Chapitre 6, «Personnalisation de votre charge de travail à l'aide des tables de variables», à la page 121.

Définition de table de variables

Une table de variables est un objet regroupant plusieurs variables. Tous les paramètres globaux (désormais appelés *variables*) que vous utilisez dans une planification de charge de travail sont contenues dans au moins une table de variables. Il existe deux méthodes pour définir des variables :

- Lors de la définition d'une table de variables de la manière présentée ici. Il s'agit de la méthode recommandée.
- Définissez-les individuellement avec la commande **composer $parm** au format [*nom_table.*]*nom_variable "valeur_variable"*. Si vous n'indiquez pas de nom de table, la nouvelle variable est placée dans la table de variables par défaut.

Vous n'êtes pas obligé de créer des tables de variables pour créer et utiliser des variables. Vous pouvez ne jamais créer de table et ne jamais en utiliser de manière explicite. Dans tous les cas, le planificateur fournit une table par défaut et à chaque fois que vous créez et gérez une variable sans préciser le nom de la table, il la stocke ou la recherche ici.

Vous pouvez définir plusieurs variables avec le même nom mais différentes valeurs, et les placer dans des tables différentes. Grâce à ces tables, vous attribuez différentes valeurs à la même variable, et réutilisez donc la même variable dans les définitions de travail et lors de la définition des invites et des dépendances de fichier. Les tables de variables peuvent être affectées au niveau des cycles d'exécution, des flot de travaux et des postes de travail.

Les tables de variables peuvent s'avérer particulièrement utiles pour les définitions de travail, notamment lorsque vous utilisez une définition de travail en tant que modèle pour un travail appartenant à plusieurs flots de travaux. Par exemple, vous pouvez affecter différentes valeurs à la même variable et réutiliser la même définition de travail dans plusieurs flots de travaux.

Pour plus d'informations, voir Chapitre 6, «Personnalisation de votre charge de travail à l'aide des tables de variables», à la page 121.

Syntaxe

vartable *nom_table*
[**description** *"description"*]
[**isdefault**]
members
[*nom_variable "valeur_variable"*]
...
[*nom_variable "valeur_variable"*]
end

Arguments

vartable *nom_table*
Nom de la table de variables. Il doit commencer par une lettre et peut contenir des caractères alphanumériques, des tirets et des traits de soulignement. Il peut contenir jusqu'à 80 caractères.

description "*description_table*"
Description de la table de variables. Ce texte doit être délimité par des guillemets. La description peut contenir jusqu'à 120 caractères alphanumériques. Il ne peut pas comporter les caractères suivants : guillemet (") (autres que ceux encadrant la description), deux-points (:), point-virgule(;) et la perluète (&).

isdefault
Si ce mot clé est indiqué, il s'agit de la table par défaut. Vous ne pouvez pas indiquer plusieurs tables comme table par défaut. Si vous indiquez qu'une table de variables est celle par défaut, la table de variables par défaut en cours ne l'est plus. Lors de la migration de la base de données à partir d'une version précédente, le produit crée la table de variables par défaut avec toutes les variables déjà définies.

members *nom_variable* "*valeur_variable*"
Liste des variables et de leurs valeurs séparées par des espaces. Ce nom peut comporter jusqu'à 16 caractères alphanumériques, incluant les tirets (-) et les traits de soulignement (_), et doit commencer par une lettre. La valeur peut comporter jusqu'à 72 caractères alphanumériques. Une valeur doit être délimitée par des guillemets.

Exemple

L'exemple ci-dessous illustre une table de variables et son contenu.

```
VARTABLE TEST1
  MEMBERS
   DEVBATCH "DOMD\IMSBATCH\SAME"
   PARAM_01 "date"
   PARAM_02 "root"
   PARM_01 "PARM_001"
   PRPT_02 "PARM_002"
   PRPT_03 "PARM_003"
   PRPT_04 "PARM_004"
   PRPT_05 "PARM_005"
   SAME17 "test/for/variable âwith samename > variable/table"
   SLAV10 "/nfsdir/billingprod/crmb/MAESTRO_JOB/AG82STGGDWHSCART"
   SLAV11 "/nfsdir/billingprod/crmb/MAESTRO_JOB/AG82CDMGALLBCV"
   SLAV12 "/nfsdir/billingprod/crmb/MAESTRO_JOB/AG82CDMGRISCTRAF"
   SLAV13 "/opt/crm/DWH_OK/Businness_Copy_ok"
   SLAV14 "/opt/crm/DWH_OK/DW_Canc_Cust_Gior_ok_"
   TRIGGER "/usr/local/samejobtriggers"
   VFILE2 "testforvarwithsamename2.sh"
   VUSER2 "same_user2"
   WRAPPER "/usr/local/sbin/same/phi_job.ksh"
END
```

Considérations relative au fichier de sécurité

Du point de vue des autorisations du fichier de sécurité, les droits d'intervention sur les entrées de variables contenues dans une table de variables dépendent des droits globaux accordés à la table de variables, comme indiqué dans le tableau suivant.

Tableau 47. Mot clé d'accès nécessaire de la table de variables dans le fichier de sécurité (objet vartable) et actions autorisées sur ses entrées de variable.

Mot clé d'accès au fichier de sécurité requis sur la table de variables	Action autorisée sur les entrées de variable répertoriées
Modifier	Ajouter
	Suppression
	Modifier
	Renommer
Afficher	Afficher
Déverrouiller	Déverrouiller

Voir aussi

Pour plus d'informations sur la façon d'effectuer la même tâche à partir de Dynamic Workload Console, voir :

la section "Conception de votre charge de travail" dans le document Dynamic Workload Console - Guide d'utilisation.

Définition d'invite

Une invite est un texte s'affichant sous forme de message et interrompant le traitement d'un travail ou flot de travaux jusqu'à ce qu'une réponse affirmative soit envoyée (la réponse peut être indiquée manuellement par l'opérateur ou automatiquement par une action de règle d'événement). Une fois la réponse à l'invite obtenue, le traitement reprend. Les invites peuvent servir de dépendances dans les travaux ou les flots de travaux. Vous pouvez utiliser des variables dans les invites.

Il existe deux types d'invites :

Invites locales ou non nommées
: Une invite non nommée est une invite définie dans une définition de travail ou de flot de travaux à l'aide du mot clé **prompt**. Aucun nom ne lui est affecté et elle n'est pas définie en tant qu'objet de planification dans la base de données. Elle ne peut donc pas être utilisée par d'autres travaux ou flots de travaux.

Invites globales ou nommées
: Une invite globale est définie dans la base de données en tant qu'objet de planification, identifiée par un nom unique et utilisable par n'importe quel travail ou flot de travaux. Les variables des invites globales sont toujours résolues à l'aide de la table de variables par défaut. En effet, l'invite globale est utilisée par tous les travaux et flots de travaux de manière à n'utiliser qu'une seule valeur pour la résolution des variables.

La présente section décrit les invites globales. Pour plus d'informations sur les invites locales, voir «Travail», à la page 772 et «Définition de flot de travaux», à la page 236.

Remarque : Chaque fois que le travail **JnextPlan** est exécuté, les définitions d'invite globale ou prédéfinie sont réinitialisées.

Syntaxe

$prompt
nom_invite "**[:** | **!]***texte*"

[*nom_invite* ...]

Arguments

nom_invite
Nom de l'invite. Ce nom peut comporter jusqu'à 8 caractères alphanumériques, incluant les tirets (-) et les traits de soulignement (_), et doit commencer par une lettre.

texte

Fournit le texte de l'invite. Le texte de l'invite peut contenir jusqu'à huit caractères alphanumériques. Selon le caractère précédant son libellé, l'invite peut se comporter de manières différentes :

- S'il commence par un signe deux-points (:), l'invite s'affiche mais aucune réponse n'est requise pour la poursuite du processus.
- Si le texte commence par un point d'exclamation (!), l'invite s'affiche, mais il n'est pas enregistré dans le fichier journal.

Un ou plusieurs paramètres peuvent être utilisés pour une partie ou l'ensemble de la chaîne de texte de l'invite. Si un paramètre est utilisé, celui-ci doit être placé entre carets (^). Pour un exemple, voir «Définition des variables et des paramètres», à la page 217.

Remarque : Dans les invites locales, les carets (^) ne désignant pas de paramètre doivent être précédés d'une barre oblique inversée (\) pour ne pas provoquer d'erreurs. Dans les invites globales, il n'est pas nécessaire de faire précéder les carets d'une barre oblique inversée.
Pour créer une ligne, vous pouvez inclure une barre oblique inversée suivie de la lettre n (**\n**) dans le texte.

Exemples

L'exemple suivant permet de définir trois invites :

```
$prompt
     prmt1 "prêt pour job4? (y/n)"
     prmt2 ":job4 lancé"
     prmt3 "!poursuivre?"
```

Voir aussi

Pour plus d'informations sur la façon d'effectuer la même tâche à partir de Dynamic Workload Console, voir :

la section "Conception de votre charge de travail" dans le document Dynamic Workload Console - Guide d'utilisation.

Définition de ressource

Les ressources correspondent à des ressources de planification physiques ou logiques qui peuvent servir de dépendances pour les travaux et les flots de

travaux. Les ressources ne peuvent être utilisées comme dépendances que par les travaux ou les flots de travaux exécutés sur le poste de travail où la ressource est définie.

En raison du mécanisme de résolution des dépendances de ressources, une dépendance de ressource au niveau du flot de travaux peut être considérée comme 'locale' (et son utilisation est prise en charge) et non comme 'globale', lorsqu'à la fois le flot de travaux et tous ses travaux sont définis sur le même poste de travail que la ressource. Cependant, un agent standard et son hôte peuvent faire référence aux mêmes ressources. Pour plus d'informations, voir le mot-clé «needs», à la page 266.

Syntaxe

$resource
*poste_de_travail***#***nom_ressource unités* [*"description"*]

[*poste_de_travail***#***nom_ressource* ...]

Arguments

poste de travail
Indique le nom du poste de travail ou de la classe de postes de travail sur lequel la ressource est utilisée.

nom_ressource
Indique le nom de la ressource. Ce nom peut comporter jusqu'à huit caractères alphanumériques, incluant les tirets (-) et les traits de soulignement (_), et doit commencer par une lettre.

unités Indique le nombre d'unités de ressources disponibles. Les valeurs peuvent être comprises entre **0** et **1024**.

"*description***"**

Fournit une description de la ressource. La description peut contenir jusqu'à 120 caractères alphanumériques. Elle doit figurer entre guillemets.

Les unités de ressource impliquées dans les dépendances needs pour un travail ou un flot de travaux restent occupées jusqu'à ce que le travail ou le flot de travaux soit terminé (qu'il ait ou non abouti). Les unités de ressource sont libérées dès que le travail ou le flot de travaux est terminé.

Lorsque plusieurs travaux et flots de travaux dépendent de la même ressource et que le nombre d'unités actuellement disponibles n'est pas suffisant pour tous ces travaux et flots, la ressource est affectée en fonction de la priorité du travail ou du flot de travaux. Le statut d'un travail ou d'un flot de travaux devient READY (Prêt) dès que toutes ses dépendances sont résolues. Si la quantité limite d'UC définie sur le poste de travail ne lui permet pas de s'exécuter pour le moment, il attend à l'état READY. La seule exception à ce comportement est lorsque le travail ou le flot de travaux a le statut GO ou HI, auquel cas il démarre, quelle que soit la valeur définie pour la quantité limite d'UC.

Exemples

L'exemple suivant permet de définir quatre ressources :

```
$resource
  ux1#tapes 3 "unités de bande"
  ux1#jobslots 24 "emplacements de travaux"
  ux2#tapes 2 "unités de bande"
  ux2#jobslots 16 "emplacements de travaux"
```

Voir aussi

Pour plus d'informations sur la façon d'effectuer la même tâche à partir de Dynamic Workload Console, voir :

la section "Conception de votre charge de travail" dans le document Dynamic Workload Console - Guide d'utilisation.

Définition d'un groupe de cycle d'exécution

Un groupe de cycle d'exécution est un objet de base de données dans lequel un ou plusieurs cycles d'exécution sont définis. Les cycles d'exécution combinés ensemble produisent un ensemble de dates d'exécution pour un flot de travaux. Chaque définition de groupe de cycle d'exécution possède le format et les arguments suivants :

Syntaxe

$runcyclegroup
nom_groupe_cycle_exécution [*"description"*]
  vartable*nom_table*
    [**freedays** *nom_agenda* [**-sa**] [**-su**]]
 [**on** [**runcycle** *nom*]
    [**validfrom** *date*] [**validto** *date*]
    [**description** *"texte"*]
    [**vartable** *nom_table*]
   {*date* | *jour* | *agenda* | *demande* | *"icalendar"*}
[**,...**]
    [**fdignore** | **fdnext** | **fdprev**][**subset** *nom_sous-ensemble* **AND** | **OR**]
    [({**at** *heure* [+*n* **day**[**s**]] |
    schedtime *heure* [+*n* **day**[**s**]]}
    [**until** *heure* [+*n* **day**[**s**]] [**onuntil** *action*]]
    [**deadline** *heure* [+*n* **day**[**s**]]])]
  [**,...**]
  [**except** [**runcycle** *nom*]
      [**validfrom** *date*] [**validto** *date*]
      [**description** *"texte"*]
      {*date* | *jour* | *agenda* | *demande* | *"icalendar"*}
[**,...**]
      [**fdignore** | **fdnext** | **fdprev**][**subset** *nom_sous-ensemble* **AND** | **OR**]
      [{(**at** *heure* [+*n* **day**[**s**]])] |
      (**schedtime** *heure* [+*n* **day**[**s**]])}]
  [**,...**]
  [{**at** *heure* [***timezone*** | ***tz*** *fuseau_horaire*] [+*n* **day**[**s**]] |
  schedtime *heure* [***timezone*** | ***tz*** *fuseau_horaire*] [+*n* **day**[**s**]]}]
  [**until** *heure* [**timezone** | **tz** *fuseau_horaire*] [+*n* **day**[**s**]] [**onuntil** *action*]]
  [**deadline** *heure* [**timezone** | **tz** *fuseau_horaire*] [+*n* **day**[**s**]]]
end

Arguments

nom_groupe_cycle_exécution
Indique le nom du groupe de cycle d'exécution. Ce nom peut comporter jusqu'à huit caractères alphanumériques, incluant les tirets (-) et les traits de soulignement (_), et doit commencer par une lettre.

"*description***"**
Fournit une description du groupe de cycle d'exécution. La description peut contenir jusqu'à 120 caractères alphanumériques. Elle doit figurer entre guillemets. Son nom peut comporter des caractères alphanumériques à condition de commencer par une lettre. Les caractères suivants sont admis : virgule (,), point (.), tiret (-), plus (+), apostrophe (') et égal (=). Il ne peut pas comporter les caractères suivants : guillemet (") (autres que ceux encadrant la description), deux-points (:), point-virgule(;) et la perluète (&).

vartable *nom_table*
Nom de la table de variables. Il doit commencer par une lettre et peut contenir des caractères alphanumériques, des tirets et des traits de soulignement. Il peut contenir jusqu'à 80 caractères.

[freedays *nom_agenda* **[-sa] [-su]]**
Indique un agenda des jours chômés pour le calcul des jours ouvrés correspondant au flot de travaux. Les samedis et dimanches peuvent également être définis comme jours ouvrés.

Nom_agenda
Nom de l'agenda devant être utilisé comme agenda des jours chômés pour le flot de travaux. Si *Nom_agenda* ne se trouve pas dans la base de données, Tivoli Workload Scheduler génère un message d'erreur lorsque vous sauvegardez le flot de travaux. Si *Nom_agenda* ne se trouve pas dans la base de données lors de l'exécution de la commande **schedulr**, Tivoli Workload Scheduler génère un message d'erreur et utilise l'agenda par défaut **holidays** à la place. N'utilisez pas les noms des jours de la semaine pour nommer un agenda.

-sa Le samedi est considéré comme un *jour ouvré*.

-su Le dimanche est considéré comme un *jour ouvré*.

Voir «freedays», à la page 258 pour obtenir des détails et des exemples.

runcycle *nom*
Indique un libellé avec un nom usuel pour le cycle d'exécution spécifié dans les lignes suivantes.

valide de *date* **... valide jusqu'à** *date*
Délimite la période au cours de laquelle le flot de travaux est actif, c'est-à-dire pendant laquelle il est ajouté au plan de production. Notez que la date spécifiée comme valeur **valide jusqu'à** n'est pas incluse dans le cycle d'exécution et le flot de travaux n'est donc pas actif à cette date.

description "*texte***"**
Contient une description du cycle d'exécution.

vartable
Indique le nom de la table de variables que le cycle d'exécution doit utiliser.

date Indique un cycle d'exécution qui s'exécute à des dates spécifiques. La syntaxe utilisée pour ce type de cycle d'exécution est la suivante :

jjmmaaaa [,aaaammjj][,...]Par exemple, pour un flot de travaux dont l'exécution est prévue les 25 mai 2009 et 12 juin 2009, la valeur est la suivante :

```
on
20090525,20090612
```

jour Indique un cycle d'exécution qui s'exécute des jours spécifiques. La syntaxe utilisée pour ce type de cycle d'exécution est la suivante :

{mo | tu | we | th | fr | sa | su}Par exemple, pour un flot de travaux qui est planifié pour s'exécuter chaque lundi, la valeur est la suivante :

```
on
mo
```

agenda Dates indiquées dans un agenda portant ce nom. Le nom de l'agenda peut être suivi d'un décalage au format suivant :

{**+** | **-**}*n* {**day[s]** | **weekday[s]** | **workday[s]**}

où :

n Indique le nombre de jours, jours de la semaine ou jours ouvrés.

jours Chaque jour de la semaine.

weekdays
Tous les jours de la semaine sauf le samedi et le dimanche.

workdays
Tous les jours de la semaine excepté les samedis et dimanches (sauf indication contraire par le mot clé **freedays**) et les dates figurant dans un agenda désigné des jours chômés ou dans l'agenda **holidays**.

demande
Sélectionne le flot de travaux uniquement lorsque celui-ci est demandé. Ce paramètre est utilisé pour les flots de travaux qui sont sélectionnés par nom et non par date. Pour empêcher un flot de travaux planifié d'être sélectionné pour le travail **JnextPlan**, redéfinissez-le ON REQUEST.

Remarque : Si vous tentez d'exécuter un flot de travaux contenant plusieurs heures "à la demande" (on request), tenez compte des points suivants :

- Les indicateurs "On request" ont la priorité sur les indicateurs "at".
- Les indicateurs "On request" n'ont jamais la priorité sur les indicateurs "On".

icalendar
Représente une norme utilisée pour spécifier une règle récurrente qui décrit le moment où un flot de travaux s'exécute.

La syntaxe utilisée pour un cycle d'exécution de type *icalendar* est la suivante :

FREQ={DAYLY | WEEKLY | MONTHLY | YEARLY}

[;INTERVAL=[-]*n*]

[;{BYFREEDAY | BYWORKDAY | BYDAY=*liste_jours_semaine* |

BYMONTHDAY=*liste_jours_mois***}]**

où la valeur par défaut du mot clé **INTERVAL** est 1.

A l'aide du mot clé *icalendar*, vous pouvez spécifier qu'un flot de travaux s'exécute :

tous les *n* **jours**
en utilisant le format suivant :

FREQ=DAILY[;INTERVAL=*n*]

où la valeur définie pour **valide de** est le premier jour des dates résultantes.

Par exemple, pour un flot de travaux planifié pour s'exécuter chaque jour, la valeur est la suivante :

```
FREQ=DAILY
```

Pour un flot de travaux planifié pour s'exécuter un jour sur deux, la valeur est la suivante :

```
FREQ=DAILY;INTERVAL=2
```

chaque jour chômé ou ouvré
en utilisant le format suivant :

FREQ=DAILY[;INTERVAL=*n*]

;BYFREEDAY|BYWORKDAY

Par exemple, pour un flot de travaux planifié pour s'exécuter chaque jour chômé, la valeur est la suivante :

```
FREQ=DAILY;BYFREEDAY
```

Pour un flot de travaux planifié pour s'exécuter un jour ouvré sur deux, la valeur est la suivante :

```
FREQ=DAILY;INTERVAL=2;BYWORKDAY
```

toutes les *n* **semaines, à des** *jours de la semaine* **spécifiques**
en utilisant le format suivant :

FREQ=WEEKLY[;INTERVAL=*n*]

;BYDAY=*liste_jours_semaine*

où la valeur définie pour *liste_jours_semaine* est

```
[SU][,MO][,TU][,WE][,TH][,FR][,SA]
```

Par exemple, pour un flot de travaux planifié pour s'exécuter chaque semaine le vendredi et le samedi, la valeur est la suivante :

```
FREQ=WEEKLY;BYDAY=FR,SA
```

Pour un flot de travaux planifié pour s'exécuter toutes les trois semaines, le vendredi, la valeur est la suivante :

```
FREQ=WEEKLY;INTERVAL=3;BYDAY=FR
```

tous les *n* **mois, à des dates spécifiques**
en utilisant le format suivant :

FREQ=MONTHLY[;INTERVAL=*n*]

;BYMONTHDAY=*liste_jours_mois*

où la valeur définie pour *liste_jours_mois* est représentée par une liste de

[+*nombre_de_jours_à_partir_début_du_mois*]
[-*nombre_de_jours_à_partir_fin_du_mois*]
[*nombre_de_jours_du_mois*]

Par exemple, pour un flot de travaux planifié pour s'exécuter le 27ème jour de chaque mois, la valeur est la suivante :

```
FREQ=MONTHLY;BYMONTHDAY=27
```

Pour un flot de travaux planifié pour s'exécuter tous les six mois, le 15 du mois et le dernier jour du mois, la valeur est la suivante :

```
FREQ=MONTHLY;INTERVAL=6;BYMONTHDAY=15,-1
```

tous les *n* mois, à des jours spécifiques de la semaine
en utilisant le format suivant :

FREQ=MONTHLY[;INTERVAL=*n*]

;BYDAY=*numéro_de_la_semaine_du_mois_jour_semaine*

où la valeur définie pour *numéro_de_la_semaine_du_mois_jour_semaine* est représentée par une liste de

[+*numéro_semaine_à_partir_début_du_mois*]
[-*numéro_semaine_à_partir_fin_du_mois*]
[*jour_semaine*]

Par exemple, pour un flot de travaux planifié pour s'exécuter chaque mois, le premier lundi et le dernier vendredi du mois, la valeur est la suivante :

```
FREQ=MONTHLY;BYDAY=1MO,-1FR
```

Pour un flot de travaux planifié pour s'exécuter tous les six mois, le deuxième mardi du mois, la valeur est la suivante :

```
FREQ=MONTHLY;INTERVAL=6;BYDAY=2TU
```

toutes les *n* années
en utilisant le format suivant :

FREQ=YEARLY[;INTERVAL=*n*]

où la valeur définie pour **valide de** est le premier jour des dates résultantes.

Par exemple, pour un flot de travaux planifié pour s'exécuter chaque année, la valeur est la suivante :

```
FREQ=YEARLY
```

Pour un flot de travaux planifié pour s'exécuter tous les deux ans, la valeur est la suivante :

```
FREQ=YEARLY;INTERVAL=2
```

fdignore | fdnext | fdprev
Indique la règle à appliquer si la date sélectionnée pour l'exécution du travail ou du flot de travaux coïncide avec un jour chômé. Les valeurs possibles sont les suivantes :

fdignore
La date n'est pas ajoutée.

fdnext Le jour ouvré le plus proche qui suit le jour chômé est ajouté.

fdprev
Le jour ouvré le plus proche précédant le jour chômé est ajouté.

[subset *nom_sous-ensemble* **AND|OR]**

subset *nom_sous-ensemble*
Indique le nom du sous-ensemble. Si vous ne spécifiez pas de nom, SUBSET_1 est utilisé par défaut.

AND|OR
Par défaut, les cycles d'exécution dans un sous-ensemble sont dans une relation logique **OR** mais vous pouvez la changer en relation logique **AND** tant que le résultat du groupe de cycle d'exécution est une date ou un ensemble de dates positif (inclusif).

runcycle *nom*
Indique un libellé avec un nom usuel pour le cycle d'exécution spécifié dans les lignes suivantes.

valide de *date* **... valide jusqu'à** *date*
Délimite la période au cours de laquelle le flot de travaux est actif, c'est-à-dire pendant laquelle il est ajouté au plan de production. Notez que la date spécifiée comme valeur **valide jusqu'à** n'est pas incluse dans le cycle d'exécution et le flot de travaux n'est donc pas actif à cette date.

description ***"****texte****"***
Contient une description du cycle d'exécution.

date Indique un cycle d'exécution qui s'exécute à des dates spécifiques. La syntaxe utilisée pour ce type de cycle d'exécution est la suivante :

jjmmaaaa [,aaaammjj][,...]Par exemple, pour un flot de travaux dont l'exécution est prévue les 25 mai 2009 et 12 juin 2009, la valeur est la suivante :

```
on
20090525,20090612
```

jour Indique un cycle d'exécution qui s'exécute des jours spécifiques. La syntaxe utilisée pour ce type de cycle d'exécution est la suivante :

{mo|tu|we|th|fr|sa|su}Par exemple, pour un flot de travaux qui est planifié pour s'exécuter chaque lundi, la valeur est la suivante :

```
on
mo
```

agenda Dates indiquées dans un agenda portant ce nom. Le nom de l'agenda peut être suivi d'un décalage au format suivant :

{**+** | **-**}*n* {**day**[**s**] | **weekday**[**s**] | **workday**[**s**]}

Où :

n Indique le nombre de jours, jours de la semaine ou jours ouvrés.

jours Chaque jour de la semaine.

weekdays
Tous les jours de la semaine sauf le samedi et le dimanche.

workdays
Tous les jours de la semaine excepté les samedis et dimanches (sauf indication contraire par le mot clé **freedays**) et les dates figurant dans un agenda désigné des jours chômés ou dans l'agenda **holidays**.

demande
Sélectionne le flot de travaux uniquement lorsque celui-ci est demandé. Ce

paramètre est utilisé pour les flots de travaux qui sont sélectionnés par nom et non par date. Pour empêcher un flot de travaux planifié d'être sélectionné pour le travail **JnextPlan**, redéfinissez-le ON REQUEST.

Remarque : Si vous tentez d'exécuter un flot de travaux contenant plusieurs heures "à la demande" (on request), tenez compte des points suivants :

- Les indicateurs "On request" ont la priorité sur les indicateurs "at".
- Les indicateurs "On request" n'ont jamais la priorité sur les indicateurs "On".

icalendar

Représente une norme utilisée pour spécifier une règle récurrente qui décrit le moment où un flot de travaux s'exécute.

La syntaxe utilisée pour un cycle d'exécution de type *icalendar* est la suivante :

FREQ={DAYLY | WEEKLY | MONTHLY | YEARLY}

[;INTERVAL=[-]*n*]

[;{BYFREEDAY | BYWORKDAY | BYDAY=*liste_jours_semaine* |

BYMONTHDAY=*liste_jours_mois***}]**

où la valeur par défaut du mot clé **INTERVAL** est 1.

A l'aide du mot clé *icalendar*, vous pouvez spécifier qu'un flot de travaux s'exécute :

tous les *n* jours
en utilisant le format suivant :

FREQ=DAILY[;INTERVAL=*n*]

où la valeur définie pour **valide de** est le premier jour des dates résultantes.

Par exemple, pour un flot de travaux planifié pour s'exécuter chaque jour, la valeur est la suivante :

```
FREQ=DAILY
```

Pour un flot de travaux planifié pour s'exécuter un jour sur deux, la valeur est la suivante :

```
FREQ=DAILY;INTERVAL=2
```

chaque jour chômé ou ouvré
en utilisant le format suivant :

FREQ=DAILY[;INTERVAL=*n*]

;BYFREEDAY | BYWORKDAY

Par exemple, pour un flot de travaux planifié pour s'exécuter chaque jour chômé, la valeur est la suivante :

```
FREQ=DAILY;BYFREEDAY
```

Pour un flot de travaux planifié pour s'exécuter un jour ouvré sur deux, la valeur est la suivante :

```
FREQ=DAILY;INTERVAL=2;BYWORKDAY
```

toutes les *n* **semaines, à des** *jours de la semaine* **spécifiques**
en utilisant le format suivant :

FREQ=WEEKLY[;INTERVAL=*n*]

;BYDAY=*liste_jours_semaine*

où la valeur définie pour *liste_jours_semaine* est
`[SU][,MO][,TU][,WE][,TH][,FR][,SA]`

Par exemple, pour un flot de travaux planifié pour s'exécuter chaque semaine le vendredi et le samedi, la valeur est la suivante :
```
FREQ=WEEKLY;BYDAY=FR,SA
```

Pour un flot de travaux planifié pour s'exécuter toutes les trois semaines, le vendredi, la valeur est la suivante :
```
FREQ=WEEKLY;INTERVAL=3;BYDAY=FR
```

tous les *n* **mois, à des dates spécifiques**
en utilisant le format suivant :

FREQ=MONTHLY[;INTERVAL=*n*]

;BYMONTHDAY=*liste_jours_mois*

où la valeur définie pour *liste_jours_mois* est représentée par une liste de
```
[+nombre_de_jours_à_partir_début_du_mois]
[-nombre_de_jours_à_partir_fin_du_mois]
[nombre_de_jours_du_mois]
```

Par exemple, pour un flot de travaux planifié pour s'exécuter le 27ème jour de chaque mois, la valeur est la suivante :
```
FREQ=MONTHLY;BYMONTHDAY=27
```

Pour un flot de travaux planifié pour s'exécuter tous les six mois, le 15 du mois et le dernier jour du mois, la valeur est la suivante :
```
FREQ=MONTHLY;INTERVAL=6;BYMONTHDAY=15,-1
```

tous les *n* **mois, à des jours spécifiques de la semaine**
en utilisant le format suivant :

FREQ=MONTHLY[;INTERVAL=*n*]

;BYDAY=*numéro_de_la_semaine_du_mois_jour_semaine*

où la valeur définie pour *numéro_de_la_semaine_du_mois_jour_semaine* est représentée par une liste de
```
[+numéro_semaine_à_partir_début_du_mois]
[-numéro_semaine_à_partir_fin_du_mois]
[jour_semaine]
```

Par exemple, pour un flot de travaux planifié pour s'exécuter chaque mois, le premier lundi et le dernier vendredi du mois, la valeur est la suivante :
```
FREQ=MONTHLY;BYDAY=1MO,-1FR
```

Pour un flot de travaux planifié pour s'exécuter tous les six mois, le deuxième mardi du mois, la valeur est la suivante :
```
FREQ=MONTHLY;INTERVAL=6;BYDAY=2TU
```

toutes les *n* **années**
en utilisant le format suivant :

FREQ=YEARLY[;INTERVAL=*n*]

où la valeur définie pour **valide de** est le premier jour des dates résultantes.

Par exemple, pour un flot de travaux planifié pour s'exécuter chaque année, la valeur est la suivante :

```
FREQ=YEARLY
```

Pour un flot de travaux planifié pour s'exécuter tous les deux ans, la valeur est la suivante :

```
FREQ=YEARLY;INTERVAL=2
```

fdignore | fdnext | fdprev
Indique une règle qui doit être appliquée lorsque la date devant être exclue correspond à un jour chômé. Il peut s'agir de l'une des règles suivantes :

fdignore
La date n'est pas exclue.

fdnext Le jour ouvré le plus proche suivant le jour chômé est exclu.

fdprev
Le jour ouvré le plus proche précédant le jour chômé est exclu.

subset *nom_sous-ensemble*
Indique le nom du sous-ensemble. Si vous ne spécifiez pas de nom, SUBSET_1 est utilisé par défaut.

AND | OR
Par défaut, les cycles d'exécution dans un sous-ensemble sont dans une relation logique **OR** mais vous pouvez la changer en relation logique **AND** tant que le résultat du groupe de cycle d'exécution est une date ou un ensemble de dates positif (inclusif).

Exemple

L'exemple suivant définir un groupe de cycle d'exécution, RCG2, qui contient un cycle d'exécution inclusif, RUN_CYCLE1, et deux cycles d'exécution inclusifs, RUN_CYCLE2 et RUN_CYCLE3. Pour déterminer le calendrier d'exécution du flot de travaux associé à ce groupe de cycle d'exécution, l'intersection des deux cycles de diffusion et en exclusivité (les deux cycles d'exécution exclusifs ont une relation ET logique entre elles) est soustrait du cycle d'exécution inclusif. Voici les caractéristiques du groupe de cycle d'exécution :

Cycle d'exécution inclusif RUN_CYCLE1
où :

- Le calendrier CAL1 définit les jours considérés comme chômés pour le flot de travaux. Samedi et dimanche sont considérés comme des jours ouvrés.
- Le flot de travail ne s'exécute pas avant deux jours après le 31 mars 2008 (2 avril) et pas plus tard que deux jours après le 12 avril 2008 (14 avril). Chaque jour, le flot de travaux est retardé de deux jours.
- Les flots de travaux s'exécutent tous les jours (après un retard de deux jours) à partir de 7 h du matin mais pas après 9 h, auquel cas le flot de travaux est supprimé sans être exécuté. Le flot de travaux doit se terminer à 10 h du matin.

Cycle d'exécution exclusif RUN_CYCLE2
Si le flot de travaux tombe un jour chômé, le jour ouvrable le plus proche avant le jour chômé est exclu.

Si le flot de travaux tombe le 1er avril 2008 et que ce jour se trouve être chômé, le jour ouvrable le plus proche après le jour chômé est exclu.

Cycle d'exécution exclusif RUN_CYCLE3
Si le flot de travaux tombe le 1er avril 2008 et que ce jour se trouve être chômé, le jour ouvrable le plus proche après le jour chômé est exclu.

```
RUNCYCLEGROUP RCG2
DESCRIPTION "Exemple de groupe de cycle d'exécution"
VARTABLE TABLE1
FREEDAYS CAL1 -SA -SU
  ON     RUNCYCLE RUN_CYCLE1 VALIDFROM 03/31/2008 VALIDTO 04/12/2008 DESCRIPTION
         "Inclusive Run Cycle" VARTABLE TABLE1 "FREQ=DAILY;" FDIGNORE
         (AT 0700 +2 DAYS UNTIL 0900 +2 DAYS ONUNTIL SUPPR DEADLINE 1000 +2 DAYS)

  EXCEPT RUNCYCLE RUN_CYCLE2 VALIDFROM 03/31/2008 VALIDTO 04/12/2008 DESCRIPTION
         "Exclusive Run Cycle" CAL1 FDPREV SUBSET SUBSET_A AND
         (AT 0700 +2 DAYS)

  EXCEPT RUNCYCLE RUN_CYCLE3 VALIDFROM 03/31/2008 VALIDTO 04/12/2008 DESCRIPTION
         "Exclusive Run Cycle" 04/01/2008 FDNEXT SUBSET SUBSET_A AND
         (SCHEDTIME 0700 +2 DAYS)
SCHEDTIME 0700 TZ Europe/Berlin +2 DAYS UNTIL 0900 TZ Europe/Berlin +2 DAYS ONUNTIL
          CONT DEADLINE 1000 TZ Europe/Berlin +2 DAYS
END
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle décrite dans :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Création de définitions de flot de travaux".

Définition de flot de travaux

Un flot de travaux comprend une séquence des travaux à exécuter, ainsi que des horaires, des priorités et d'autres dépendances qui déterminent l'ordre de traitement.

Un flot de travaux commence avec un mot clé **schedule** suivi par des attributs et des dépendances. Le signe deux-points (:) en tant que délimiteur introduit les travaux appelés par le flot de travaux. Chaque travail possède ses attributs et dépendances.

Syntaxe

schedule [*poste_de_travail*#]*nom_flot_travaux*
commentaire
[**validfrom** *date*]
[**timezone**|**tz** *fuseau_horaire*]
[**description** *"texte"*]
[**draft**]
[**vartable** *nom_table*]
[**freedays** *nom_agenda* [**-sa**] [**-su**]]
[**on** [**runcycle** *nom*]

```
    [validfrom date] [validto date]
    [description "texte"]
    [vartable nom_table]
   {date | jour | agenda | demande | "icalendar" | groupe_cycle_exécution} [,...]
    [fdignore | fdnext | fdprev]
    [({at heure [+n day[s]] |
    schedtime heure [+n day[s]]}
    [until heure [+n day[s]] [onuntil action]]
    [deadline heure [+n day[s]]])]
  [,...]
  [except [runcycle nom]
      [validfrom date] [validto date]
      [description "texte"]
      {date | jour | agenda | demande | "icalendar" | groupe_cycle_exécution} [,...]
      [fdignore | fdnext | fdprev]
      [{(at heure [+n day[s]])] |
      (schedtime heure [+n day[s]])}]
  [,...]
  [{at heure [timezone | tz fuseau_horaire] [+n day[s]] |
  schedtime heure [timezone | tz fuseau_horaire] [+n day[s]]}]
  [until heure [timezone | tz fuseau_horaire] [+n day[s]] [onuntil action]]
  [deadline heure [timezone | tz fuseau_horaire] [+n day[s]]]
  [carryforward]
  [matching {previous | sameday | relative from [+ | -] heure to [+ | -] heure |
    from heure [+ | -n day[s]] to heure [+ n day[s]] [,...]}]
  [follows {[agent_réseau::][poste_de_travail#]nom_flot_travaux[.nom_travail |@] [previous |
    sameday | relative from [+ | -] heure to [+ | -] heure |
    from heure [+ | -n day[s]] to heure [+ | -n day[s]]
     ]} ] [,...]] [...]
  [keysched]
  [limit nombre_max_travaux]
  [needs { [n] [poste_de_travail#]nom_ressource } [,...] ] [...]
  [opens { [poste_de_travail#]"file_name" [ (qualifiant) ] [,...] }] [...]
  [priority nombre | hi | go]
  [prompt {nom_invite | "[: | !]texte"} [,...] ] [...]
 :
instruction-travail
  # commentaire
  [{at heure [timezone | tz fuseau_horaire] [+n day[s]] |
  schedtime heure [timezone | tz fuseau_horaire] [+n day[s]]}][,...]
  [until heure [timezone | tz fuseau_horaire] [+n day[s]] [onuntil action]
  [deadline heure [timezone | tz fuseau_horaire] [+n day[s]]] [,...]
  [maxdur heure | pourcentage % onmaxdur action]
  [mindur heure | pourcentage % onmindur action]
  [every fréquence]
  [follows {[agent_réseau::][poste_de_travail#]nom_flot_travaux{.nom_travail @} [previous |
    sameday | relative from [+ | -] heure to [+ | -] heure |
    from heure [+ | -n day[s]] to heure [+ | -n day[s]]
     ]} ] [,...]] [...]
  [confirmed]
  [critical]
  [keyjob]
  [needs { [n] [poste_de_travail#]nom_ressource } [,...] ] [...]
  [opens { [poste_de_travail#]"file_name" [ (qualifiant) ] [,...] }] [...]
  [priority nombre | hi | go]
  [prompt {nom_invite | "[: | !]texte"} [,...] ] [...]
```

[*instruction-travail*...]
end

Arguments

Le tableau 48 contient une brève description des mots clés des définitions de flots de travaux. Vous trouverez une description détaillée de chaque mot clé de planification dans les sous-sections suivantes.

Tableau 48. Liste des mots clés de planification

Mot clé	Description	Page
at	Définit la première heure à laquelle l'exécution d'un travail ou d'un flot de travaux peut être lancée. Lorsqu'il est défini dans un cycle d'exécution, ce mot clé spécifie la première heure à laquelle un travail ou un flot de travaux peut être lancé pour ce cycle d'exécution spécifique.	«at», à la page 242
carryforward	Reporte le flot de travaux lorsque celui-ci n'est pas terminé.	«carryforward», à la page 243
comment	Inclut des commentaires dans la définition d'un flot de travaux ou dans un travail appartenant au flot de travaux.	«comment», à la page 244
confirmed	Indique qu'une confirmation de l'état d'achèvement du travail est nécessaire.	«confirmed», à la page 244
critical	Permet d'indiquer que le travail a une importance critique et doit donc être géré en priorité.	«critical», à la page 244
deadline	Indique le temps nécessaire pour l'achèvement d'un travail ou d'un flot de travaux. Lorsqu'il est défini dans un cycle d'exécution, ce mot clé spécifie le délai dans lequel un travail ou un flot de travaux doit être terminé pour ce cycle d'exécution spécifique.	«deadline», à la page 245
description	Contient une description du flot de travaux. La longueur maximale de cette zone est 120 caractères.	«description», à la page 246
draft	Indique que le processus de génération du plan doit ignorer ce flot de travaux.	«draft», à la page 246
end	Marque la fin d'un flot de travaux.	«end», à la page 247
every	Lance le travail de manière répétitive à une fréquence précise.	«every», à la page 247
except	Spécifie les dates **on** où le flot de travaux ne sera pas exécuté.	«except», à la page 251
fdignore | fdnext | fdprev	Indique une règle qui doit être appliquée lorsque la date devant être exclue correspond à un jour chômé.	«except», à la page 251
follows	Indique les travaux ou flots de travaux qui doivent aboutir avant le lancement du travail ou du flot de travaux en cours de définition.	«follows», à la page 255

Tableau 48. Liste des mots clés de planification (suite)

Mot clé	Description	Page
freedays	Indique un agenda des jours chômés pour le calcul des *jours ouvrés* correspondant au flot de travaux. Les samedis et dimanches peuvent également être définis comme *jours ouvrés*.	«freedays», à la page 258
job statement	Définit un travail et les dépendances qui lui sont associées.	«job statement», à la page 259
keyjob	Qualifie un travail comme étant clé dans la base de données et dans le plan, afin qu'il soit surveillé par différentes applications, notamment IBM Tivoli Business Systems Manager ou IBM Tivoli Enterprise Console.	«keyjob», à la page 261
keysched	Qualifie un flot de travaux comme étant clé dans la base de données et dans le plan, afin qu'il soit surveillé par différentes applications, notamment IBM Tivoli Business Systems Manager ou IBM Tivoli Enterprise Console.	«keysched», à la page 261
limit	Définit le nombre maximal de travaux qui peuvent être lancés simultanément à partir de ce flot de travaux.	«limit», à la page 262
matching	Définit les critères de correspondance utilisés lorsqu'aucun critère de correspondance n'est spécifié dans les spécifications prédécesseur/successeur de la définition du flot de travaux ou de la définition de travail dans le flot de travaux.	«matching», à la page 262
maxdur	Indique la durée maximale d'exécution d'un travail. Vous pouvez exprimer cette durée en minutes, ou sous la forme d'un pourcentage de la durée estimée du travail la plus récente.	«maxdur», à la page 263
mindur	Indique la durée la plus courte au cours de laquelle un travail s'exécute normalement et se termine.	«mindur», à la page 265
needs	Définit le nombre d'unités d'une ressource requises par ce travail ou ce flot de travaux pour que celui-ci puisse être lancé. Le plus grand nombre de ressources dont le flot de travaux peut être dépendant est 1024.	«needs», à la page 266
on	Définit les dates d'exécution du flot de travaux.	«on», à la page 267
opens	Définit les ressources qui doivent être disponibles avant le lancement d'un travail ou d'un flot de travaux.	«opens», à la page 273
onuntil	Indique l'action à effectuer sur un travail ou flot de travaux dont l'heure until a été atteinte.	«until», à la page 280

Tableau 48. Liste des mots clés de planification (suite)

Mot clé	Description	Page
priority	Définit la priorité d'un travail ou d'un flot de travaux.	«priority», à la page 275
prompt	Définit les invites qui doivent avoir reçu une réponse pour que le travail ou le flot de travaux puisse être lancé.	«prompt», à la page 276
runcycle	Indique un libellé avec un nom usuel pour le cycle d'exécution.	• «except», à la page 251 • «on», à la page 267
schedule	Attribue un nom au flot de travaux.	«schedule», à la page 279
schedtime	Indique l'heure utilisée pour définir le flot de travaux dans le diagramme chronologique du plan afin de déterminer les successeurs et les prédécesseurs.	«schedtime», à la page 277
timezone | tz	Indique le fuseau horaire à utiliser lors du calcul de l'heure de début.	«timezone», à la page 280
until	Définit une dernière heure à laquelle un travail ou un flot de travaux peut être lancé. Lorsqu'il est défini dans un cycle d'exécution, ce mot clé spécifie la dernière heure à laquelle un travail ou un flot de travaux peut être lancé pour ce cycle d'exécution spécifique.	«until», à la page 280
validfrom	Définit la date à laquelle l'instance du flot de travaux démarre.	«validfrom/validto», à la page 283
validto	Définit la date de fin de l'instance de flot de travaux.	«validfrom/validto», à la page 283
vartable	Définit la table de variables que le flot de travaux et le cycle d'exécution doivent utiliser.	«vartable», à la page 284

Remarque :

1. Les flots de travaux planifiés pour être exécutés sur des postes de travail marqués comme *ignorés* ne sont pas ajoutés au plan de production lorsque le plan est créé ou étendu.
2. Les mots clés saisis incorrectement dans les définitions de travail entraînent le stockage de définitions de travaux tronquées dans la base de données. En fait, le mot clé erroné est considéré comme externe à la définition de travail et il est interprété comme le nom de travail d'une définition de travail supplémentaire. Généralement, cette mauvaise interprétation entraîne également une erreur de syntaxe ou une erreur de définition de travail inexistant pour la définition de travail supplémentaire.

Règles de spécification du fuseau horaire

Vous pouvez indiquer un fuseau horaire à différents niveaux de mot clé dans une définition de flot de travaux. En d'autres termes :

- Pour l'ensemble du flot de travaux (y compris toutes ses spécifications de mot clé)
- Au niveau de restriction temporelle (avec les mots clés `at`, `deadline`, `schedtime` et `until`)

- Pour chaque instruction de travail incluse

Les règles ci-dessous s'appliquent lors de la résolution des fuseaux horaires spécifiés dans une définition de flot de travaux :

- Lorsque vous indiquez le fuseau horaire au niveau du flot de travaux, cela s'applique aux définitions temporelles du cycle d'exécution (définies avec le mot clé `on`) et à celles des restrictions temporelles.
- Si vous indiquez un fuseau horaire au niveau du flot de travaux et de la restriction temporelle, ils doivent être identiques. Si vous n'indiquez aucun fuseau horaire, tant au niveau du flot de travaux que de la restriction temporelle, le fuseau horaire spécifié sur le poste de travail est utilisé.
- Le fuseau horaire indiqué au niveau du travail peut être différent de celui indiqué au niveau du flot de travaux et peut le remplacer. Si vous n'indiquez aucun fuseau horaire, tant au niveau du flot de travaux que du travail, le fuseau horaire spécifié sur le poste qui exécute le travail est utilisé.

Règles de spécification de la restriction temporelle

Dans une définition de flot de travaux, vous pouvez spécifier des restrictions temporelles (avec les mots clés `at`, `deadline`, `schedtime` et `until`) au niveau du flot de travaux et du cycle d'exécution. Si ces deux restrictions sont spécifiées, les restrictions temporelles au niveau du cycle d'exécution remplacent celles spécifiées au niveau du flot de travaux.

Exemples

Voici un exemple de définition de flot de travaux :

```
SCHEDULE M235062_99#SCHED_FIRST1 VALIDFROM 06/30/2005
ON RUNCYCLE SCHED1_PREDSIMPLE VALIDFROM 07/18/2005 "FREQ=DAILY;INTERVAL=1"
   ( AT 1010 )
ON RUNCYCLE SCHED1_PRED_SIMPLE VALIDFROM 07/18/2005 "FREQ=DAILY;INTERVAL=1"
CARRYFORWARD
PROMPT "parto o no?"
PRIORITY 55
:
M235062_99#JOBMDM
 PRIORITY 30
 NEEDS 16 M235062_99#JOBSLOTS
 PROMPT PRMT3

B236153_00#JOB_FTA
 FOLLOWS JOBMDM
END
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle décrite dans :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Création de définitions de flot de travaux".

Détail des mots clés pour la définition des flots de travaux

Cette section décrit les mots clés des définitions de flots de travaux répertoriés dans le tableau 48, à la page 238.

at

Indique une dépendance horaire. Si le mot clé **at** est utilisé, le travail ou le flot de travaux ne peut pas démarrer avant l'heure définie dans ce mot clé.

Syntaxe

at *heure* [**timezone** | **tz** *fuseau_horaire*][+*n* **jour**[**s**]] [**absolute** | **abs**]

Arguments

heure Indique une heure. Les valeurs admises sont comprises entre **0000** et **2359**.

tzname Indique le fuseau horaire à utiliser lors du calcul de l'heure de début. Voir Chapitre 16, «Gestion des fuseaux horaires», à la page 653 pour les noms des fuseaux horaires. Il s'agit par défaut du fuseau horaire correspondant au poste de travail sur lequel le travail ou le flot de travaux est lancé.

Remarque : Si des heures **at** et **until** ou **deadline** sont indiquées, les fuseaux horaires doivent être identiques.

n Indique un décalage, en jours, à partir de la date et de l'heure de début planifiées.

absolute
Indique que la date de début est fonction du jour calendaire plutôt que du jour de production.

Commentaires

Si l'heure n'est pas spécifiée à l'aide du mot clé **at** pour un travail ou un flot de travaux, l'heure de lancement est déterminée par ses dépendances et sa priorité. En outre, sa position dans le plan de préproduction est déterminée par la valeur affectée au mot clé **schedtime**. Pour plus d'informations sur le mot clé **schedtime**, voir «schedtime», à la page 277.

Si les heures de début du cycle d'exécution et du flot de travaux sont définies, celle du cycle d'exécution l'emporte lorsque le flot de travaux est planifié avec **JNextPlan**. Si le flot de travaux est lancé avec la commande **submit**, l'heure de début du cycle d'exécution n'est pas utilisée.

La valeur de temps dans l'option **at** est considérée comme suit :

- Si la valeur de temps est inférieure à l'heure définie dans l'option globale ***startOfDay***, elle est prise en compte pour le jour suivant.
- Si la valeur de temps est supérieure à l'heure définie dans l'option globale ***startOfDay***, elle est prise en compte pour le jour courant.

Si le gestionnaire de domaine maître de votre réseau s'exécute lorsque la valeur `yes` est attribuée aux options `enLegacyStartOfDayEvaluation` et `enTimeZone` afin de convertir l'heure `startOfDay` du gestionnaire de domaine maître en fuseau horaire local sur chaque poste de travail du réseau, vous devez ajouter le mot clé **absolute** pour le faire fonctionner lors de la soumission d'un travail ou d'un flot de travaux.

Si aucun des mots clés **at** et **schedtime** n'est spécifié dans la définition du flot de travaux, l'instance de travail ou de flot de travaux est positionnée par défaut dans le plan à l'heure indiquée dans l'option globale *startOfDay*.

Exemples

Dans les exemples suivants, on considère que le jour de traitement Tivoli Workload Scheduler commence à 6 heures du matin.

- Comme le flot de travaux suivant doit être exécuté tous les mardis, il est lancé au plus tôt le mercredi à 3 heures du matin. Les deux travaux qu'il contient sont lancés dès que possible après cette heure.

```
schedule sked7 on tu at 0300:
job1
job2
end
```

- L'exemple ci-dessous lance le flot de travaux mysked tous les dimanches à 8:00. Les travaux job1, job2 et job3 sont tous lancés le dimanche.

```
schedule mysked on fr at 0800 + 2 days
:
job1
job2 at 0900
job3 follows job2 at 1200
end
```

- Le fuseau horaire du poste de travail `sfran` correspond à l'heure du pacifique America/Los_Angeles et le fuseau horaire du poste de travail `nycity` correspond à l'heure de la côte Est America/New_York. Le flot de travaux suivant est sélectionné pour s'exécuter le vendredi. Il est lancé sur le poste de travail `sfran` à 10 heures du matin sur le fuseau horaire America/Los_Angeles le samedi. Le travail `job1` est lancé sur `sfran` dès que possible après cette heure. Le travail `job2` est lancé sur `sfran` à 14 heures sur le fuseau horaire America/New_York (11:00 a.m. America/Los_Angeles) le samedi. Le travail `job3` est lancé sur le poste de travail `nycity` à 16 heures sur le fuseau horaire America/New_York (1:00 p.m. America/Los_Angeles) le samedi.

```
sfran#schedule sked8 on fr at 1000 + 1 day:
job1
job2 at 1400 tz America/New_York
nycity#job3 at 1600
end
```

carryforward

Autorise le report d'un flot de travaux au plan de production suivant, s'il n'est pas terminé avant la fin du plan de production courant.

Syntaxe

carryforward

Exemples

Le flot de travaux suivant est reporté si les travaux qu'il contient ne sont pas terminés avant le début du processus de préproduction de la période de production suivante.

```
schedule sked43 on th
carryforward
:
job12
job13
job13a
end
```

comment

Inclut des commentaires dans une définition de flot de travaux et dans les travaux contenus dans un flot de travaux.

Syntaxe

*texte*

Commentaires

Insère une ligne de commentaire. Le premier caractère de la ligne doit être un signe dièse (#).

Vous pouvez ajouter des commentaires à une définition de flots de travaux immédiatement après la ligne qui contient le mot clé **schedule** ou dans un travail contenu dans une définition de flot de travaux immédiatement après la ligne *job statement*.

Exemples

L'exemple suivant contient les deux types de commentaire :

```
schedule wkend on fr at 1830
#########################
# Les travaux de nettoyage hebdomadaires
#########################
#
carryforward
:
job1
 # totaux et rapports finals
job2
 # mettre à jour la base de données
end
```

confirmed

Indique que la fin d'un travail doit être confirmée par l'exécution d'une commande **confirm** de **conman**. Pour plus d'informations, voir «confirm», à la page 409.

Syntaxe

confirmed

Exemples

Dans le flot de travaux suivant, la confirmation de la fin du travail `job1` doit être reçue avant le lancement de `job2` et de `job3`.

```
schedule test1 on fr:
job1 confirmed
job2 follows job1
job3 follows job1
end
```

critical

Permet d'indiquer que le travail a une importance critique et doit donc être traité en conséquence.

Un travail critique bénéficie d'un traitement spécifique. Compte tenu de l'échéance et de la durée estimée, le planificateur :

- lors de la conception du plan, ou à chaque exécution de la commande **submit**, calcule l'heure de début au plus tard pour le lancement de chacun de ses prédécesseurs afin que le travail respecte l'échéance fixée. Il s'agit de l'*heure de début critique*. Un heure de début critique est attribuée au travail critique et à chacun de ces prédécesseurs.

  L'ensemble des prédécesseurs du travail critique est appelé *réseau critique* du travail.
- recalcule de manière dynamique, lors de l'exécution du plan, les heures de début critiques dans le réseau critique.

  Lorsqu'un prédécesseur risque de compromettre la réalisation opportune du travail critique, il est *promu*. En d'autres termes, grâce à différents mécanismes du système d'exploitation (implémentation de la commande **nice** sous UNIX ou modification du niveau de priorité sous Windows, par exemple), des ressources supplémentaires lui sont attribuées et sa soumission est prioritaire sur les autres travaux ne se trouvant pas dans le réseau critique. Cette action est exécutée de manière récurrente sur tous les prédécesseurs à l'intérieur du réseau critique et, le cas échéant, sur le travail critique tant qu'il n'existe aucun risque de retard de ce travail.

Important : Une échéance doit être spécifiée pour les travaux critiques au niveau du travail, du flot de travaux ou du cycle d'exécution, notamment dans le plan car ils sont associés à un cycle d'exécution. Alors que les travaux critiques soumis à la demande dans le plan peuvent ne pas avoir d'échéance, dans ce cas, l'option globale `deadlineOffset` est utilisée.

Syntaxe

critical

deadline

Indique le temps nécessaire pour l'achèvement d'un travail ou d'un flot de travaux. Les travaux et flots de travaux qui n'ont pas encore démarré ou qui sont encore en cours d'exécution lorsque la date d'échéance arrive à expiration sont considérés comme étant *en retard* dans le plan. Lorsqu'un travail (ou flot de travaux) est en retard, les actions suivantes sont exécutées :

- Le travail apparaît en retard dans **conman**.
- Le programme envoie un événement à Tivoli Enterprise Console et IBM Tivoli Business Systems Manager.
- Un message est envoyé à *stdlist* et aux journaux de la console.

Lorsqu'un travail ne se termine pas avant son délai, un message d'avertissement est affiché. Si ce travail ne fait pas partie d'un flot de travaux réacheminé et que vous exécutez `JnextPlan` alors qu'il est encore en cours d'exécution, le travail est inséré dans USERJOBS. Dans ce cas, un autre message d'avertissement sur le délai expiré est ajouté dans le fichier *rép_base_TWS*`/stdlist/logs/`*yyyymmdd*`_TWSMERGE.log`.

Remarque : Lorsque vous utilisez le mot-clé **deadline**, veillez à ce qu'une valeur supérieure à 0 soit attribuée à l'option **bm check deadline** dans le fichier de configuration `localopts` sur les postes de travail que vous utilisez. Définissez l'option **bm check deadline** sur chaque poste de travail sur lequel vous souhaitez

être notifié de l'expiration du délai, ou, si vous souhaitez obtenir des informations à jour sur l'ensemble de l'environnement, définissez l'option dans le gestionnaire de domaine maître. Les échéances des travaux critiques sont évaluées automatiquement, indépendamment de l'option **bm check deadline**. Pour plus d'informations sur l'option **bm check deadline**, voir le *Guide d'administration*.

Syntaxe

deadline *heure* [**timezone** | **tz** *fuseau_horaire*][**+***n* **day**[**s**] [**,...**]

Arguments

heure Indique une heure. Les valeurs admises sont comprises entre **0000** et **2359**.

tzname Indique le fuseau horaire à utiliser lors du calcul de l'heure d'échéance. Voir Chapitre 16, «Gestion des fuseaux horaires», à la page 653 pour les noms des fuseaux horaires. Il s'agit par défaut du fuseau horaire correspondant au poste de travail sur lequel le travail ou le flot de travaux est lancé.

n Indique un décalage en jours à partir de l'heure d'échéance planifiée.

Remarque : Si des heures **deadline** et **until** ou **at** sont indiquées, les fuseaux horaires doivent être identiques.

Exemples

L'exemple suivant lance le flot de travaux `sked7` chaque jour, ainsi que le travail `jobc`. Ils doivent démarrer à 14h30 et être terminés à 16h00.

```
schedule sked7 on everyday :
    jobc at 1430 deadline 1600
end
```

description

Inclut une description du flot de travaux.

Syntaxe

description *"texte"*

Commentaires

La longueur maximale de cette zone est 120 caractères.

Exemples

```
schedule test1
description "Revenue at the end of the month"
on monthend
:
job1
job2
job3
end
```

draft

Marque un flot de travaux comme épreuve. Un flot de travaux au stade d'épreuve n'est pas ajouté au plan de production.

Syntaxe

draft

Commentaires

Un flot de travaux à l'état de brouillon n'est pas pris en compte lors de la résolution des dépendances et n'est pas ajouté au plan de production. Après avoir supprimé le mot clé draft d'un flot de travaux, vous devez exécuter la commande **JnextPlan** pour ajouter le flot de travaux au plan de préproduction, puis au plan de production.

Exemples

```
schedule test1 on monthend
draft
:
job1
job2
job3
end
```

end

Signale la fin d'une définition de flot de travaux.

Syntaxe

end

Exemples

```
schedule test1 on monthend
:
job1
job2
job3
end << end of job stream >>
```

every

Définit la fréquence de répétition d'un travail. Le travail est lancé de façon répétitive selon la fréquence indiquée. Si une dépendance du travail n'est pas satisfaite, l'itération n'a lieu qu'une fois la dépendance satisfaite.

Syntaxe

every *fréquence*

Arguments

fréquence

Fréquence de répétition exprimée en heures et en minutes, au format *hhmm*. La fréquence peut dépasser 24 heures.

Commentaires

- L'itération **every** d'un travail ne s'arrête pas, même si l'une des répétitions du travail s'arrête de manière anormale.
- Si vous avez utilisé l'option **every** sans la dépendance **at**, les travaux réexécutés sont planifiés selon la fréquence **every** définie, à partir de l'heure de démarrage réel du travail.

- Dans le cas particulier où l'option **every** est utilisée avec la dépendance **at** et qu'une réexécution est retardée (pour une dépendance ou toute autre cause) alors, pendant que Tivoli Workload Scheduler se réaligne sur l'heure **at**, il est possible qu'une ou deux itérations ne respectent pas la fréquence **every**. Dans tous les autres cas, la fréquence EVERY est toujours respectée.

  L'exemple 2 explique comment Tivoli Workload Scheduler se réaligne sur l'heure **at** si le travail commence après l'heure **at** définie et si des itérations sont perdues.
- Si une instance **every** ne démarre pas à l'heure de début attendue, utilisez l'option **bm late every** pour définir le nombre maximal de minutes qui peuvent s'écouler avant que Tivoli Workload Scheduler n'ignore le travail. La valeur de l'option doit être définie dans le fichier `<TWSHOME>/localopts` :

  bm late every = xx
  Où *xx* est le nombre de minutes.

  Cette option est locale pour chaque agent, c'est pourquoi elle doit être définie sur chaque agent tolérant aux pannes qui a **every** travaux dont le jeu d'options est **bm late every**.

  L'option **bm late every** s'applique uniquement aux travaux pour lesquels sont définies à la fois l'option **every** et la dépendance de temps **at** et n'a aucun impact sur les travaux pour lesquels seule l'option **every** est définie. Seuls les travaux pour lesquels la fréquence **every** est supérieure à la valeur **bm late every** seront concernés.

  L'exemple 4, à la page 250 montre le comportement de Tivoli Workload Scheduler lorsque le retard d'une instance **every** ne dépasse pas la valeur de l'option **bm late every**.

  L'exemple 5, à la page 250 montre le comportement de Tivoli Workload Scheduler lorsque le retard d'une instance **every** dépasse la valeur de l'otpion **bm late every**.

  L'exemple 6, à la page 250 montre le comportement de Tivoli Workload Scheduler lorsque la première instance d'un travail ne s'exécute pas à l'heure de début attendue et dépasse la valeur de l'option **bm late every**.
- Si le mot-clé **every** est défini pour un travail lorsque l'heure d'été (DST) est désactivée, c'est-à-dire lorsque l'horloge est reculée d'une heure, la commande **every job** est avertie de ce changement et s'exécute également lors du deuxième intervalle de temps répété.

Exemples

1. L'exemple ci-dessous exécute le travail `testjob` toutes les heures :
   ```
   testjob every 100
   ```
2. Dans l'exemple ci-dessous, le travail `testjob1` doit s'exécuter toutes les 15 minutes entre 18 et 20 heures :
   ```
   testjob1 at 1800 every 15 until 2000
   ```

   Le travail est censé être exécuté à 18:00, 18:15, 18:30, et ainsi de suite toutes les 15 minutes.

   Si le travail est soumis `adhoc` à 18:33, il est exécuté à 18:33, 18:34, 18:45, etc. La raison est la suivante :

   Notez tout d'abord que dans un travail, deux valeurs de temps doivent être prises en compte :

   - *start_time*. Il s'agit de l'heure à laquelle il a été prévu d'exécuter le travail. Elle est définie sur l'heure **at** spécifiée pour le travail ou sur l'heure où la

réexécution doit être lancée. Cette valeur peut être affichée à l'aide de `conman showjobs` avant que l'itération du travail ne démarre.

- *time_started*. Il s'agit de l'heure à laquelle le travail a réellement démarré (18:33, par exemple). Cette valeur peut être affichée à l'aide de `conman showjobs` une fois que l'itération du travail a démarré.

Etant donné que `testjob1` a été soumis `adhoc` à 18:33, cette information s'affiche tout de suite après la soumission :

avec `conman showjobs`

```
TESTJOB1 HOLD 1800
```

dans le fichier Symphony

`start_time=1800` (parce que le travail est planifié pour s'exécuter à (**at**) 18:00)

`time_started=NULL` (parce que le travail n'a pas encore démarré)

Puisque l'heure de début, définie par `start_time` (1800), est plus petite que l'heure en cours (1833), `testjob1` démarre immédiatement et les informations mises à jour sont alors :

avec `conman showjobs`

```
TESTJOB1 SUCC 1833
```

dans le fichier Symphony

`start_time=1800` (parce que le travail était planifié pour s'exécuter à (**at**) 18:00)

`time_started=1833` (parce que le travail a démarré à 18:33)

Lorsque **batchman** calcule l'heure de la prochaine itération, il utilise les données suivantes :

`start_time=1800`

`rate=0015`

`current_time=1833`

Etant donné que l'heure de la prochaine itération (1800+0015=1815) serait encore antérieure à la valeur *current_time* (1833), **batchman** identifie la dernière itération planifiée qui n'a pas été exécutée en ajoutant à l'heure *start_time* la valeur de la fréquence *every_rate* autant de fois que possible sans dépasser l'heure *current_time*

1800 + 0015 + 0015 = **1830** < 1833

puis lance la commande pour exécuter cette itération. En supposant que cette itération est exécutée à 18:34, les informations sont les suivantes une fois que le travail a commencé :

avec `conman showjobs`

```
TESTJOB1 SUCC 1834
```

dans le fichier Symphony

`start_time=1830` (parce que cette itération du travail était planifiée pour s'exécuter à (**at**) 18:30)

`time_started=1834` (parce que cette itération du travail a démarré à 1834)

Une fois cette itération du travail terminée, **batchman** calcule à nouveau l'heure à laquelle l'itération suivante doit démarrer en utilisant les valeurs mises à jour suivantes :

```
start_time=1830
rate=0015
current_time=1834
```

Le fait que l'heure d'itération suivante (1830+0015=1845) est ultérieure à la valeur *current_time* (1834) indique à **batchman** que l'itération a repris. L'heure de l'itération, à partir de 18:45, peut à présent être réalignée sur les heures d'itération prévues dans la définition du travail par les mots clés **at** et **every**.

3. L'exemple suivant ne démarre pas l'itération du travail `testjob2` tant que le travail `testjob1` n'a pas abouti.

   ```
   testjob2 every 15 follows testjob1
   ```

4. Dans l'exemple suivant, le retard d'une instance de travail **every** ne dépasse pas la valeur de l'option **bm late every** :

   ```
   bm late every = 10
   JOB AT 1400 EVERY 0030
   ```

   Ce travail est censé être exécuté à 14h00, à 14h30, à 15h00, et ainsi de suite toutes les trente minutes.

   Si le serveur est en panne de 14h35 à 16h05, les instances de 15h00, 15h30 et 16h00 ne s'exécutent pas. A 16h05, Tivoli Workload Scheduler redémarre. Lorsqu'il analyse le fichier Symphony, il détermine que l'heure potentielle la plus propice pour la prochaine instance de travail **every** est 16h00. Tivoli Workload Scheduler vérifie si l'heure potentielle la plus propice (16h00) ne dépasse pas le retard maximal autorisé pour un travail**every** (10 minutes).

   Dans ce cas, le retard n'a pas dépassé l'option **bm late every**, c'est pourquoi Tivoli Workload Scheduler adopte un comportement habituel et crée une instance de travail **every** dont l'heure de début est 16h00. Les instances suivantes sont à 16h30, 17h00 et ainsi de suite, toutes les trente minutes.

5. Dans l'exemple suivant, le retard d'une instance d'un travail **every** dépasse la valeur de l'option **bm late every** :

   ```
   bm late every = 10
   JOB AT 1400 EVERY 00030
   ```

   Ce travail est censé être exécuté à 14h00, 14h30, 15h00 et ainsi de suite, toutes les trente minutes.

   Si le serveur est en panne de 14h35 à 16h20, les instances de 15h00, 15h30 et 16h00 ne s'exécutent pas. A 16h20, Tivoli Workload Scheduler redémarre. Lorsqu'il analyse le fichier Symphony, il détermine que l'heure potentielle la plus propice pour la prochaine instance de travail **every** est 16h00. Tivoli Workload Scheduler vérifie si l'heure potentielle la plus propice (16h00) ne dépasse pas le retard maximal autorisé pour une instance de travail **every** (10 minutes).

   Dans ce cas, le retard dépasse l'option **bm late every**, c'est pourquoi Tivoli Workload Scheduler adopte un nouveau comportement, il ne lance pas l'instance de travail **every** à 16h00 et il crée une instance de travail **every** dont l'heure de début est 16h30.

6. L'exemple suivant montre le comportement de Tivoli Workload Scheduler lorsque la première instance d'un travail ne s'exécute pas à l'heure de début attendue et dépasse la valeur de l'option **bm late every** :

   ```
   bm late every = 10
   JOB AT 1400 EVERY 00030
   ```

Ce travail est censé être exécuté à 14h00, 14h30, 15h00 et ainsi de suite, toutes les trente minutes.

Si le serveur est en panne de 10h00 à 14h15, la première instance de travail ne s'exécute pas. A 14h15, Tivoli Workload Scheduler redémarre. Lorsqu'il analyse le fichier Symphony, il détermine que la première instance de ce travail **every** ne s'est pas exécutée. Dans ce cas, Tivoli Workload Scheduler lance le travail à 14h15.

except

Définit les dates qui constituent des exceptions par rapport aux dates définies par le mot clé **on** d'un flot de travaux. Pour plus d'informations, voir «on», à la page 267.

Syntaxe

except [**runcycle** *nom*]
[**validfrom** *date*] [**validto** *date*]
[**description** *"texte"*]
{*date* | *jour* | *agenda* | *demande* | *"icalendar"*}
[**,...**]
[**fdignore** | **fdnext** | **fdprev**][**subset** *nom_sous-ensemble* **AND** | **OR**]

Arguments

runcycle *nom*
Indique un libellé avec un nom usuel pour le cycle d'exécution spécifié dans les lignes suivantes.

valide de *date* **... valide jusqu'à** *date*
Délimite la période au cours de laquelle le flot de travaux est actif, c'est-à-dire pendant laquelle il est ajouté au plan de production. Notez que la date spécifiée comme valeur **valide jusqu'à** n'est pas incluse dans le cycle d'exécution et le flot de travaux n'est donc pas actif à cette date.

description "*texte***"**
Contient une description du cycle d'exécution.

date Indique un cycle d'exécution qui s'exécute à des dates spécifiques. La syntaxe utilisée pour ce type de cycle d'exécution est la suivante :

jjmmaaaa [,aaaammjj][,...]Par exemple, pour un flot de travaux dont l'exécution est prévue les 25 mai 2009 et 12 juin 2009, la valeur est la suivante :

```
on
20090525,20090612
```

jour Indique un cycle d'exécution qui s'exécute des jours spécifiques. La syntaxe utilisée pour ce type de cycle d'exécution est la suivante :

{mo | tu | we | th | fr | sa | su}Par exemple, pour un flot de travaux qui est planifié pour s'exécuter chaque lundi, la valeur est la suivante :

```
on
mo
```

agenda Dates indiquées dans un agenda portant ce nom. Le nom de l'agenda peut être suivi d'un décalage au format suivant :

{**+** | **-**}*n* {**day**[**s**] | **weekday**[**s**] | **workday**[**s**]}

Où :

n Indique le nombre de jours, jours de la semaine ou jours ouvrés.

jours Chaque jour de la semaine.

weekdays
Tous les jours de la semaine sauf le samedi et le dimanche.

workdays
Tous les jours de la semaine excepté les samedis et dimanches (sauf indication contraire par le mot clé **freedays**) et les dates figurant dans un agenda désigné des jours chômés ou dans l'agenda **holidays**.

demande
Sélectionne le flot de travaux uniquement lorsque celui-ci est demandé. Ce paramètre est utilisé pour les flots de travaux qui sont sélectionnés par nom et non par date. Pour empêcher un flot de travaux planifié d'être sélectionné pour le travail **JnextPlan**, redéfinissez-le ON REQUEST.

Remarque : Si vous tentez d'exécuter un flot de travaux contenant plusieurs heures "à la demande" (on request), tenez compte des points suivants :

- Les indicateurs "On request" ont la priorité sur les indicateurs "at".
- Les indicateurs "On request" n'ont jamais la priorité sur les indicateurs "On".

icalendar
Représente une norme utilisée pour spécifier une règle récurrente qui décrit le moment où un flot de travaux s'exécute.

La syntaxe utilisée pour un cycle d'exécution de type *icalendar* est la suivante :

FREQ={DAYLY | WEEKLY | MONTHLY | YEARLY}

[;INTERVAL=[-]*n*]

[;{BYFREEDAY | BYWORKDAY | BYDAY=*liste_jours_semaine* |

BYMONTHDAY=*liste_jours_mois***}]**

où la valeur par défaut du mot clé **INTERVAL** est 1.

A l'aide du mot clé *icalendar*, vous pouvez spécifier qu'un flot de travaux s'exécute :

tous les *n* **jours**
en utilisant le format suivant :

FREQ=DAILY[;INTERVAL=*n*]

où la valeur définie pour **valide de** est le premier jour des dates résultantes.

Par exemple, pour un flot de travaux planifié pour s'exécuter chaque jour, la valeur est la suivante :

```
FREQ=DAILY
```

Pour un flot de travaux planifié pour s'exécuter un jour sur deux, la valeur est la suivante :

```
FREQ=DAILY;INTERVAL=2
```

chaque jour chômé ou ouvré
en utilisant le format suivant :

FREQ=DAILY[;INTERVAL=*n*]

;BYFREEDAY|BYWORKDAY

Par exemple, pour un flot de travaux planifié pour s'exécuter chaque jour chômé, la valeur est la suivante :

```
FREQ=DAILY;BYFREEDAY
```

Pour un flot de travaux planifié pour s'exécuter un jour ouvré sur deux, la valeur est la suivante :

```
FREQ=DAILY;INTERVAL=2;BYWORKDAY
```

toutes les *n* **semaines, à des** *jours de la semaine* **spécifiques**
en utilisant le format suivant :

FREQ=WEEKLY[;INTERVAL=*n*]

;BYDAY=*liste_jours_semaine*

où la valeur définie pour *liste_jours_semaine* est

```
[SU][,MO][,TU][,WE][,TH][,FR][,SA]
```

Par exemple, pour un flot de travaux planifié pour s'exécuter chaque semaine le vendredi et le samedi, la valeur est la suivante :

```
FREQ=WEEKLY;BYDAY=FR,SA
```

Pour un flot de travaux planifié pour s'exécuter toutes les trois semaines, le vendredi, la valeur est la suivante :

```
FREQ=WEEKLY;INTERVAL=3;BYDAY=FR
```

tous les *n* **mois, à des dates spécifiques**
en utilisant le format suivant :

FREQ=MONTHLY[;INTERVAL=*n*]

;BYMONTHDAY=*liste_jours_mois*

où la valeur définie pour *liste_jours_mois* est représentée par une liste de

```
[+nombre_de_jours_à_partir_début_du_mois]
[-nombre_de_jours_à_partir_fin_du_mois]
[nombre_de_jours_du_mois]
```

Par exemple, pour un flot de travaux planifié pour s'exécuter le 27ème jour de chaque mois, la valeur est la suivante :

```
FREQ=MONTHLY;BYMONTHDAY=27
```

Pour un flot de travaux planifié pour s'exécuter tous les six mois, le 15 du mois et le dernier jour du mois, la valeur est la suivante :

```
FREQ=MONTHLY;INTERVAL=6;BYMONTHDAY=15,-1
```

tous les *n* **mois, à des jours spécifiques de la semaine**
en utilisant le format suivant :

FREQ=MONTHLY[;INTERVAL=*n*]

;BYDAY=*numéro_de_la_semaine_du_mois_jour_semaine*

où la valeur définie pour *numéro_de_la_semaine_du_mois_jour_semaine* est représentée par une liste de

[*+numéro_semaine_à_partir_début_du_mois*]
[*-numéro_semaine_à_partir_fin_du_mois*]
[*jour_semaine*]

Par exemple, pour un flot de travaux planifié pour s'exécuter chaque mois, le premier lundi et le dernier vendredi du mois, la valeur est la suivante :

```
FREQ=MONTHLY;BYDAY=1MO,-1FR
```

Pour un flot de travaux planifié pour s'exécuter tous les six mois, le deuxième mardi du mois, la valeur est la suivante :

```
FREQ=MONTHLY;INTERVAL=6;BYDAY=2TU
```

toutes les *n* **années**
en utilisant le format suivant :

FREQ=YEARLY[;INTERVAL=*n*]

où la valeur définie pour **valide de** est le premier jour des dates résultantes.

Par exemple, pour un flot de travaux planifié pour s'exécuter chaque année, la valeur est la suivante :

```
FREQ=YEARLY
```

Pour un flot de travaux planifié pour s'exécuter tous les deux ans, la valeur est la suivante :

```
FREQ=YEARLY;INTERVAL=2
```

fdignore | fdnext | fdprev
Indique une règle qui doit être appliquée lorsque la date devant être exclue correspond à un jour chômé. Il peut s'agir de l'une des règles suivantes :

fdignore
La date n'est pas exclue.

fdnext Le jour ouvré le plus proche suivant le jour chômé est exclu.

fdprev
Le jour ouvré le plus proche précédant le jour chômé est exclu.

subset *nom_sous-ensemble*
Indique le nom du sous-ensemble. Si vous ne spécifiez pas de nom, SUBSET_1 est utilisé par défaut.

AND | OR
Par défaut, les cycles d'exécution dans un sous-ensemble sont dans une relation logique **OR** mais vous pouvez la changer en relation logique **AND** tant que le résultat du groupe de cycle d'exécution est une date ou un ensemble de dates positif (inclusif).

Pour plus d'informations sur les autres mots clés contenus dans la syntaxe **except**, voir «on», à la page 267.

Commentaires

Vous pouvez définir plusieurs instances du mot clé **except** pour le même flot de travaux. Chaque instance correspond à un cycle d'exécution auquel vous pouvez associer une règle de jour chômé.

Si une définition de flot de travaux contient plusieurs instances **except**, celles-ci doivent être consécutives.

Chaque instance du mot clé doit contenir l'une des valeurs autorisées par la syntaxe **except**.

Exemples

Dans l'exemple suivant, le flot de travaux `testskd2` est sélectionné pour être exécuté tous les jours de la semaine, à l'exception des dates figurant dans les agendas intitulés `monthend` et `holidays` :

```
schedule testskd2 on weekdays
except monthend,holidays
```

Dans l'exemple suivant, le flot de travaux `testskd3` est sélectionné pour être exécuté tous les jours de la semaine à l'exception du 15 mai 2005 et du 23 mai 2005 :

```
schedule testskd3 on weekdays
except 05/15/2005,05/23/2005
```

Dans l'exemple suivant, le flot de travaux `testskd4` est sélectionné pour être exécuté tous les jours sauf pendant les deux jours précédant toute date figurant dans un agenda intitulé `monthend` :

```
schedule testskd4 on everyday
except monthend-2 weekdays
```

Sélectionnez le flot de travaux `sked4` de sorte qu'il s'exécute les lundis, mardis et 2 jours de la semaine avant chaque date répertoriée dans l'agenda `monthend`. Si la date d'exécution correspond à un jour chômé, exécutez le flot de travaux le jour ouvré suivant le plus proche. N'exécutez pas le flot de travaux le mercredi.

```
schedule sked4
on mo
on tu, MONTHEND -2 weekdays fdnext
except we
```

Sélectionnez le flot de travaux `testskd2` pour qu'il s'exécute chaque jour de la semaine sauf les jours répertoriés dans `monthend`. Si une date figurant dans `monthend` correspond à un jour chômé, excluez le jour de la semaine le plus proche précédant cette date. Dans cet exemple, les jours chômés sont les samedis, les dimanches et toutes les dates répertoriées dans l'agenda `holidays` par défaut.

```
schedule testskd2
on weekdays
except MONTHEND fdprev
```

follows

Définit les autres travaux et flots de travaux qui doivent aboutir avant le lancement d'un travail ou d'un flot de travaux.

Commentaires

Utilisez la syntaxe suivante pour les flots de travaux :

[**follows** {[*agent_réseau*::][*poste_de_travail*#]*nom_flot_travaux*[**.***nom_travail* | **@**]

[**previous** | **sameday** | **relative from [+/-]** *heure* **to [+/-]** *heure* | **from** *heure* [+/-*n* day[s]] **to** *heure* [+/-*n* day[s]]

Utilisez la syntaxe suivante pour les travaux :

[**follows** {[*agent_réseau*::][*poste_de_travail*#]*nom_flot_travaux*{.*nom_travail* | **@**}

[**previous** | **sameday** | **relative from [+/-]** *heure* **to [+/-]** *heure* | **from** *heure* [+/-*n* day[s]] **to** *heure* [+/-*n* day[s]]

Arguments

agent_réseau
Nom de l'agent réseau dans lequel la dépendance interréseau est définie.

poste de travail
Le poste de travail sur lequel s'exécute le travail ou le flot de travaux qui doit être terminé. Par défaut, il s'agit du même poste de travail que le travail ou le flot de travaux dépendant.

Si aucun *poste de travail* n'est spécifié avec l'*agent réseau*, le poste de travail auquel l'agent réseau est connecté est utilisé par défaut.

nom_flot_travaux
Le nom du flot de travaux qui doit être terminé. Pour un travail, il s'agit par défaut du même flot de travaux que le travail dépendant.

heure Indique une heure. Les valeurs admises sont comprises entre **0000** et **2359**.

nom_travail
Le nom du travail qui doit être terminé. Un signe `at` (@) peut être utilisé pour indiquer que toutes les tâches du flot de travaux doivent se terminer correctement.

Commentaires

Les critères de résolution des dépendances permettent de définir la manière dont le flot de travaux ou un travail référencé par une dépendance de prédécesseur/successeur externe correspond à un flot de travaux particulier ou à une instance de travail dans le plan. Parce que le plan autorise l'inclusion de plusieurs instances du même travail ou flot de travaux, vous pouvez identifier l'instance qui résout la dépendance de prédécesseur/successeur externe en fonction du critère de résolution suivant :

Valeur précédente la plus proche
L'instance de travail ou de flot de travaux qui résout la dépendance est la plus proche précédant l'instance qui inclut la dépendance.

Même jour
L'instance de travail ou de flot de travaux qui résout la dépendance est la plus proche dans le temps planifiée pour commencer le jour où l'instance qui inclut la dépendance est planifiée pour exécution.

Dans un intervalle relatif
L'instance de travail ou de flot de travaux qui résout la dépendance est la plus proche, dans un intervalle de temps de votre choix, définie relativement à l'heure de début planifié de l'instance dépendante.

Dans un intervalle absolu
L'instance de travail ou de flot de travaux qui résout la dépendance est la plus proche dans un intervalle de temps de votre choix. L'intervalle de temps n'est pas lié à l'heure de début planifié de l'instance dépendante.

Quels que soient les critères de correspondance utilisés, si plusieurs instances d'un flot de travaux predecessor potentielle existent dans l'intervalle de temps spécifié, la règle utilisée par le produit pour identifier l'instance predecessor correcte est la suivante :

1. Tivoli Workload Scheduler recherche l'instance la plus proche précédant l'heure de début du travail ou du flot de travaux dépendant. Si une telle instance existe, il s'agit de l'instance prédécesseur.
2. S'il n'y a pas d'instance précédente, Tivoli Workload Scheduler considère l'instance prédécesseur correcte comme étant l'instance la plus proche qui commence après l'heure de début du travail ou du flot de travaux dépendant.

L'agenda classe les dépendances de prédécesseur/successeur comme *internes* lorsqu'elles ne sont spécifiées que par leur nom de travail dans le flot de travaux. Il les classe en temps que dépendances *externes* lorsqu'elles sont spécifiées dans le format *jobStreamName.workstationName.jobName*.

Lorsqu'un flot de travaux inclut un travail possédant une dépendance de prédécesseur/successeur qui partage le même nom de flot de travaux (par exemple, le flot de travaux `schedA` inclut un travail appelé `job6` qui possède une dépendance de prédécesseur/successeur sur `schedA.job2`), la dépendance est ajoutée au plan en tant que dépendance de prédécesseur/successeur *externe*. Depuis la version 8.3, contrairement aux versions précédentes, étant donné que l'agenda utilise le critère de correspondance `sameday` pour résoudre des dépendances externes, les dépendances générées de cette manière ne sont jamais ajoutées lors de la première soumission de l'objet.

Pour davantage d'informations et d'exemples sur la façon dont les dépendances de prédécesseur/successeur externes sont résolues dans le plan, voir «Gestion des dépendances externes de prédécesseur/successeur des travaux et des flots de travaux», à la page 65.

Exemples

L'exemple suivant indique que le flot de travaux `skedc` ne doit pas être lancé tant que l'instance de flot de travaux `sked4` précédente la plus proche sur le poste de travail `site1` n'a pas abouti :

```
schedule skedc on fr follows site1#sked4 previous
```

L'exemple suivant indique que le flot de travaux `skedc` ne doit pas être lancé tant que l'instance de flot de travaux `sked4` sur le poste de travail `site1` qui s'exécute entre 12:00, 3 jours avant, jusqu'à 3:00 un jour après n'a pas abouti :

```
schedule skedc on fr follows site1#sked4 from 1200 –3 days to 0300 1 day
```

L'exemple suivant indique que le flot de travaux `skedc` ne doit pas être lancé tant que le flot de travaux `sked4` sur le poste de travail `site1` et le travail `joba` du flot de travaux `sked5` sur le poste de travail `site2` n'ont pas abouti :

```
schedule skedc on fr
follows site1#sked4,site2#sked5.joba
```

Ne lancez pas `sked6` tant que le travail `jobx` du flot de travaux `skedx` sur l'agent réseau `cluster4` n'a pas abouti :

```
sked6 follows cluster4::site4#skedx.jobx
```

L'exemple suivant indique que le travail jobd ne doit pas être lancé tant que le travail joba du même flot de travaux et le travail job3 du flot de travaux skeda n'ont pas abouti :

```
jobd follows joba,skeda.job3
```

freedays

Utilisez **freedays** pour indiquer le nom d'un agenda des jours chômés dans lequel figurent les jours non travaillés dans votre entreprise. Les conditions d'exécution d'un flot de travaux pendant ces jours particuliers sont définies dans une règle *freedays* lors de la configuration du cycle d'exécution. Tivoli Workload Scheduler utilise ce type d'agenda comme agenda de base pour calculer les *jours ouvrés* pour le flot de travaux.

Le mot clé affecte uniquement la planification des flots de travaux pour lesquels il est indiqué.

Syntaxe

freedays *Nom_agenda* **[-sa] [-su]**

Arguments

Nom_agenda
: Nom de l'agenda devant être utilisé comme agenda des jours chômés pour le flot de travaux. Si *Nom_agenda* ne se trouve pas dans la base de données, Tivoli Workload Scheduler génère un message d'erreur lorsque vous sauvegardez le flot de travaux. Si *Nom_agenda* ne se trouve pas dans la base de données lors de l'exécution de la commande **schedulr**, Tivoli Workload Scheduler génère un message d'erreur et utilise l'agenda par défaut **holidays** à la place. N'utilisez pas les noms des jours de la semaine pour nommer un agenda.

-sa
: Le samedi est considéré comme un *jour ouvré*.

-su
: Le dimanche est considéré comme un *jour ouvré*.

Commentaires

Si vous spécifiez un agenda des jours chômés dans la définition du flot de travaux, le concept *workdays* prend la valeur suivante : *workdays = tous les jours sauf le samedi et le dimanche (à moins que vous n'ayez indiqué -sa ou -su avec freedays) et toutes les dates de Nom_agenda*

Si vous ne précisez pas **freedays** dans la définition du flot de travaux, alors : *workdays = tous les jours sauf le samedi, le dimanche et les congés*

Par défaut, le *samedi* et le *dimanche* sont considérés comme des jours chômés, sauf si vous indiquez le contraire en ajoutant **-sa**, **-su** ou les deux après *Nom_agenda*.

Exemples

Sélectionnez le flot de travaux sked2 pour qu'il s'exécute le 01/01/2005 ainsi que tous les jours ouvrés ne figurant pas sur l'agenda des jours chômés nommé GERMHOL.

```
schedule sked2
freedays GERMHOL
on 01/01/2005, workdays
```

Sélectionnez le flot de travaux sked3 pour qu'il s'exécute deux jours ouvrés avant chaque date de l'agenda PAYCAL. Les jours ouvrés incluent tous les jours de la semaine, du lundi au samedi, non répertoriés dans l'agenda des jours chômés nommé USAHOL.

```
schedule sked3
freedays USAHOL -sa
on PAYCAL -2 workdays
```

Sélectionnez le flot de travaux sked3 pour les dates répertoriées dans l'agenda APDATES. Si une date sélectionnée correspond à un jour chômé, n'exécutez pas le flot de travaux. Dans le présent exemple, les dimanches et toutes les dates répertoriées dans l'agenda GERMHOL sont considérés comme jours chômés. A l'exception des dates spécifiques répertoriées dans GERMHOL, tous les jours de la semaine du lundi au samedi sont des jours ouvrés.

```
schedule sked3
freedays GERMHOL -sa
on APDATES fdignore
```

Sélectionnez le flot de travaux testsked3 pour qu'il s'exécute chaque jour de la semaine sauf le 15/5/2005 et le 23/5/2006. Si le 23/05/2006 correspond à un jour chômé, ne l'excluez pas. Dans le présent exemple, les samedis, dimanches et toutes les dates répertoriées dans GERMHOL doivent être considérés comme des jours chômés. A l'exception des dates spécifiques répertoriées dans GERMHOL, tous les jours de la semaine du lundi au vendredi sont des jours ouvrés.

```
schedule testskd3
freedays GERMHOL
on weekdays
except 5/15/2005 fdignore
except 5/23/2006
```

Sélectionnez le flot de travaux testsked4 pour qu'il s'exécute tous les jours sauf 2 jours de la semaine avant chaque date répertoriée dans l'agenda MONTHEND. Si la date à exclure correspond à un jour chômé, ne l'excluez pas, mais excluez le jour ouvré suivant le plus proche. Dans l'exemple suivant, les jours chômés correspondent à toutes les dates répertoriées dans USAHOL, tandis que les jours ouvrés sont tous les jours de la semaine du lundi au dimanche ne figurant pas dans USAHOL.

```
schedule testskd4
freedays USAHOL -sa -su
on everyday
except MONTHEND -2 weekdays fdnext
```

job statement

Les travaux peuvent être définis dans la base de données indépendamment (voir «Travail», à la page 772) ou dans le cadre des flots de travaux. Dans l'un ou l'autre cas, les modifications sont apportées dans la base de données et n'ont pas d'incidence sur le plan de production jusqu'au début d'un nouveau plan de production.

Syntaxe

Pour définir un travail comme faisant partie d'un flot de travaux, utilisez la syntaxe suivante dans la définition du flot de travaux :

[*poste_de_travail***#**]*nom_travail* [**as** *nomveau_nom*]
{**scriptname** *file_name* | **docommand** *"commande"*}
streamlogon *nom_utilisateur*

[**description** *"description"*]
[**tasktype** *type_tâche*]
[**interactive**]
[**rccondsucc** *"Condition de succès"*]
[**recovery**
{**stop** | **continue** | **rerun**}
[**after** [*poste_de_travail*#]*nom_travail*]
[**abendprompt** *"texte"*]]

Pour utiliser un travail déjà défini dans la base de données dans la définition du flot de travaux, définissez *instruction_travail* en utilisant la syntaxe suivante :

[*poste_de_travail*#]*nom_travail* [**as** *nomveau_nom*]

Arguments

as Nom à utiliser pour faire référence à l'instance de travail dans ce flot de travaux.

Pour les autres mots clés, voir «Travail», à la page 772.

Commentaires

Lors de la définition d'un travail comme faisant partie d'un flot de travaux, l'ajout de la définition du flot de travaux à la base de données entraîne celui de la définition du nouveau travail, laquelle peut désormais être référencée depuis d'autres flots de travaux.

Remarque : Les mots clés saisis incorrectement dans les définitions de travail entraînent le stockage de définitions de travaux tronquées dans la base de données. En fait, le mot clé erroné est considéré comme externe à la définition de travail et il est interprété comme le nom de travail d'une définition de travail supplémentaire. Généralement, cette mauvaise interprétation entraîne également une erreur de syntaxe ou une erreur de définition de travail inexistant pour la définition de travail supplémentaire.

Lors de l'ajout ou de la modification d'un flot de travaux, les attributs ou les options de reprise de ses travaux sont également ajoutés ou modifiés. N'oubliez pas que lors de l'ajout ou du remplacement d'un flot de travaux, les éventuelles modifications apportées aux travaux ont une incidence sur tous les autres flots de travaux qui les utilisent. Le rapport de références croisées, *xref*, peut être utilisé pour déterminer les noms des flots de travaux qui contiennent un travail particulier. Pour plus d'informations sur le rapport de références croisées, voir «xref», à la page 617.

Remarque : Les travaux planifiés pour être exécutés sur des postes de travail marqués comme étant *ignorés*, et appartenant à des flots de travaux planifiés pour s'exécuter sur des postes de travail actifs, sont ajoutés au plan bien qu'ils ne soient pas traités.

Exemples

L'exemple suivant définit un flot de travaux comportant trois travaux précédemment définis :

```
schedule bkup on fr at 20:00:
  cpu1#jbk1
  cpu2#jbk2
```

```
    needs 1 tape
  cpu3#jbk3
    follows jbk1
end
```

La définition de flot de travaux suivante contient des instructions de travail qui ajoutent ou modifient les définitions de deux travaux de la base de données :

```
schedule sked4 on mo:
  job1 scriptname "d:\apps\maestro\scripts\jcljob1"
    streamlogon jack
    recovery stop abendprompt "continue production"
  site1#job2 scriptname "d:\apps\maestro\scripts\jcljob2"
    streamlogon jack
    follows job1
end
```

keyjob

Le mot clé **keyjob** est utilisé pour indiquer qu'un travail est une clé dans la base de données, ainsi que dans le plan, et qu'il doit être surveillé par des applications telles que Tivoli Business Systems Manager ou Tivoli Enterprise Console. Pour plus d'informations sur l'activation du mécanisme des indicateurs clés, voir *IBM Tivoli Workload Scheduler - Intégration à d'autres produits*.

Syntaxe

keyjob

Exemples

Voici un exemple :

```
SCHEDULE cpu1#sched1
ON everyday
KEYSCHED
AT 0100
cpu1#myjob1 KEYJOB
END
```

keysched

Le mot clé **keysched** est utilisé pour indiquer qu'un travail est une clé dans la base de données, ainsi que dans le plan, et qu'il doit être surveillé par des applications telles que Tivoli Business Systems Manager. Pour plus d'informations sur l'activation du mécanisme des indicateurs clés, voir *IBM Tivoli Workload Scheduler - Intégration à d'autres produits*.

Syntaxe

keysched

Exemples

Voici un exemple :

```
SCHEDULE cpu1#sched1
ON everyday
KEYSCHED
AT 0100
cpu1#myjob1 KEYJOB
END
```

limit

Le mot clé **limit** limite le nombre maximal de travaux pouvant être exécutés simultanément dans un flot de travaux sur le même poste de travail.

Syntaxe

limit *nombre_max_travaux*

Arguments

nombre_max_travaux

Indique le nombre maximal de travaux qui peuvent être exécutés en même temps dans le flot de travaux. Les valeurs admises sont comprises entre **0** et **1024**. Si vous indiquez **0**, vous empêchez le lancement de tous les travaux, même de ceux pour lesquels la priorité **GO** ou **HI** est définie.

Exemples

Dans l'exemple suivant, le nombre de travaux pouvant être exécutés simultanément dans le flot de travaux `sked2` est limité à cinq :

```
schedule sked2 on fr
  limit 5:
```

matching

Définit une valeur par défaut pour les critères de correspondance à utiliser dans toutes les dépendances de prédécesseur/successeur lorsqu'aucun critère de correspondance n'est défini dans la définition du flot de travaux ou dans les travaux contenus dans le flot de travaux.

Syntaxe

matching {**previous** |**sameday** | **relative from [+/-]** *heure* **to [+/-]** *heure*

Arguments

Pour plus d'informations sur le mot clé utilisé avec **matching** voir le mot clé «follows», à la page 255.

Exemples

L'exemple suivant montre la définition du flot de travaux `SCHED2` qui :

- contient un travail `job1` pouvant s'exécuter aujourd'hui uniquement s'il a été exécuté la veille ;
- nécessite que l'instance du flot de travaux `SCHED1` qui s'exécute le même jour ait abouti avant de pouvoir s'exécuter.

```
SCHEDULE PDIVITA1#SCHED2
ON RUNCYCLE RULE1 "FREQ=DAILY;"
ON RUNCYCLE CALENDAR2 CAL1
MATCHING PREVIOUS
FOLLOWS PDIVITA1#SCHED1.@ SAMEDAY
FOLLOWS PDIVITA1#SCHED2.JOB1
:
PDIVITA1#JOB1

PDIVITA1#JOB2
END
```

Dans cet exemple, la dépendance de prédécesseur/successeur externe de PDIVITA1#SCHED2.JOB1 hérite des critères de correspondance spécifiés dans le mot clé **matching**.

Commentaires

Remarque : si vous supprimez un flot de travaux, puis l'ajoutez à nouveau à la base de données, le flot de travaux obtient un autre identificateur. Par conséquent, si le flot de travaux contient des dépendances FOLLOWS avec des critères PREVIOUS, ces dépendances ne sont pas mises en correspondance lorsque **JnextPlan** s'exécute à nouveau, car il s'agit de dépendances entre les plans qui font référence à un ancien identificateur.

Dans l'exemple suivant, si vous supprimez le flot de travaux JS01 afin de garantir l'intégrité référentielle de la base de données, FOLLOWS TWS851MASTER#JS01.@ est également supprimé de la définition de JS02 et du plan de préproduction.

Si vous supprimez le flot de travaux JS03 afin de garantir l'intégrité référentielle de la base de données, FOLLOWS TWS851MASTER#JS03.@ est également supprimé de la définition de JS02 et du plan de préproduction.

Si vous supprimez le flot de travaux JS02, puis l'ajoutez à nouveau au plan, ses dépendances FOLLOWS sont également ajoutées à nouveau. Lorsque le plan est étendu, la dépendance FOLLOWS TWS851MASTER#JS03.@ PREVIOUS du flot de travaux JS02 ne correspond pas à l'instance du flot de travaux JS03 provenant du plan précédent et cette dépendance n'est pas ajoutée.

A l'extension de plan suivante, le processus fonctionne à nouveau.

```
SCHEDULE TWS851MASTER#JS01
ON RUNCYCLE RULE1 "FREQ=DAILY;"
:
TWS851MASTER#J02
END
SCHEDULE TWS851MASTER#JS03
ON RUNCYCLE RULE1 "FREQ=DAILY;"
SCHEDTIME 1000
CARRYFORWARD
:
TWS851MASTER#J03
END
SCHEDULE TWS851MASTER#JS02
ON RUNCYCLE RULE1 "FREQ=DAILY;"
:
TWS851MASTER#J01
FOLLOWS TWS851MASTER#JS01.@
FOLLOWS TWS851MASTER#JS03.@ PREVIOUS
END
```

Pour éviter ce problème, utilisez la commande Composer **replace**. Dans ce cas, les identificateurs du flot de travaux ne sont pas modifiés.

maxdur

Indique la durée maximale d'exécution d'un travail. Vous pouvez exprimer cette durée en minutes, ou sous la forme d'un pourcentage de la durée estimée du travail la plus récente. Si un travail est en cours d'exécution et que la durée maximale a été dépassée, les actions suivantes sont réalisées :

- L'une des actions suivantes est déclenchée : Kill ou Continue.
- La durée du travail apparaît comme étant dépassée dans les cas suivants :

- Lors de l'exécution de **showjob** à partir de la ligne de commande **conman**, MaxDurationExceeded s'affiche.
- A partir de Dynamic Workload Console dans les propriétés du travail.
- Un message d'information est consigné dans le fichier *rép_base_TWS*/stdlist/logs/*aaaammjj*_TWSMERGE.log.

Si ce travail est toujours en cours d'exécution lorsque JnextPlan s'exécute, il est inséré dans le flot de travaux USERJOBS. Le paramètre de durée maximale n'est pas conservé pour le travail dans le flot de travaux USERJOBS et ne sera pas surveillé. Pour que le flot de travaux soit reporté et pour éviter que le travail ne soit déplacé vers le flot de travaux USERJOBS, marquez le flot de travaux initial où le paramètre de durée maximale a été spécifié sous la forme d'un flot de travaux carryforward (en paramétrant le mot-clé carryforward dans le flot de travaux) si l'option globale enCarryForward est définie sur yes ; sinon, définissez l'option globale enCarryForward sur all.

Syntaxe

maxdur *heure* | *pourcentage* % **onmaxdur** *action*

Arguments

heure Indique une durée exprimée à l'aide de la syntaxe *HHHMM*, où :

HHH Représente le nombre d'heures sous la forme d'un nombre compris entre 000 et 500.

MM Représente le nombre de minutes sous la forme d'un nombre compris entre 00 et 59.

pourcentage
Indique le pourcentage de la durée estimée la plus récente. Il peut s'agir d'un nombre compris entre 0 et 1000000.

onmaxdur *action*
Indique l'action à déclencher sur un travail toujours en cours d'exécution lorsque la durée maximale spécifiée pour ce travail a été dépassée. Les valeurs possibles du paramètre de l'*action* sont :

Kill Indique d'arrêter le travail en cours d'exécution. A la suite de la commande kill, les travaux sont à l'état ABEND (arrêt anormal). Les travaux ou les flots de travaux qui dépendent de ces travaux ne sont pas libérés. Ces travaux ne peuvent pas être exécutés à nouveau.

Continue
Indique de poursuivre l'exécution du travail en cours, même s'il a dépassé la durée maximale.

Lors de la soumission d'une commande **conman** pour définir ou modifier l'action **onmaxdur**, vous devez également spécifier le mot-clé **maxdur** associé à l'argument **onmaxdur**.

Exemples

L'exemple suivant indique de poursuivre l'exécution d'un travail en cours s'il s'exécute toujours après une heure et 20 minutes :

```
MAXDUR 80 ONMAXDUR CONT
```

L'exemple suivant indique d'arrêter un travail en cours si celui-ci s'exécute plus d'une heure et 20 minutes :

```
MAXDUR 80 ONMAXDUR KILL
```

L'exemple suivant indique de poursuivre l'exécution d'un travail en cours s'il s'exécute toujours après avoir dépassé 120 % de sa durée maximale, basée sur la durée estimée la plus récente :

```
MAXDUR 120 % ONMAXDUR KILL
```

mindur

Indique la durée la plus courte au cours de laquelle un travail s'exécute normalement et se termine. Si un travail se termine avant que cette durée minimale soit atteinte, les actions suivantes sont exécutées :

- L'une des actions suivantes est déclenchée : Abend, Confirm ou Continue.
- Le travail apparaît comme ne pas avoir atteint sa durée minimale dans les cas suivants, uniquement si le travail se termine avec le statut SUCCESS :
  - Lors de l'exécution de **showjobs** et **showjobs ;props** à partir de la ligne de commande **conman**, `MinDurationNotReached` s'affiche.
  - A partir de Dynamic Workload Console dans les propriétés du travail.
  - Un message d'information est consigné dans le fichier *rép_base_TWS*`/stdlist/logs/`*aaaammjj*`_TWSMERGE.log`.

Si ce travail est toujours en cours d'exécution lorsque JnextPlan s'exécute, il est inséré dans le flot de travaux USERJOBS. Le paramètre de durée minimale n'est pas conservé pour le travail dans le flot de travaux USERJOBS et ne sera pas surveillé. Pour que le flot de travaux soit reporté et pour éviter que le travail ne soit déplacé vers le flot de travaux USERJOBS, marquez le flot de travaux initial où le paramètre de durée minimale a été spécifié sous la forme d'un flot de travaux carryforward (en paramétrant le mot-clé `carryforward` dans le flot de travaux) si l'option globale `enCarryForward` est définie sur `yes` ; sinon, définissez l'option globale `enCarryForward` sur `all`.

Syntaxe

mindur *heure* | *pourcentage* % **onmindur** *action*

Arguments

heure Indique une durée exprimée à l'aide de la syntaxe *HHHMM*, où :

HHH Représente le nombre d'heures sous la forme d'un nombre compris entre 000 et 500.

MM Représente le nombre de minutes sous la forme d'un nombre compris entre 00 et 59.

pourcentage
Indique le pourcentage de la durée estimée la plus récente. Il peut s'agir d'un nombre compris entre 0 et 1000000.

onmindur *action*
Indique l'action à déclencher sur un travail qui se termine avant sa durée minimale. Les valeurs possibles du paramètre de l'*action* sont :

Abend
Le travail est défini sur le statut **ABEND**.

Confirm
Le travail est défini sur le statut **CONFIRM**. La charge de travail nécessite une confirmation de l'utilisateur pour se poursuivre.

Continue
L'exécution de la charge de travail se poursuit sans qu'aucune action ne soit à prendre.

Exemples

L'exemple suivant indique de poursuivre l'exécution d'une charge de travail même si le travail n'atteint pas une durée minimale d'au moins 80 minutes :

```
MINDUR 120 ONMINDUR CONT
```

L'exemple suivant indique de définir le statut du travail sur Error si le travail ne s'exécute pas pendant 80 minutes au minimum :

```
MINDUR 120 ONMINDUR ABEND
```

L'exemple suivant nécessite qu'un utilisateur confirme le travail lorsqu'il a atteint 50 % ou la moitié de sa durée minimale estimée la plus récente :

```
MINDUR 50 % ONMINDUR CONFIRM
```

needs

Le mot clé **needs** définit les ressources qui doivent être disponibles avant le lancement d'un travail ou d'un flot de travaux. Vous pouvez utiliser le mot clé **needs** soit dans une définition de flot de travaux, soit dans la définition des travaux contenus, mais pas dans les deux à la fois.

Syntaxe

needs [*n*] [*poste_de_travail***#**]*nom_ressource* [,...]

Arguments

n
Indique le nombre d'unités de ressources requises. Les valeurs admises sont comprises entre **1** et **1024** pour chaque instruction **needs**. La valeur par défaut est 1.

poste de travail
Indique le nom du poste de travail sur lequel la ressource est définie localement. Si elle n'est pas spécifiée, la valeur par défaut est le poste de travail sur lequel le travail ou flot de travaux dépendant s'exécute. Les ressources ne peuvent être utilisées comme dépendances que par les travaux ou les flots de travaux exécutés sur le poste de travail où la ressource est définie.

En raison du mécanisme de résolution des dépendances de ressources, une dépendance de ressource au niveau du flot de travaux peut être considérée comme 'locale' (et son utilisation est prise en charge) et non comme 'globale', lorsqu'à la fois le flot de travaux et tous ses travaux sont définis sur le même poste de travail que la ressource.

Cependant, un agent standard et son hôte peuvent faire référence aux mêmes ressources.

nom_ressource
Indique le nom de la ressource.

Commentaires

Un travail ou flot de travaux peut nécessiter un maximum de 1024 unités d'une ressource dans une instruction **needs**. Au moment de l'exécution, chaque instruction **needs** est convertie en *réservation*, chacune contenant un maximum de 32 unités d'une ressource spécifique. 32 réservations au maximum peuvent exister pour une seule ressource, quelle que soit sa quantité d'unités disponible. Si 32 réservations sont déjà définies pour une ressource, le travail ou flot de travaux suivant qui attend cette ressource attend jusqu'à ce qu'une réservation en cours se termine ET que la quantité de ressource requise soit disponible.

Exemples

Dans l'exemple suivant, le lancement du flot de travaux `sked3` est interdit tant que trois unités de `CPUtime` et deux unités de `tapes` ne sont pas disponibles :

```
schedule sked3 on fr
  needs 3 cputime, 2 tapes :
```

La ressource `jlimit` a été définie avec deux unités disponibles. Dans l'exemple suivant, l'exécution simultanée d'un maximum de deux travaux est autorisée dans le flot de travaux `sked4` :

```
schedule sked4 on mo,we,fr:
  joba needs 1 jlimit
  jobb needs 1 jlimit
  jobc needs 2 jlimit  <<runs alone>>
  jobd needs 1 jlimit
end
```

on

Ce mot clé de flot de travaux définit le moment et la fréquence de sélection d'un flot de travaux en vue de son exécution. S'il est omis, le flot de travaux n'est pas ajouté au plan de préproduction. Le mot clé **on** doit suivre le mot clé **schedule**. Pour plus d'informations, voir «except», à la page 251.

Syntaxe

on [**runcycle** *nom*]

[**valide de** *date*] [**valide jusqu'à** *date*]

[**description** *"texte"*]

[**vartable** *nom_table*]

{*date* | *jour* | *agenda* | *demande* | *"icalendar"*} **[,...]**

[**fdignore** | **fdnext** | **fdprev**][**subset** *nom_sous-ensemble* **AND** | **OR**]

Arguments

runcycle *nom*
: Indique un libellé avec un nom usuel pour le cycle d'exécution spécifié dans les lignes suivantes.

valide de *date* **... valide jusqu'à** *date*
: Délimite la période au cours de laquelle le flot de travaux est actif, c'est-à-dire pendant laquelle il est ajouté au plan de production. Notez que

la date spécifiée comme valeur **valide jusqu'à** n'est pas incluse dans le cycle d'exécution et le flot de travaux n'est donc pas actif à cette date.

description *"texte"*
Contient une description du cycle d'exécution.

vartable
Indique le nom de la table de variables que le cycle d'exécution doit utiliser.

date Indique un cycle d'exécution qui s'exécute à des dates spécifiques. La syntaxe utilisée pour ce type de cycle d'exécution est la suivante :

*jjmmaaaa [,aaaammjj][,...]*Par exemple, pour un flot de travaux dont l'exécution est prévue les 25 mai 2009 et 12 juin 2009, la valeur est la suivante :

```
on
20090525,20090612
```

jour Indique un cycle d'exécution qui s'exécute des jours spécifiques. La syntaxe utilisée pour ce type de cycle d'exécution est la suivante :

{mo | tu | we | th | fr | sa | su}Par exemple, pour un flot de travaux qui est planifié pour s'exécuter chaque lundi, la valeur est la suivante :

```
on
mo
```

agenda Dates indiquées dans un agenda portant ce nom. Le nom de l'agenda peut être suivi d'un décalage au format suivant :

{**+** | **-**}*n* {**day[s]** | **weekday[s]** | **workday[s]**}

où :

n Indique le nombre de jours, jours de la semaine ou jours ouvrés.

jours Chaque jour de la semaine.

weekdays
Tous les jours de la semaine sauf le samedi et le dimanche.

workdays
Tous les jours de la semaine excepté les samedis et dimanches (sauf indication contraire par le mot clé **freedays**) et les dates figurant dans un agenda désigné des jours chômés ou dans l'agenda **holidays**.

demande
Sélectionne le flot de travaux uniquement lorsque celui-ci est demandé. Ce paramètre est utilisé pour les flots de travaux qui sont sélectionnés par nom et non par date. Pour empêcher un flot de travaux planifié d'être sélectionné pour le travail **JnextPlan**, redéfinissez-le ON REQUEST.

Remarque : Si vous tentez d'exécuter un flot de travaux contenant plusieurs heures "à la demande" (on request), tenez compte des points suivants :

- Les indicateurs "On request" ont la priorité sur les indicateurs "at".
- Les indicateurs "On request" n'ont jamais la priorité sur les indicateurs "On".

icalendar
Représente une norme utilisée pour spécifier une règle récurrente qui décrit le moment où un flot de travaux s'exécute.

La syntaxe utilisée pour un cycle d'exécution de type *icalendar* est la suivante :

FREQ={DAYLY | WEEKLY | MONTHLY | YEARLY}

[;INTERVAL=[-]*n*]

[;{BYFREEDAY | BYWORKDAY | BYDAY=*liste_jours_semaine* |

BYMONTHDAY=*liste_jours_mois***}]**

où la valeur par défaut du mot clé **INTERVAL** est 1.

A l'aide du mot clé *icalendar*, vous pouvez spécifier qu'un flot de travaux s'exécute :

tous les *n* **jours**
en utilisant le format suivant :

FREQ=DAILY[;INTERVAL=*n*]

où la valeur définie pour **valide de** est le premier jour des dates résultantes.

Par exemple, pour un flot de travaux planifié pour s'exécuter chaque jour, la valeur est la suivante :

```
FREQ=DAILY
```

Pour un flot de travaux planifié pour s'exécuter un jour sur deux, la valeur est la suivante :

```
FREQ=DAILY;INTERVAL=2
```

chaque jour chômé ou ouvré
en utilisant le format suivant :

FREQ=DAILY[;INTERVAL=*n*]

;BYFREEDAY | BYWORKDAY

Par exemple, pour un flot de travaux planifié pour s'exécuter chaque jour chômé, la valeur est la suivante :

```
FREQ=DAILY;BYFREEDAY
```

Pour un flot de travaux planifié pour s'exécuter un jour ouvré sur deux, la valeur est la suivante :

```
FREQ=DAILY;INTERVAL=2;BYWORKDAY
```

toutes les *n* **semaines, à des** *jours de la semaine* **spécifiques**
en utilisant le format suivant :

FREQ=WEEKLY[;INTERVAL=*n*]

;BYDAY=*liste_jours_semaine*

où la valeur définie pour *liste_jours_semaine* est

```
[SU][,MO][,TU][,WE][,TH][,FR][,SA]
```

Par exemple, pour un flot de travaux planifié pour s'exécuter chaque semaine le vendredi et le samedi, la valeur est la suivante :

```
FREQ=WEEKLY;BYDAY=FR,SA
```

Pour un flot de travaux planifié pour s'exécuter toutes les trois semaines, le vendredi, la valeur est la suivante :

```
FREQ=WEEKLY;INTERVAL=3;BYDAY=FR
```

tous les *n* mois, à des dates spécifiques
en utilisant le format suivant :

FREQ=MONTHLY[;INTERVAL=*n*]

;BYMONTHDAY=*liste_jours_mois*

où la valeur définie pour *liste_jours_mois* est représentée par une liste de

[+*nombre_de_jours_à_partir_début_du_mois*]
[-*nombre_de_jours_à_partir_fin_du_mois*]
[*nombre_de_jours_du_mois*]

Par exemple, pour un flot de travaux planifié pour s'exécuter le 27ème jour de chaque mois, la valeur est la suivante :

```
FREQ=MONTHLY;BYMONTHDAY=27
```

Pour un flot de travaux planifié pour s'exécuter tous les six mois, le 15 du mois et le dernier jour du mois, la valeur est la suivante :

```
FREQ=MONTHLY;INTERVAL=6;BYMONTHDAY=15,-1
```

tous les *n* mois, à des jours spécifiques de la semaine
en utilisant le format suivant :

FREQ=MONTHLY[;INTERVAL=*n*]

;BYDAY=*numéro_de_la_semaine_du_mois_jour_semaine*

où la valeur définie pour *numéro_de_la_semaine_du_mois_jour_semaine* est représentée par une liste de

[+*numéro_semaine_à_partir_début_du_mois*]
[-*numéro_semaine_à_partir_fin_du_mois*]
[*jour_semaine*]

Par exemple, pour un flot de travaux planifié pour s'exécuter chaque mois, le premier lundi et le dernier vendredi du mois, la valeur est la suivante :

```
FREQ=MONTHLY;BYDAY=1MO,-1FR
```

Pour un flot de travaux planifié pour s'exécuter tous les six mois, le deuxième mardi du mois, la valeur est la suivante :

```
FREQ=MONTHLY;INTERVAL=6;BYDAY=2TU
```

toutes les *n* années
en utilisant le format suivant :

FREQ=YEARLY[;INTERVAL=*n*]

où la valeur définie pour **valide de** est le premier jour des dates résultantes.

Par exemple, pour un flot de travaux planifié pour s'exécuter chaque année, la valeur est la suivante :

```
FREQ=YEARLY
```

Pour un flot de travaux planifié pour s'exécuter tous les deux ans, la valeur est la suivante :

```
FREQ=YEARLY;INTERVAL=2
```

fdignore | fdnext | fdprev
Indique la règle à appliquer si la date sélectionnée pour l'exécution du travail ou du flot de travaux coïncide avec un jour chômé. Les valeurs possibles sont les suivantes :

fdignore
La date n'est pas ajoutée.

fdnext Le jour ouvré le plus proche qui suit le jour chômé est ajouté.

fdprev
Le jour ouvré le plus proche précédant le jour chômé est ajouté.

[subset *nom_sous-ensemble* **AND | OR]**

subset *nom_sous-ensemble*
Indique le nom du sous-ensemble. Si vous ne spécifiez pas de nom, SUBSET_1 est utilisé par défaut.

AND | OR
Par défaut, les cycles d'exécution dans un sous-ensemble sont dans une relation logique **OR** mais vous pouvez la changer en relation logique **AND** tant que le résultat du groupe de cycle d'exécution est une date ou un ensemble de dates positif (inclusif).

Commentaires

Vous pouvez définir plusieurs instances du mot clé **on** pour le même flot de travaux. Si une définition de flot de travaux contient plusieurs instances **on**, celles-ci doivent être consécutives. Chaque instance correspond à un cycle d'exécution auquel vous pouvez associer une règle de jour chômé.

Chaque instance du mot clé doit contenir l'une des valeurs autorisées par la syntaxe **on**.

Si les heures de début du cycle d'exécution et du flot de travaux sont définies, celle du cycle d'exécution l'emporte lorsque le flot de travaux est planifié avec **JNextPlan**. Si le flot de travaux est lancé avec la commande **submit**, l'heure de début du cycle d'exécution n'est pas utilisée.

Exemples

Dans l'exemple suivant, le flot de travaux sked1 est sélectionné les lundis et les mercredis :

```
schedule sked1 on mo,we
```

Dans l'exemple suivant, le flot de travaux sked3 est sélectionné le 15 juin 2008 et aux dates répertoriées dans l'agenda apdates :

```
schedule sked3 on 6/15/08,apdates
```

Dans l'exemple suivant, le flot de travaux sked4 est sélectionné deux jours avant chaque date figurant dans l'agenda monthend :

```
schedule sked4 on monthend -2 weekdays
```

Dans l'exemple suivant, le flot de travaux testskd1 est sélectionné tous les jours de la semaine sauf les mercredis :

```
schedule testskd1 on weekdays
  except we
```

Dans l'exemple suivant, le flot de travaux testskd3 est sélectionné tous les jours de la semaine à l'exception des 15 et 24 mai 2008 :

```
schedule testskd3 on weekdays
  except 05/16/2008,05/24/2008
```

Dans l'exemple suivant, le flot de travaux testskd4 est sélectionné tous les jours sauf pendant les deux jours précédant toute date figurant dans un agenda intitulé monthend :

```
schedule testskd4 on everyday
  except monthend -2 weekdays
```

Sélectionnez le flot de travaux sked1 pour qu'il s'exécute les lundis et vendredis ainsi que le 29/12/2009. Si les lundis et le 29/12/2009 correspondent à des jours chômés, exécutez le flot de travaux le jour ouvré suivant le plus proche. Si les vendredis correspondent à des jours chômés, exécutez le flot de travaux le jour ouvré précédent le plus proche. Dans cet exemple, les jours chômés sont les samedis, les dimanches et toutes les dates répertoriées dans l'agenda HOLIDAYS par défaut. Les jours ouvrés incluent tous les jours de la semaine du lundi au vendredi ne figurant pas dans l'agenda HOLIDAYS.

```
schedule
sked1
on mo, 12/29/2009 fdnext
on fr fdprev
```

L'exemple suivant montre le résultat de la commande display du flot de travaux testcli défini pour s'exécuter lors de cycles d'exécution différents sur le poste de travail site2 :

```
display js=site2#testcli
```

obtenu dans un format à 120 colonnes par la commande *MAESTROCOLUMNS*=120 avant l'accès à la ligne de commande **composer** :

```
JobstreamName Workstation Draft Valid From Valid To UpdatedBy UpdatedOn LockedBy
------------- ----------- ----- ---------- ------- --------- --------- --------
TESTCLI    SITE2    Y   08/25/2008 -    mdmDBE4  08/25/2008 mdmDBE4

SCHEDULE W5#TESTCLI VALID FROM 08/25/2008 TIMEZONE ACT
DESCRIPTION "Job stream with several run cycle settings."
DRAFT
ON RUNCYCLE M5 VALID FROM 08/25/2008
   DESCRIPTION "monthly"
   "FREQ=MONTHLY;INTERVAL=5;BYMONTHDAY=-3,1"
   ( AT 0000 )
ON RUNCYCLE W4 VALID FROM 08/25/2008
   DESCRIPTION "weekly"
   "FREQ=WEEKLY;INTERVAL=5;BYDAY=MO,WE"
   FDNEXT ( AT 0000 )
ON RUNCYCLE D3 VALID FROM 08/25/2008
   DESCRIPTION "daily"
   "FREQ=DAILY;INTERVAL=2"
   FDPREV ( AT 0000 )
ON RUNCYCLE C2 VALID FROM 08/25/2008
   DESCRIPTION "calendar"
   ITALY +2 DAYS
   ( AT 0000 )
ON RUNCYCLE M6 VALID FROM 08/25/2008
   DESCRIPTION "monthly"
   "FREQ=MONTHLY;INTERVAL=2;BYDAY=1MO,1TH,2WE"
   ( AT 0000 +2 DAYS )
ON RUNCYCLE Y7 VALID FROM 08/25/2008
   DESCRIPTION "yearly"
   "FREQ=YEARLY;INTERVAL=7"
```

```
   ( AT 0100 )
ON RUNCYCLE SS1 VALID FROM 08/25/2008
   08/10/2008,08/18/2008,08/20/2008,08/25/2008
   ( AT 0000 UNTIL 0000 +1 DAYS ONUNTIL SUPPR DEADLINE 0000 +2 DAYS )
EXCEPT RUNCYCLE S1 VALID FROM 08/25/2008
   DESCRIPTION "simple"
   08/26/2008,08/28/2008,08/30/2008,09/13/2008
   ( AT 0000 )

CARRYFORWARD
MATCHING SAMEDAY
FOLLOWS LAB235004#SROBY2.@
FOLLOWS X8#COPYOFJS2.RR
FOLLOWS XA15::TPA
KEYSCHED
LIMIT 22
PRIORITY 15
:
X8#PIPPO AS JOBTC
 CONFIRMED
 PRIORITY 13
 KEYJOB
 FOLLOWS W5#POPO.@
 FOLLOWS X8#JS2.F3
END

AWSBIA291I Total objects: 1
```

L'agenda ITALY est un agenda personnalisé défini dans la base de données qui détermine les jours ouvrés et chômés de l'agenda en vigueur en Italie.

opens

Indique les fichiers qui doivent être disponibles avant le lancement d'un travail ou d'un flot de travaux.

Syntaxe

opens [*poste_de_travail*#]"*file_name*" [**(***qualifiant***)**] [**,**...]

Arguments

poste de travail
: Indique le nom du poste de travail ou de la classe de postes de travail qui contient le fichier. Par défaut, il s'agit du poste de travail ou de la classe de postes de travail du travail ou du flot de travaux dépendant. Si une classe de postes de travail est utilisée, celle-ci doit être identique à celle du flot de travaux qui contient cette instruction.

file_name
: Indique le nom du fichier, placé entre guillemets. Vous pouvez utiliser les paramètres de Tivoli Workload Scheduler dans tout ou partie de la chaîne de noms de fichier. Si un paramètre est utilisé, celui-ci doit être placé entre carets (^). Voir «Définition des variables et des paramètres», à la page 217 pour plus d'informations et d'exemples.

qualifier
: Indique une condition de test valide. Sous UNIX, le qualifiant est transmis à une commande **test**, qui s'exécute en tant que **racine** dans bin/sh.

  Sous Windows, la fonction de test est réalisée en tant qu'utilisateur Tivoli Workload Scheduler.

Les qualifiants acceptés sont les suivants :

-d %p Si le fichier existe et qu'il s'agit d'un répertoire.

-e %p Si le fichier existe.

-f %p Si le fichier existe et qu'il est ordinaire.

-r %p Si le fichier existe et qu'il peut être lu.

-s %p Si le fichier existe et que sa taille n'est pas nulle.

-w %p Si le fichier existe et qu'il peut être écrit.

-a Opérateur booléen AND.

-o Opérateur booléen OR.

Sous UNIX et Windows, l'expression **%p** est utilisée pour transmettre la valeur attribuée à *file_name* à la fonction de test.

Entrer **(notempty)** est équivalent à **(-s %p)**. Si aucun qualifiant n'est indiqué, la valeur par défaut est **(-f %p)**.

Commentaires

Combinaison entre le *chemin de fichier* et les *qualifiants* (limitée à 120 caractères) et le *nom de fichier* (limité à 28 caractères).

Exemples

L'exemple suivant permet de vérifier, avant le lancement de `ux2#sked6`, que le fichier `c:\users\fred\datafiles\file88` du poste de travail `nt5` est accessible en lecture :

```
schedule ux2#sked6 on tu opens nt5#"c:\users\fred\datafiles\file88"
```

L'exemple suivant permet de vérifier, avant le lancement du travail `jobr2`, l'existence des trois répertoires `/jean`, `/marie` et `/roger` sous le répertoire `/users` :

```
jobr2 opens "/users"(-d %p/jean -a -d %p/marie -a -d %p/roger)
```

L'exemple suivant permet de vérifier si `cron` a créé son fichier FIFO avant de lancer le travail `job6` :

```
job6 opens "/usr/lib/cron/FIFO"(-p %p)
```

L'exemple suivant permet de vérifier, avant le lancement du travail `jobt2`, que le fichier `d:\work\john\execit1` du poste de travail `dev3` existe et qu'il n'est pas vide :

```
jobt2 opens dev3#"d:\work\john\execit1"(notempty)
```

L'exemple suivant permet de vérifier, avant le lancement du travail `jobt2`, que le fichier `c:\tech\checker\startf` du poste de travail `nyc` existe et qu'il n'est pas vide :

```
job77 opens nyc#"C:\tech\checker\startf"(-s %p -a -w %p)
```

Sécurité des commandes test(1) :
Sous UNIX, une fonction de sécurité spéciale empêche toute utilisation non autorisée d'autres commandes dans le qualifiant. Par exemple, le fichier ci-dessous contient une commande dans le qualifiant :

```
/users/xpr/hp3000/send2(-n "`ls /users/xpr/hp3000/m*`" -o -r %p)
```

Si le qualifiant ne contient aucune autre commande, les vérifications suivantes sont effectuées :

- L'option locale ***jm no root*** doit avoir la valeur no.
- Dans le fichier de sécurité, l'utilisateur qui documente le programme ou qui ajoute la dépendance Open Files avec une commande **conman adddep**, doit avoir soumis l'accès à un travail avec les attributs suivants :
  name=`cmdstest.fileeq`
  logon=root
  jcl=chemin d'accès des fichiers OPEN
  cpu=poste de travail sur lequel résident les fichiers OPEN

  Notez que `cmdstest` et `fileeq` n'existent pas.

priority

Définit la priorité d'un travail ou d'un flot de travaux. En affectant une priorité différente aux travaux ou flots de travaux, vous déterminez lequel démarre en premier, à condition que les dépendances soient résolues.

En supposant que les travaux et flots de travaux soient prêts à être lancés, si vous définissez une priorité pour les flots de travaux et pour les travaux des flots de travaux :

- Le flot de travaux qui commence le premier est celui dont la priorité est maximale.
- Parmi les travaux du flot de travaux ayant la priorité maximale, le travail qui démarre le premier est celui dont la priorité est maximale.

Syntaxe

priority *nombre* | **hi** | **go**

Arguments

nombre Indique la priorité. Les valeurs admises sont comprises entre **0** et **99**. Une priorité de 0 empêche le lancement du travail ou du flot de travaux.

hi Représente une valeur supérieure à toute valeur pouvant être spécifiée à l'aide d'un nombre. Une fois défini, le travail ou le flot de travaux est immédiatement lancé lorsqu'il ne contient aucune dépendance.

go Représente la plus haute priorité pouvant être définie. Une fois défini, le travail ou le flot de travaux est immédiatement lancé lorsqu'il ne contient aucune dépendance.

Commentaires

Les travaux et les flots de travaux avec des niveaux de priorité **hi** ou **go** sont lancés dès que toutes leurs dépendances sont résolues. Dans ce cas :

- Les flots de travaux remplacent le nombre maximal de travaux de l'unité centrale.
- Les travaux remplacent le nombre maximal de travaux de l'unité centrale, mais ils ne remplacent ni le nombre maximal de travaux planifiés ni la priorité minimale de travail de l'unité centrale.

Exemples

L'exemple suivant illustre la relation entre les priorités des flots de travaux et celles des travaux. Les deux flots de travaux, `sked1` et `sked2` ont les définitions suivantes dans la base de données :

```
schedule sked1 on tu
priority 50
:
job1 priority 15
job2 priority 10
end

schedule sked2 on tu
priority 10
:
joba priority 60
jobb priority 50
end
```

Etant donné que le flot de travaux `sked1` possède la priorité maximale, les travaux sont lancés dans l'ordre suivant : `job1`, `job2`, `joba`, `jobb`.

Si les priorités de flot de travaux étaient identiques, les travaux seraient lancés dans l'ordre suivant : `joba`, `jobb`, `job1`, `job2`.

Si `job2` possède une dépendance **A** et `job1` une dépendance **B** et si la dépendance **A** est résolue (tandis que **B** reste non résolue), alors `job2` démarre avant `job1`, même si la priorité de `job2` est plus faible que celle de `job1`.

prompt

Définit les invites auxquelles une réponse affirmative doit être donnée avant le lancement d'un travail ou d'un flot de travaux.

Syntaxe

prompt *nom_invite* [**,**...]

prompt "[**:** | **!**]*texte*" [,...]

Arguments

nom_invite
: Indique le nom d'une invite de la base de données. Vous pouvez spécifier plusieurs valeurs *nom_invite* séparées par des virgules, mais vous ne pouvez pas mélanger sous le même mot clé **prompt** des invites définies dans la base de données avec des invites littérales.

texte
: Indique une invite littérale sous forme de chaîne de caractères placée entre guillemets (""). Plusieurs chaînes séparées par une barre oblique inversée et **n** (\n) peuvent être utilisées pour des messages longs. Si la chaîne commence par un signe deux-points (:), le message s'affiche mais aucune réponse n'est requise. Si la chaîne commence par un point d'exclamation (!), le message s'affiche, mais il n'est pas enregistré dans le fichier journal. Vous pouvez entrer une barre oblique suivie d'un **n** ("\n") dans le texte pour les retours à la ligne.

  Un ou plusieurs paramètres peuvent être utilisés dans tout ou partie de la chaîne de caractères. Pour utiliser un paramètre, placez son nom entre

carets (^). Voir «Définition des variables et des paramètres», à la page 217 pour plus d'informations et d'exemples.

Remarque : Dans une invite locale, lorsqu'ils ne désignent pas de paramètre, les carets (^) doivent être précédés d'une barre oblique inversée pour ne pas provoquer d'erreur. Dans les invites globales, il n'est pas nécessaire de faire précéder les carets d'une barre oblique inversée.

Exemples

L'exemple suivant illustre des invites littérales et nommées. La première correspond à une invite littérale qui utilise un paramètre intitulé `sys1`. Lorsqu'une seule réponse affirmative est reçue pour l'invite nommée `apmsg`, les dépendances des deux travaux `job1` et `job2` sont respectées.

```
schedule sked3 on tu,th
  prompt "All ap users logged out of ^sys1^? (y/n)"
:
  job1 prompt apmsg
  job2 prompt apmsg
end
```

L'exemple suivant définit une invite littérale qui apparaît sur plusieurs lignes. Il contient une barre oblique inversée suivie de la lettre **n** (\n) à la fin de chaque ligne :

```
schedule sked5 on fr
  prompt "Les travaux du flot de travaux occupent \n
un temps UC énorme.\n
Voulez-vous les lancer maintenant ? (o/n)"
:
  j1
  j2 follows j1
end
```

schedtime

Représente l'heure à laquelle le flot de travaux est positionné dans le plan. La valeur affectée à **schedtime** ne représente pas une dépendance pour le flot de travaux. Pendant que le plan de production est en cours, l'instance de travail ou de flot de travaux démarre parfois son traitement avant l'heure définie dans le mot clé **schedtime** si toutes ses dépendances sont résolues et si sa priorité lui permet de démarrer.

Syntaxe

schedtime *heure* [**timezone** | **tz** *fuseau_horaire*][+*n* **day**[**s**]] [**,...**]

Arguments

heure Indique une heure de la journée au format : HHHHmm. Les valeurs admises sont comprises entre **0000** et **320000**.

tzname Indique le fuseau horaire à utiliser lors du calcul de l'heure de début. Voir Chapitre 16, «Gestion des fuseaux horaires», à la page 653 pour les noms des fuseaux horaires. Il s'agit par défaut du fuseau horaire correspondant au poste de travail sur lequel le travail ou le flot de travaux est lancé.

n Indique un décalage, en jours, à partir de la date et de l'heure de début planifiées.

Commentaires

A la différence du mot clé **at**, le mot clé **schedtime** ne représente pas une dépendance horaire : il n'indique pas d'heure avant laquelle un travail ou flot de travaux ne peut pas démarrer. Au lieu de cela, la valeur spécifiée dans le mot clé **schedtime** est utilisée uniquement pour positionner l'instance de travail ou de flot de travaux spécifique dans le plan de préproduction. Pendant que le plan de production est en cours, l'instance de travail ou de flot de travaux démarre parfois son traitement avant l'heure définie dans le mot clé **schedtime** si toutes ses dépendances sont résolues et si sa priorité lui permet de démarrer.

Pour plus d'informations sur l'identification des prédécesseurs dans le plan de préproduction à l'aide du mot clé **schedtime**, voir «Gestion des dépendances externes de prédécesseur/successeur des travaux et des flots de travaux», à la page 65.

Les mots clés **at** et **schedtime** s'excluent mutuellement. Si **schedtime** n'est pas spécifié et si le mot clé **at** est spécifié dans le travail ou le flot de travaux, sa valeur est utilisée pour positionner l'instance dans le plan de préproduction.

Si aucun des mots clés **at** et **schedtime** n'est spécifié dans la définition du travail ou du flot de travaux, le système adopte par défaut la valeur affectée à l'option globale *startOfDay* définie sur le gestionnaire de domaine maître.

Pour les flots de travaux comportant une définition **schedtime**, la valeur de la zone `Heure de début` affichée sur Dynamic Workload Console dépend du paramétrage de l'option globale `enPreventStart` (qui détermine si les flots de travaux sans dépendance `at` peuvent démarrer immédiatement, sans attendre le cycle d'exécution spécifié dans le flot de travaux) :

- Si `enPreventStart` est défini sur la valeur `yes`, l'heure de démarrage est "12:00 AM" convertie dans le fuseau horaire spécifié dans l'interface graphique.
- Si `enPreventStart` n'est pas défini sur la valeur `no`, la zone de l'heure de démarrage est vide.

Exemples

Dans les exemples suivants, on considère que le jour de traitement Tivoli Workload Scheduler commence à 6 heures du matin.

- Le flot de travaux suivant qui doit être exécuté tous les mardis est planifié pour démarrer à 3:00 le mercredi matin. Les deux travaux qu'il contient sont lancés dès que possible après le début du traitement du flot de travaux.

  ```
  schedule sked7 on tu schedtime 0300:
  job1
  job2
  end
  ```

- Le fuseau horaire du poste de travail `sfran` correspond à l'heure du pacifique America/Los_Angeles et le fuseau horaire du poste de travail `nycity` correspond à l'heure de la côte Est America/New_York. Le flot de travaux `sked8` est sélectionné pour s'exécuter le vendredi. Il est planifié pour démarrer sur le poste de travail `sfran` à 10 heures du matin, heure de Los Angeles, le samedi (tel que spécifié par le décalage `+ 1 day`). Le travail `job1` est lancé sur `sfran` dès que possible après le début du traitement du flot de travaux. Le travail `job2` est lancé sur `sfran` à 14 heures America/New_York (11:00 a.m.

America/Los_Angeles) le samedi. Le travail job3 est lancé sur le poste de travail nycity à 16 heures sur le fuseau horaire America/New_York (1:00 p.m. America/Los_Angeles) le samedi.

```
sfran#schedule sked8 on fr schedtime 1000 + 1 day:
job1
job2 at 1400 tz America/New_York
nycity#job3 at 1600
end
```

schedule

Indique le nom du flot de travaux. A l'exception des commentaires, il doit correspondre au premier mot clé d'un flot de travaux et être suivi du mot clé **on**.

Syntaxe

schedule [*poste_de_travail***#**]*nom_flot_travaux*

[**timezone** | **tz** *fuseau_horaire*]

Arguments

poste de travail
: Indique le nom du poste de travail sur lequel le flot de travaux est lancé. Par défaut, il s'agit du poste de travail sur lequel **composer** s'exécute pour ajouter le flot de travaux.

nom_flot_travaux
: Indique le nom du flot de travaux. Il doit commencer par une lettre et peut contenir des caractères alphanumériques, des tirets et des traits de soulignement. Il peut comporter jusqu'à 16 caractères.

timezone | **tz** *fuseau_horaire*
: Indique le fuseau horaire à utiliser lors de la gestion du flot de travaux. Ce paramètre est ignoré si l'option globale *enTimeZone* est définie sur **no** sur le gestionnaire de domaine maître. Pour plus d'informations sur les paramètres de fuseau horaire, voir Chapitre 16, «Gestion des fuseaux horaires», à la page 653.

Commentaires

Dans une définition de flot de travaux, vous pouvez définir un fuseau horaire pour l'ensemble du flot de travaux en utilisant le mot clé **timezone** dans l'intervalle de validité, ou en spécifiant des restrictions temporelles à l'aide des mots clés **at**, **until** ou **deadline**.

Vous pouvez également définir un fuseau horaire pour un travail contenu dans un flot de travaux en définissant les mots clés **at**, **until** ou **deadline** pour ce travail.

Que vous soyez en train de définir un travail ou un flot de travaux, si vous utilisez un fuseau horaire dans un restriction temporelle, par exemple, **at**, vous devez utiliser le même fuseau horaire lorsque vous spécifiez les autres restrictions temporelles, telles que **deadline** et **until**.

Dans une définition de flot de travaux, vous pouvez définir un fuseau horaire pour l'ensemble du flot de travaux et pour les travaux qu'il contient. Ces fuseaux horaires peuvent être différents, auquel cas le fuseau horaire défini pour le travail est converti dans le fuseau horaire défini pour le flot de travaux.

Pour gérer tous les paramètres de fuseau horaire possibles, la conversion de fuseau horaire effectuée lors du traitement des travaux et des flots de travaux dans l'ensemble du réseau Tivoli Workload Scheduler respecte les critères suivants :

1. Si aucun fuseau horaire n'est défini pour un travail contenu dans un flot de travaux, ce travail hérite du fuseau horaire défini sur le poste de travail où il est censé s'exécuter.
2. Si aucun fuseau horaire n'est défini pour un flot de travaux, le fuseau horaire utilisé correspond à celui déjà défini sur le poste de travail où le flot de travaux est censé s'exécuter.
3. Si aucun des fuseaux horaires mentionnés n'est défini, le fuseau horaire utilisé est celui défini sur le gestionnaire de domaine maître.

Exemples

Voici la définition d'un fuseau horaire du travail `sked8` exécuté sur le poste de travail `sfran` sur lequel le fuseau horaire `America/New_York` est défini. Le fuseau horaire défini pour l'exécution de `job2` sur le poste de travail `LA` est défini sur `America/Los_Angeles`.

```
schedule sfran#sked8
tz America/New_York
on fr at 1000 + 1 day:
job1
LA#job2 at 1400 tz America/Los_Angeles
end
```

timezone

Indique (au niveau du flot de travaux) le fuseau horaire utilisé pour calculer l'heure à laquelle le flot de travaux doit démarrer le traitement.

Syntaxe

timezone | **tz** *fuseau_horaire*

Arguments

tzname Indique le nom du fuseau horaire. Voir Chapitre 16, «Gestion des fuseaux horaires», à la page 653 pour les noms des fuseaux horaires.

Commentaires

Le fuseau horaire spécifié au niveau du flot de travaux s'applique à toutes les définitions temporelles des cycles d'exécution et des restrictions temporelles (définies par les mots clés `at`, `deadline`, `schedtime` et `until`).

Si vous indiquez un fuseau horaire pour le flot de travaux et un autre pour la restriction temporelle, ils doivent être identiques.

Si vous n'indiquez aucun fuseau horaire, tant au niveau du flot de travaux que de la restriction temporelle, le fuseau horaire spécifié sur le poste de travail est utilisé.

until

Selon la définition d'objet à laquelle appartient le mot clé until, spécifie la dernière heure à laquelle un flot de travaux doit être terminé ou la dernière heure à laquelle un travail peut être lancé.

Syntaxe

until *heure* [**timezone** | **tz** *fuseau_horaire*][+*n* **day**[**s**]][**absolute** | **abs**][**onuntil** *action*]

Arguments

heure Indique l'heure. Les valeurs admises sont comprises entre **0000** et **2359**.

tzname Indique le fuseau horaire à utiliser lors du calcul de l'heure. Voir Chapitre 16, «Gestion des fuseaux horaires», à la page 653 pour les noms des fuseaux horaires. Il s'agit par défaut du fuseau horaire correspondant au poste de travail sur lequel le travail ou le flot de travaux est lancé.

Remarque : Si une heure **until** et une heure **at** ou **deadline** sont spécifiées, les fuseaux horaires doivent être identiques.

n Indique un décalage, en jours, à partir de la date et de l'heure de début planifiées.

absolute
Indique que la date **until** est fonction du jour calendaire plutôt que du jour de production.

onuntil *action*
Selon la définition d'objet à laquelle appartient le mot clé until, spécifie :

- l'action à effectuer sur un travail ou flot de travaux dont l'heure until a été atteinte mais qui n'a pas encore démarré.
- L'action à prendre sur un flot de travaux dont le temps until a expiré alors que ce flot de travaux ne s'est pas encore terminé à l'état SUCC.

Les valeurs possibles du paramètre de l'*action* sont :

suppr Le travail ou le flot de travaux et tous les travaux ou flots de travaux dépendants ne s'exécutent pas. Il s'agit du comportement par défaut.

Une fois que l'heure until est arrivée à expiration pour un flot de travaux, l'état de ce dernier est calculé en fonction des règles habituelles ; les travaux supprimés ne sont pas pris en compte dans ce calcul. Dans le cas où le flot de travaux contient au moins un travail **every**, il est à l'état HOLD.

Une fois que l'heure until arrive à expiration pour un travail, ce dernier passe à l'état HOLD ou conserve tout état antérieur qui soit un état final.

Si l'heure **until** est transmise avec les options **onuntil suppr** et **carryforward**, le flot de travaux est *reporté* par **JnextPlan** uniquement si la date **until** correspond à la date d'exécution de JnextPlan. Si les dates d'exécution de **until** et JnextPlan ne sont pas identiques, le flot de travaux n'est pas *reporté*.

cont Le travail ou flot de travaux s'exécute lorsque toutes les conditions requises sont remplies. Un message apparaît alors dans le journal lorsque l'heure définie par until est dépassée.

Si l'heure **until** est transmise avec les options **onuntil cont** et **carryforward**, le flot de travaux est toujours *reporté* par **JnextPlan**.

canc Un travail ou flot de travaux est annulé lorsque l'heure **until** définie est atteinte. Lorsque le mot clé *onuntil canc* est utilisé sur des travaux, l'opération d'annulation sur le travail est émise par

l'agent FTA sur lequel ce travail s'exécute. Tout travail ou flot de travaux qui dépendait de l'accomplissement d'un travail ou flot de travaux qui a été annulé, s'exécute car la dépendance n'est plus appliquée.

Si l'heure **until** est transmise avec les options **onuntil canc** et **carryforward**, le flot de travaux n'est pas *reporté* par **JnextPlan** car il est déjà annulé.

Remarque : Lorsque le mot clé *onuntil canc* est utilisé au niveau d'un flot de travaux, définissez comme propriétaire du flot de travaux le poste de travail le plus haut placé dans la hiérarchie de l'environnement de planification parmi tous les postes de travail qui possèdent des travaux appartenant à ce flot de travaux.

Remarque : Les mots clés **until** et **deadline** peuvent être utilisés dans la même définition, mais ils doivent être exprimés avec le même paramètre de fuseau horaire.

Exemples

Dans l'exemple suivant, le programme interdit tout lancement du processus `sked1` après 17 heures les mardis :

```
schedule sked1 on tu until 1700:
```

Le programme lance `sked1` à 17:00, lorsque son heure **until** est atteinte :

```
schedule sked1 until 1700 onuntil cont
```

Le programme lance le travail `job1` entre 13 et 17 heures pendant les jours de la semaine :

```
schedule sked2 on weekdays:
  job1 at 1300 until 1700
end
```

Dans l'exemple suivant, le programme lance le travail `joba` toutes les 15 minutes entre 22:30 et 23:30 les lundis :

```
schedule sked3 on mo:
  joba at 2230 every 0015 until 2330
end
```

Dans l'exemple suivant, le programme lance le flot de travaux `sked4` les dimanches entre 8 et 13 heures. Les travaux doivent être lancés selon l'intervalle suivant :

```
schedule sked4 on fr at 0800 + 2 days
  until 1300 + 2 days
:
  job1
  job2 at 0900 <<launched on sunday>>
  job3 follows job2 at 1200 <<launched on sunday>>
end
```

Dans l'exemple suivant, le programme lance le flot de travaux `sked8` tous les jours de la semaine à 16:00 et il doit être terminé à 17:00. Si le flot de travaux n'est pas terminé à 17:00, il est considéré comme étant en retard. Les travaux doivent être lancés comme suit : `job1` démarre à 16:00 ou au plus tard à 16:20. Si le travail `job1` n'a pas encore démarré à 16:20, le programme consigne un message dans le journal et job1 commence son exécution. Quant au travail `job2`, il s'exécute à 16:30 ou au plus tard à 16:50. Si le travail `job2` n'a pas encore démarré à 16:50, il est annulé.

```
schedule sked8 on weekdays at 1600 deadline 1700 :
   job1 at 1600 until 1620 onuntil cont
   job2 at 1630 until 1650 onuntil canc
end
```

Dans l'exemple suivant, le programme lance le flot de travaux `sked01`. Si l'événement **until** intervient, le programme exécute le flot de travaux `sked02` car le flot de travaux `sked01` passe à l'état SUCC. Par contre, le flot de travaux `sked03` n'est pas exécuté car il possède une dépendance horaire ponctuelle sur le travail `job01`, qui n'a pas été libérée.

```
SCHEDULE sked01 on everyday:
job01 until  2035 onuntil suppr
end

SCHEDULE sked02 on everyday follows sked01.@
:
job02
end

SCHEDULE sked03 on everyday follows sked01.job01
:
job03
END
```

validfrom/validto

Vous pouvez définir pour un flot de travaux un intervalle de validité qui correspond à une période du plan de préproduction et qui comprend le flot de travaux. L'intervalle de validité est défini à l'aide du mot clé **validfrom** dans la définition du flot de travaux.

Syntaxe

validfrom *date*

Arguments

validfrom *date*

Définit la date à partir de laquelle le flot de travaux est actif. Il doit faire partie d'un plan de production si celui-ci inclut cette date.

Commentaires

Différentes *versions* du même flot de travaux peuvent être définies via la création de différents flots de travaux ayant le même nom et le même poste de travail mais pas le même intervalle de validité. Le concept de versions du même flot de travaux partageant les mêmes valeurs *nom_flot_travaux* et *nom_poste_travail* est essentiel lors de la gestion de la dépendance par rapport à ce flot de travaux. En fait, lorsque vous définissez une dépendance de prédécesseur/successeur externe par rapport à un flot de travaux, vous identifiez le flot de travaux remplacé en utilisant les valeurs *nom_flot_travaux* et *nom_poste_travail*. Le flot de travaux identifié comme dépendance est celui dont l'intervalle de validité se situe dans la période d'activité de la dépendance.

Si vous modifiez la valeur *nom_flot_travaux* ou *nom_poste_travail* dans une version du flot de travaux, la modification est répercutée dans toutes ses versions.

Si vous verrouillez une version du flot de travaux, toutes les versions de ce flot de travaux le sont également.

Si vous modifiez le nom d'un travail défini dans une version du flot de travaux, le nouveau nom de travail est propagé dans toutes les versions du flot de travaux. Par conséquent, si vous modifiez une autre valeur que *nom_flot_travaux* ou *nom_poste_travail*, les associations entre les flots de travaux internes et externes restent identiques.

Lors de la définition d'une version d'un flot de travaux, il vous suffit d'entrer la date **validfrom** pour que la date **validto** soit automatiquement définie sur la valeur de la date **validfrom** de la version suivante. La date **validto** s'affiche lors de l'exécution des commandes **list** et **display** lorsque la valeur 120 est attribuée à *MAESTROCOLUMNS*. Les différentes versions du même flot de travaux continuent à partager les zones du nom et du poste de travail après leur création. Si vous modifiez le nom d'une version ou le poste de travail sur lequel il a été défini, la modification s'applique à toutes les versions de ce flot de travaux.

Remarque : Si les mots clés utilisés dans la définition du flot de travaux sont **validfrom** et **validto**, les mots clés de filtrage correspondants utilisés lors de l'émission de commandes sur des définitions stockées dans la base de données sont **validfrom** and **validto**. Pour plus d'informations, voir Chapitre 9, «Gestion des objets dans la base de données - composer», à la page 301.

La date spécifiée comme valeur **validto** n'est pas incluse dans le cycle d'exécution et le flot de travaux n'est donc pas actif à cette date.

vartable

Grâce à ces tables, vous attribuez différentes valeurs à la même variable, et réutilisez donc la même variable dans les définitions de travail et lors de la définition des invites et des dépendances de fichier.

Syntaxe

vartable *nom_table*

Arguments

vartable *nom_table*
Nom de la table de variables. Ce nom peut comporter jusqu'à 80 caractères alphanumériques, incluant les tirets (-) et les traits de soulignement (_), et doit commencer par une lettre.

Définition de règle d'événement

Une règle d'événement de planification définit un ensemble d'actions qui doivent être exécutées à l'occurrence de conditions d'événement spécifiques. La définition d'une règle d'événement établit une corrélation entre des événements et déclenche des actions.

Une définition de règle d'événement est identifiée par un nom de règle et par un ensemble d'attributs qui spécifient si la règle est en mode brouillon ou non, l'intervalle de temps pendant lequel elle est active, la durée de sa validité et d'autres informations nécessaires pour décider du moment où les actions sont déclenchées. Elle inclut des informations liées aux événements spécifiques (`eventCondition`) que la règle doit détecter et aux actions spécifiques qu'elle doit déclencher en cas de détection ou de dépassement du délai d'attente de ces événements(`action`). Des règles complexes peuvent inclure plusieurs événements et plusieurs actions.

Pour une présentation de la planification des règles d'événement, voir Chapitre 7, «Exécution de l'automatisation de la charge de travail gérée par événements», à la page 127.

Syntaxe

Vous définissez des règles d'événement directement en XML avec tout éditeur XML. Vous pouvez configurer une variable d'environnement sur votre ordinateur pour ouvrir automatiquement un éditeur XML de votre choix afin de travailler avec les définitions de règle d'événement. Pour de plus amples informations, voir «Editeur Composer», à la page 302. Le langage XML décrivant la règle d'événement doit correspondre au schéma de langage de règle défini dans les fichiers `EventRules.xsd` et `FilteringPredicate.xsd`.

Les schémas linguistiques des règles définies dans les fichiers `eventRules.xsd` et `FilteringPredicate.xsd` permettent de valider les définitions de règles et, selon l'éditeur XML dont vous êtes équipé, de fournir une aide à la syntaxe. Les fichiers sont situés dans le sous-répertoire `schemas` du répertoire d'installation de Tivoli Workload Scheduler.

Le tableau suivant répertorie tous les éléments de langage utilisés pour définir une règle d'événement. Le tableau 49 explique la signification de la notation qui suit chaque élément de langage. *n* représente un nombre non limité.

Tableau 49. Explication de la notation définissant le nombre d'occurrences pour un élément de langage.

Notation	Signification
(0, 1)	0 indique que l'élément de langage est facultatif. 1 indique que s'il est codé, une seule occurrence est autorisée.
(0, n)	0 indique que l'élément de langage est facultatif. n indique que s'il est codé, plusieurs occurrences sont autorisées.
(1, 1)	Le premier 1 indique que l'élément de langage est obligatoire. Le second 1 indique qu'une seule occurrence est autorisée.
(1, 2)	1 indique que l'élément de langage est obligatoire. 2 indique que 2 occurrences sont obligatoires.
(1, n)	1 indique que l'élément de langage est obligatoire. n indique que plusieurs occurrences sont autorisées.

- eventRule name=" " ruleType=" " isDraft=" " (1, 1)
  - description (0, 1)
  - timeZone (0, 1)
  - validity from=" " to=" " (0, 1)
  - activeTime start=" " end=" " (0, 1)
  - timeInterval amount=" " unit=" " (0, 1)
  - eventCondition eventProvider=" " eventType=" " (1, n)
    - scope (0, 1)
    - filteringPredicate (0, 1)
      - attributeFilter name=" " operator="eq" (0, n)
        - value (1, n)
      - attributeFilter name=" " operator="ne" (0, n)
        - value (1, n)
      - attributeFilter name=" " operator="le" (0, n)

      – value (1, 1)
    • attributeFilter name=" " operator="ge" (0, n)
      – value (1, 1)
    • attributeFilter name=" " operator="range" (0, 1)
      – value (1, 2)
  – correlationAttributes (0, 1)
    - attribute name=" " (1, n)
  – action actionProvider=" " actionType=" " responseType=" " (0, n)
    - description (0, 1)
    - scope (0, 1)
    - parameter name=" "(1, n)
    - value (1, 1)

Les définitions de règle d'événement sont regroupées en ensembles de règle d'événement.
• eventRuleSet (1, 1)
  – eventRule (1, n)

Utilisez l'élément de langage `eventRuleSet` également si vous devez entourer une seule définition de règle.

Remarque : Aucun commentaire que vous écrivez en XML, selon la forme `<!--texte-->`, n'est enregistré dans la base de données. A l'ouverture suivante d'une définition de règle, les commentaires que vous avez écrits la première fois n'apparaissent pas. Préférez l'attribut `description` pour écrire des informations additionnelles.

Arguments

Les mots clés décrivant une règle d'événement correspondent aux balises XML suivantes :

eventRule

L'objet de planification qui englobe la définition de plusieurs conditions d'événement et de plusieurs actions de règle en complément à un ensemble d'attributs qui définissent le moment où la règle est activée. Un élément `eventRule` inclut généralement :
• Un grand nombre d'attributs de règle obligatoires et facultatifs
• Une ou plusieurs conditions d'événement
• Une ou plusieurs actions de règle, bien que des règles sans action soient également autorisées.

Les attributs de règle sont :

• Attributs obligatoires :

name Le nom de la règle d'événement. Il peut avoir jusqu'à 40 caractères alphanumériques et doit commencer par une lettre. Il ne peut pas contenir d'espaces. Les caractères de soulignement (`_`) et le tiret (`-`) sont autorisés.

ruleType
Le type de règle est basé sur le nombre d'événements et sur leur corrélation que la règle est définie pour détecter. Il peut s'agir de l'une des règles suivantes :

filter La règle est activée sur détection d'un événement spécifique unique.

sequence
La règle est activée lorsqu'une séquence ordonnée d'événements survient dans un intervalle de temps spécifique.

set La règle est activée lorsqu'une séquence non ordonnée d'événements survient dans un intervalle de temps spécifique.

Les règles de type `set` et `sequence` peuvent également être activées sur **timeout**, lorsque un ou plusieurs événements se produisent mais que la séquence complète ne se produit pas dans la fenêtre de temps spécifiée.

isDraft
Indique si la définition de règle est actuellement activée. Les valeurs peuvent être **yes** ou **no**. La valeur par défaut est **no**.

- Attributs facultatifs :

description
Une description de la règle. Il peut contenir jusqu'à 120 caractères.

Vous devez écrire tout caractère spécial XML que vous pourriez utiliser selon la notation XML spéciale, tel que :
- `&` pour &
- `>` pour >
- `<` pour <
- `&quot;` pour "

etc.

timeZone
Indique un fuseau horaire différent pour l'exécution de la règle. Le fuseau horaire par défaut est celui défini sur le poste de travail où réside le serveur de traitement d'événements.

L'application de l'heure d'été (DST) a un impact sur l'intervalle `activeTime` (décrit plus loin) des règles d'événement :
- Au jour d'activation de l'heure d'été (l'horloge étant alors avancée d'une heure), l'exécution des règles comprises dans l'heure absorbée par l'application de l'heure d'été est retardée d'une heure. Exemple : 2:10 devient 3:10.
- Au jour de désactivation de l'heure d'été (l'horloge étant alors retardée d'une heure), les règles comprises dans l'heure dupliquée par l'application de l'heure d'été sont considérées comme devant être exécutées sans décalage.

validity
Indique la période de validité de la règle en termes de :

from *aaaa-mm-jj*
La période de validité commence à minuit (du fuseau horaire de la règle) le jour spécifié.

to *aaaa-mm-jj*
La période de validité se termine à minuit (du fuseau horaire de la règle) le jour spécifié.

activeTime
Spécifie la fenêtre de temps d'activité de la règle pour chaque jour de validité en termes de :

start *hh:mm:ss*
Le début de la fenêtre de temps lorsque la règle est active en heures, minutes et secondes.

end *hh:mm:ss*
La fin de la fenêtre de temps pendant laquelle la règle est active en heures, minutes et secondes. Si l'heure est antérieure à **start** *hh:mm:ss*, elle fait référence au jour suivant.

timeInterval
S'applique aux règles qui incluent des actions de dépassement de délai (timeout). Indique l'intervalle de temps à l'intérieur duquel tous les événements spécifiés dans la règle doivent avoir été reçus avant qu'une action de délai de dépassement corrective ne soit démarrée. L'intervalle de temps commence au moment où le premier événement spécifié dans la règle est détecté. Indique l'intervalle de temps en termes de :

quantité
La durée de l'intervalle de temps en unités de temps.

unit L'unité de temps est l'une des suivantes :
- heures
- secondes
- millisecondes

Cet attribut est obligatoire lorsqu'il y a des actions de dépassement de délai dans la définition de règle d'événement.

eventCondition
La condition d'événement est composée des attributs suivants :

- Attributs obligatoires :

eventProvider
Identifie le fournisseur de moniteur d'événements qui peut capturer un type d'événement. Les fournisseurs d'événements fournis à l'installation sont :

TWSObjectsMonitor
Surveille le statut des objets du plan Tivoli Workload Scheduler. Ce fournisseur d'événements s'exécute sur chaque agent Tivoli Workload Scheduler et envoie les événements au serveur de traitement d'événement.

TWSApplicationMonitor
Surveille les processus Tivoli Workload Scheduler, les systèmes de fichiers et la boîte de message.

FileMonitor
Surveille les événements affectant les fichiers.

eventType
Indique le type d'événement à surveiller. Chaque événement peut être associé à un fournisseur d'événements. Les tableaux suivants répertorient les types d'événement par fournisseur d'événements. Pour afficher les propriétés de chaque type d'événement, accédez à l'Annexe A, «Automatisation de la charge de travail commandée par les événements, définitions des événements et des actions», à la page 707.

Le tableau 50 répertorie les événements TWSObjectsMonitor.

Tableau 50. Evénements TWSObjectsMonitor.

Type d'événement	Heure d'envoi de l'événement.
JobStatusChanged	Modification du statut d'un travail
JobUntil	La dernière heure de début possible d'un travail est dépassée
JobSubmit	Un travail est soumis
JobCancel	Un travail est annulé
JobRestart	Un travail est relancé
JobLate	Un travail prend du retard
JobPromoted	Un travail dans un réseau critique approche l'heure de début critique et n'a pas encore démarré
JobRiskLevelChanged	• Un nouveau travail critique est ajouté au plan avec le niveau de risque défini sur élevé ou potentiel • Le niveau de risque d'un travail est défini sur risque élevé ou risque potentiel et évolue • Un travail critique avec le niveau de risque défini sur élevé ou potentiel est supprimé du plan Un événement n'est pas envoyé si le travail critique se trouve dans le flot de travaux nommé JOBS.
JobExceededMaximumDuration	Un travail dépasse la durée maximale établie pour le travail
JobDidnotReachMinimumDuration	Un travail ne s'exécute pas assez longtemps pour atteindre la durée minimale établie pour le travail
JobStreamStatusChanged	Le statut d'un flot de travaux change
JobStreamCompleted	Un flot de travaux s'est achevé
JobStreamUntil	La dernière heure de début possible d'un flot de travaux est dépassée
JobStreamSubmit	Un flot de travaux est soumis
JobStreamCancel	Un flot de travaux est annulé
JobStreamLate	Un flot de travaux prend du retard
WorkstationStatusChanged	Un poste de travail est démarré ou arrêté
ApplicationServerStatusChanged	L'instance WebSphere Application Server s'est arrêtée ou redémarre
ChildWorkstationLinkChanged	Le poste de travail parent défini dans la règle d'événement établi ou supprime un lien avec son poste de travail parent (le poste de travail parent envoie l'événement)
ParentWorkstationLinkChanged	Le poste de travail parent établit ou supprime un lien avec le poste de travail défini dans la règle d'événement (le poste de travail enfant envoie l'événement)
PromptStatusChanged	Une invite est émise ou reçoit une réponse
ProductAlert	Le fichier Symphony du poste de travail spécifié dans l'événement contient un enregistrement endommagé

Remarque :

Toute modification effectuée sur un poste de travail référencé dans une règle n'est pas signalée dans la règle. Par exemple, si vous modifiez, mettez à jour ou supprimez un poste de travail qui est référencé dans une règle, la règle ne tient pas compte du changement et continue de considérer le poste de travail comme il était quand il a été inclus dans la règle. Une règle dont le type d'événement est **ParentWorkstationLinkChanged** n'est pas applicable lorsque le poste de travail des filtres est défini comme agent, pool, pool dynamique ou moteur distant et que l'attribut**ParentWorkstation** est défini sur courtier. Pour surveiller une modification changement de statut de lien entre le serveur courtier de la charge de travail et un poste de travail géré par un serveur courtier d'une charge de travail, définissez une règle sans type d'événement égal à **ChildWorkstationLinkChanged**.

Une règle avec type d'événement égal à **ChildWorkstationLinkChanged** fonctionne uniquement lorsque le poste de travail courtier est associé, dissocié, arrêté ou démarré. Si le changement de statut du lien est dû à une opération d'arrêt ou de démarrage sur le poste de travail agent avec les commandes **StartupLwa** et **ShutdownLwa**, aucune action n'est initialisée. Pour surveiller ou démarrer des opérations sur les postes de travail agent, définissez une règle avec type d'événement égal à **WorkstationStatusChanged**.

Le tableau 51 répertorie les événements `TWSApplicationMonitor`.

Tableau 51. Evénements `TWSApplicationMonitor`.

Type d'événement	Heure d'envoi de l'événement.
MessageQueuesFilling	Une boîte aux lettres dépasse la valeur totale du pourcentage.
TivoliWorkloadSchedulerFileSystemFilling	Le système de fichiers où l'instance Tivoli Workload Scheduler est installée dépasse la valeur totale du pourcentage.
TivoliWorkloadSchedulerProcessNotRunning	Un processus spécifié n'est pas en cours d'exécution.

Le tableau 52 répertorie les événements `FileMonitor`.

Tableau 52. Evénements `FileMonitor`.

Type d'événement	Heure d'envoi de l'événement.
FileCreated	Un fichier est créé
FileDeleted	Un fichier est supprimé
ModificationCompleted	Un fichier est modifié (l'événement est envoyé uniquement si deux cycles de surveillance consécutifs se sont écoulés depuis la création ou la modification du fichier sans changement additionnel détecté)
LoggedMessageWritten	Une chaîne spécifique est trouvée dans un fichier (cet événement peut être utilisé pour surveiller les journaux d'application ou système)

- Attributs facultatifs :

scope Un ou plusieurs attributs de qualification qui décrivent l'événement. Il peut contenir jusqu'à 120 caractères. La portée est automatiquement générée à partir de ce qui est défini dans le fichier XML. Elle ne peut pas être définie par les utilisateurs.

filteringPredicate
Le prédicat de filtrage définit les conditions d'événement qui doivent être surveillées pour chaque type d'événement. Il est composé de :

attributeFilter
Le filtre d'attribut est un attribut particulier du type d'événement qui doit être surveillé :

– est défini par les éléments suivants :

name L'attribut ou nom de propriété du type d'événement qui doit être surveillé. Voir «Fournisseurs d'événements et définitions», à la page 707 pour une liste des noms de propriété pour chaque type d'événement.

opérateur
peut être :
- `eq` (égal à)
- `ne` (différent de)
- `ge` (égal ou supérieur à)
- `le` (égal ou inférieur à)
- `range` (plage)

– Inclut une ou plusieurs :

value La valeur avec laquelle l'opérateur doit concorder.

Remarque : Chaque type d'événement a un grand nombre d'attributs obligatoires ou noms de propriété. Les noms de propriété obligatoires n'ont pas tous des valeurs par défaut. Tous les noms de propriété obligatoires sans valeur par défaut doivent avoir un prédicat de filtrage.

correlationAttributes
Les attributs de corrélation permettent de diriger la règle pour créer une instance de règle séparée pour chaque groupe d'événements qui partagent des caractéristiques communes. Généralement, chaque règle active a une seule instance de règle qui s'exécute dans le serveur de traitement d'événements. Cependant, il arrive que la même règle soit requise pour différents groupes d'événements, souvent liés à différents groupes de ressources. Utiliser un ou plusieurs attributs de corrélation permet de diriger une règle afin de créer une copie de règle séparée pour chaque groupe d'événements ayant des caractéristiques communes. A utiliser avec les types de règle `set` et `sequence`.

Vous pouvez spécifier un ou plusieurs attributs. Chaque attribut est défini par :

attribute name=" "
L'attribut objet que vous corrélez.

action L'action qui doit être déclenchée si l'événement est détecté. Les définitions de règle d'événement avec les événements mais sans les actions sont syntaxiquement acceptées bien qu'elles puissent ne pas avoir de

signification pratique. Vous pouvez enregistrer de telles règles à titre de version préliminaire et ajouter des actions ultérieurement avant qu'elles ne soient déployées.

- Est défini par les attributs obligatoires suivants

  actionProvider

  Le nom du fournisseur d'actions qui peut mettre en œuvre une ou plusieurs actions configurables. Les fournisseurs d'actions disponibles à l'installation sont :

  GenericAction

  Exécute les commandes non Tivoli Workload Scheduler. Les commandes sont exécutées sur le même ordinateur que celui où s'exécute le processeur d'événements.

  MailSender

  Connecte à un serveur de protocole SMTP et envoie un courrier électronique.

  MessageLogger

  Journalise l'occurrence d'une situation dans une base de données d'audit interne.

  TECEventForwarder

  Transmet l'événement à un serveur TEC (Tivoli Enterprise Console) externe ou à toute autre application capable d'écouter les événements au format TEC.

  TWSAction

  Exécute l'une des commandes `conman` suivantes :

  – submit job (`sbj`)
  – submit job stream (`sbs`)
  – submit command (`sbd`)
  – reply to prompt (`reply`)

  TWSForZosAction

  Ajoute une occurrence d'application (flot de travaux) au plan actuel sur IBM Tivoli Workload Scheduler for z/OS. Ce fournisseur est à utiliser dans des configurations de planification de bout en bout Tivoli Workload Scheduler.

  La description d'application de l'occurrence à ajouter doit exister dans la base de données AD de IBM Tivoli Workload Scheduler for z/OS.

  actionType

  Indique le type d'action qui doit être déclenché lorsqu'un événement spécifié est détecté. Chaque action peut être associée à un fournisseur d'action. Le tableau suivant répertorie les types d'action par fournisseur d'actions. Pour afficher les propriétés de chaque type d'événement, accédez à l'Annexe A, «Automatisation de la charge de travail commandée par les événements, définitions des événements et des actions», à la page 707.

Tableau 53. Types d'action par fournisseur d'actions.

Fournisseur d'actions	Types d'action
GenericAction	RunCommand
MailSender	SendMail
MessageLogger	PostOperatorMessage
TECEventForwarder	TECFWD

Tableau 53. Types d'action par fournisseur d'actions. (suite)

Fournisseur d'actions	Types d'action
TWSAction	reply (ReplyPrompt)
	sbd (SubmitAdHocJob)
	sbj (SubmitJob)
	sbs (SubmitJobStream)
TWSForZosAction	AddJobStream

responseType
Indique le moment où l'action doit être exécutée. Les valeurs peuvent être :
onDetection
L'action démarre dès que tous les événements définis dans la règle ont été détectés. S'applique à tous les types de règle. Voir aussi «Remarques sur le fonctionnement des règles», à la page 144.
onTimeOut
L'action démarre lorsque le temps spécifié dans `timeInterval` a expiré mais que tous les événements définis dans la règle n'ont pas été reçus. S'applique uniquement aux règles `set` et `sequence`.

Notez que les actions de dépassement de délai ne sont pas exécutées si vous ne spécifiez pas un intervalle de temps. Le planificateur vous permettra cependant d'enregistrer des règles d'événement pour lesquelles des actions de dépassement de délai ont été définies sans spécifier d'intervalle de temps parce que vous pouvez définir l'intervalle de temps ultérieurement. Jusqu'à ce point, seules les actions ayant le type de réponse `onDetection` sont traitées.

Les actions de dépassement de délai d'attente pour lesquelles un intervalle de temps n'a pas été défini sont exécutées uniquement lorsque les règles sont désactivées. Une règle d'événement est désactivée dans l'un des deux cas ci-après :
– La commande `planman deploy -scratch` est exécutée
– La règle est modifiée (elle est alors désactivée dès l'exécution de la commande `planman deploy`)

Dans tous les cas, la règle est en premier lieu désactivée, puis réactivée. A cet instant, toutes les actions en attente sont exécutées.

- Inclut les attributs facultatifs suivants :
description
Une description de l'action. Il peut contenir jusqu'à 120 caractères.

Vous devez écrire tout caractère spécial XML que vous pourriez utiliser selon la notation XML spéciale, tel que :
– `&` pour &
– `>` pour >
– `<` pour <
– `&quot;` pour "

etc.

scope Un ou plusieurs attributs de qualification qui décrivent l'action. Il peut contenir jusqu'à 120 caractères. La portée est automatiquement générée à partir de ce qui est défini dans le fichier XML. Elle ne peut pas être définie par les utilisateurs.

- Inclut une liste d'un ou plusieurs paramètres ou noms de propriété. Tous les types d'action ont au moins un paramètre obligatoire. Chaque paramètre est défini par :

  parameter name=" "
  Voir «Fournisseurs d'actions et définitions», à la page 719 pour une liste des paramètres ou noms de propriété disponibles pour chaque type d'action.

  value Voir «Fournisseurs d'actions et définitions», à la page 719 pour une liste des valeurs possibles ou types de valeurs.

  Vous pouvez utiliser la substitution de variable. Lorsque vous définissez des paramètres d'action, vous pouvez utiliser les noms de propriété des événements qui déclenchent la règle d'événement pour remplacer la valeur du nom de propriété de l'action. Pour cela, inscrivez la valeur du paramètre d'action que vous souhaitez substituer, de l'une des manières suivantes :

  – `${event.property}`

    Remplace la valeur telle quelle. Cette option est utile pour transmettre les informations à une action liée à celles-ci par voie de programme, par exemple l'option `schedTime` d'un flot de travaux.

  – `%{événement.propriété}`

    Remplace la valeur formatée en syntaxe lisible. Cette option est utile pour transmettre les informations à une action chargée de faire suivre les informations à un utilisateur, par exemple, pour formater le `schedTime` d'un flot de travaux dans le corps d'un courrier électronique.

  Où :

  event est le nom défini pour le déclenchement `eventCondition`.

  property
  est le nom de `attributeFilter` dans le prédicat de filtrage de la condition d'événements déclenchante. La valeur prise par le filtre d'attribut lorsque la règle est déclenchée est remplacée en tant que valeur de paramètre dans la définition d'action avant l'exécution.

  Il est à noter que vous pouvez également recourir à une substitution de variable si aucune valeur de `attributeFilter` n'a été spécifiée pour un attribut, ainsi que lorsque l'attribut est accessible en lecture seule.

Par exemple, la tâche d'une règle d'événement doit détecter le moment où l'un des travaux dont le nom commence par `job15` se termine par une erreur, et lorsque ceci se produit, doit soumettre à nouveau ce travail. La section `eventCondition` de la règle est codée comme suit :

```
<eventCondition
      name="event1"
      eventProvider="TWSObjectsMonitor"
      eventType="JobStatusChanged">
  <filteringPredicate>
     <attributeFilter
           name="JobName"
           operator="eq">
```

```
        <value>job15*</value>
      </attributeFilter>
      <attributeFilter
            name="Workstation"
            operator="eq">
        <value>*</value>
      </attributeFilter>
      <attributeFilter
            name="Status"
            operator="eq">
        <value>Error</value>
      </attributeFilter>
    </filteringPredicate>
  </eventCondition>
```

Des caractères génériques (* pour plusieurs caractères ou ? pour un seul caractère) sont utilisés pour généraliser les conditions d'événement souhaitées pour toutes les instances de travail dont le nom commence par `job15` et pour leur poste de travail associé. La substitution de variable est utilisée dans la section `action` pour soumettre à nouveau le travail spécifique qui s'est terminé par une erreur sur le même poste de travail. La section `action` est :

```
<action
      actionProvider="TWSAction"
      actionType="sbj"
      responseType="onDetection">
  <description>Soumet de nouveau le travail qui s'est terminé par une erreur</descripti
      <parameter name="JobDefinitionName">
        <value>${event1.JobName}</value>
      </parameter>
      <parameter name="JobDefinitionWorkstationName">
        <value>${event1.Workstation}</value>
      </parameter>
</action>
```

Exemples

`JOB7` a une dépendance de fichier sur `DAILYOPS.XLS`. Dès que le fichier est reçu, `JOB7` doit commencer à le traiter. La règle suivante contrôle que `JOB7` commence dans la minute qui suit le transfert de `DAILYOPS.XLS`. Sinon, un courrier électronique est envoyé à l'opérateur d'événement. Ceci est réalisé en définissant deux conditions d'événement séquentielles qui doivent être surveillées :

1. Le premier événement qui déclenche la règle est la création du fichier `DAILYOPS.XLS` sur le poste de travail vers lequel il doit être transféré. Dès que cet événement est détecté, une instance de règle est créée et un décompte d'une minute est démarré pour détecter la condition d'événement suivante.
2. Le second événement est la soumission de `JOB7`. Si cet événement n'est pas détecté dans l'intervalle de temps spécifié, la règle expire et l'action `SendMail` est démarrée.

```
<?xml version="1.0"?>
<eventRuleSet
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://www.ibm.com/xmlns/prod/tws/1.0/event-management/rules"
    xsi:schemaLocation="http://www.ibm.com/xmlns/prod/tws/1.0/event-management/rules
    EventRules.xsd">
  <eventRule
      name="sample_rule"
      ruleType="sequence"
      isDraft="no">
    <description>An email is sent if job JOB7 does not start within
                une minute après que le fichier DAILYOPS.XLS a été créé</description>
    <timeZone>America/Indianapolis</timeZone>
```

```
<validity
      from="2007-01-01"
      to="2007-12-31" />
<activeTime
      start="20:00:00"
      end="22:00:00" />
<timeInterval
      amount="60"
      unit="seconds" />
<eventCondition
      name="DAILYOPS_FTPed_event"
      eventProvider="FileMonitor"
      eventType="FileCreated">
   <filteringPredicate>
      <attributeFilter
            name="FileName"
            operator="eq">
         <value>c:/dailybus/DAILYOPS.XLS</value>
      </attributeFilter>
      <attributeFilter
            name="Workstation"
            operator="eq">
         <value>ACCREC03</value>
      </attributeFilter>
      <attributeFilter
            name="SampleInterval"
            operator="eq">
         <value>300</value>
      </attributeFilter>
   </filteringPredicate>
</eventCondition>
<eventCondition
      name="job_JOB7_problem_event"
      eventProvider="TWSObjectsMonitor"
      eventType="JobSubmit">
  <filteringPredicate>
     <attributeFilter
           name="JobStreamWorkstation"
           operator="eq">
        <value>ACCREC03</value>
     </attributeFilter>
     <attributeFilter
           name="Workstation"
           operator="eq">
        <value>ACCREC03</value>
     </attributeFilter>
     <attributeFilter
           name="JobStreamName"
           operator="eq">
        <value>JSDAILY</value>
     </attributeFilter>
     <attributeFilter
           name="JobName"
           operator="eq">
        <value>JOB7</value>
     </attributeFilter>
  </filteringPredicate>
</eventCondition>
<action
      actionProvider="MailSender"
      actionType="SendMail"
      responseType="onTimeOut">
   <description>Envoi d'un courrier électronique à l'opérateur du soir pour signaler que le travail n
               démarré</description>
   <parameter name="To">
      <value>eveoper@bigcorp.com</value>
   </parameter>
   <parameter name="Subject">
      <value>Job JOB7 failed to start within scheduled time on
             workstation ACCREC03.</value>
```

```
        </parameter>
      </action>
   </eventRule>
</eventRuleSet>
```

Il est à noter que la portée n'est pas indiquée lors de la définition initiale de la règle.

Pour d'autres exemples de règles d'événement, voir «Exemples de règles d'événement», à la page 138.

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle décrite dans :

le manuel Dynamic Workload Console - Guide d'utilisation, section Création d'une règle d'événement.

Définition des applications de charge de travail

Vous pouvez utiliser une applications de charge de travail pour normaliser une solution d'automatisation de charge de travail de sorte que la solution puisse être réutilisée dans un ou plusieurs environnements Tivoli Workload Scheduler automatisant de ce fait des processus métier.

Les applications de charge de travail sont définies dans un environnement Tivoli Workload Scheduler, également appelé environnement source, à l'aide de l'interface graphique du Concepteur de charge de travail accessible à partir de Dynamic Workload Console. Vous pouvez créer un modèle d'application de charge de travail, puis lui ajouter un ou plusieurs flots de travaux. Pour reproduire la solution d'automatisation de charge de travail dans un autre environnement Tivoli Workload Scheduler, le modèle d'application de charge de travail est exporté, puis après l'avoir personnalisé manuellement, il peut être importé dans l'environnement cible.

Le processus d'exportation génère un fichier compressé contenant tous les fichiers et informations requises pour exécuter la charge de travail dans un autre environnement Tivoli Workload Scheduler. Le fichier compressé contient :

Un fichier de définitions
Fichier XML, *nom du modèle d'application de charge de travail*_Definitions.UTF8.xml, qui contient les définitions de tous les objets exportés. Ces définitions seront déployées dans l'environnement cible de sorte à remplir la base de données cible avec les mêmes objets qui existent dans l'environnement source.

Fichier de mappage
Fichier de mappage, *nom du modèle d'application de charge de travail*_Mapping.UTF8.properties, requis pour traiter les objets qui sont dépendants de la topologie de l'environnement et qui ne peuvent pas être reproduits sans avoir été personnalisés manuellement. L'utilisateur cible modifie le fichier en remplaçant les noms des objets dans l'environnement source par les noms que ces objets portent dans l'environnement cible.

Un fichier d'informations de référence
Fichier d'informations de référence, *nom du modèle d'application de charge de travail*_SourceEnv_reference.txt, contenant les définitions des postes de

travail utilisées dans le modèle d'application de charge de travail et d'autres informations pouvant être utiles pour mapper correctement l'environnement source dans l'environnement cible afin d'exécuter l'application de charge de travail.

Les applications de charge de travail sont gérées à l'aide de l'interface utilisateur de ligne de commande **wappman**. Pour importer une application de charge de travail dans un environnement cible, le fichier de mappage personnalisé et le fichier de définitions sont transmis à la commande wappman en tant qu'entrée de sorte que tous les objets soient créés automatiquement dans la base de données Tivoli Workload Scheduler de l'environnement cible. Voir «Commande wappman», à la page 369 pour obtenir la syntaxe de commande complète et son utilisation.

Création d'un modèle d'application de charge de travail

Comment définir un modèle application de charge de travail à l'aide de Dynamic Workload Console.

Pour s'assurer que la solution d'automatisation de charge de travail puisse être facilement reproduite dans un autre environnement, il existe des valeurs recommandées à prendre en compte lors de la création du modèle application de charge de travail :

Définitions de travaux
Les travaux qui se rapportent aux éléments qui dépendent de l'environnement ou de la topologie dans lesquels ils se trouvent, comme les travaux de service Web, les travaux de transfert de fichier et les travaux de base de données, entre autres, devraient utiliser des variables lorsqu'ils indiquent ces éléments, tels que des accréditations, des chemins d'accès, et des numéros de port. Les variables peuvent être gérées dans le fichier de mappage de sorte que les valeurs correctes puissent être affectées à la variable.

Noms des postes de travail
Quand des travaux et des flots de travaux sont extraits de application de charge de travail pendant l'exportation, les noms des postes de travail sont extraits lorsqu'ils sont détectés dans l'environnement source. Les noms significatifs ou une convention de dénomination normalisée peuvent simplifier le procédé de mappage.

Utilisateurs
Les utilisateurs sont également extraits lorsqu'ils sont détectés dans l'environnement source. Si le même utilisateur n'est pas présent dans l'environnement source et cible, des variables devraient être utilisées pour indiquer l'utilisateur.

Fichier de mappage
Le fichier de mappage devrait être mis à jour après le processus d'importation. Dans ce cas, il peut être utile de remplacer une application application de charge de travail ou de la mettre à jour en apportant les modifications nécessaires au fichier de mappage.

Table de variable du flot de travaux
Toutes les variables utilisées pour représenter génériquement des objets dans application de charge de travail devraient être ajoutées à une table de variable spécifique en relation au flot de travaux dans application de charge de travail. Ceci permet à la personnalisation du flot de travaux de refléter l'environnement cible via le fichier de mappage. Evitez d'associer la table de variable par défaut à un flot de travaux. La table de variable par

défaut est extraite comme n'importe quelle autre table et devra être renommée, sinon le processus d'importation échouera car une table du même nom existe déjà. L'environnement cible a déjà une table de variable par défaut, MAIN_TABLE, définie.

Table de variables du cycle d'exécution
Toutes les variables utilisées pour représenter génériquement des objets dans application de charge de travail devraient être ajoutées à une table de variable spécifique en relation au cycle d'exécution dans application de charge de travail. Ceci permet à la personnalisation du cycle d'exécution de refléter l'environnement cible via le fichier de mappage. Evitez d'associer la table de variable par défaut à un cycle d'exécution. La table de variable par défaut est extraite comme n'importe quelle autre table et devra être renommée, sinon le processus d'importation échouera car une table du même nom existe déjà. L'environnement cible a déjà une table de variable par défaut, MAIN_TABLE, définie.

Dans le Concepteur de charge de travail, vous pouvez créer le modèle d'une charge de travail qui peut ensuite être importée et exécutée dans un autre environnement. Vous pouvez créer un modèle de application de charge de travail contenant un ou plusieurs flots de travaux avec tous les travaux connexes et les dépendances internes ou externes (telles que des fichiers, des ressources, des invites) afin d'avoir un flux autonome. Vous pouvez ensuite exporter le modèle de application de charge de travail pour le déployer et l'exécuter dans un autre environnement. Pour créer un modèle de application de charge de travail, procédez comme suit :

1. A partir de la barre d'outils de navigation, cliquez sur **Administration** > **Conception de la charge de travail** > **Gestion des définitions de charge de travail**
2. Indiquez le nom d'un moteur réparti. Concepteur de charge de travail s'ouvre.
3. Dans le panneau Liste de travail, sélectionnez **Nouveau** > **Modèle d'application de charge de travail**. Le modèle de application de charge de travail est créé dans la vue de détails et sa page de propriétés s'affiche.
4. Dans le panneau des propriétés, spécifiez les attributs pour le modèle de application de charge de travail que vous créez :

   Nom Zone obligatoire qui contient le nom du modèle de application de charge de travail. La longueur est limitée à 80 caractères.

   Description
   Texte descriptif facultatif pour aider les utilisateurs de application de charge de travail à comprendre l'objectif et les caractéristiques de application de charge de travail. La longueur est limitée à 120 caractères.

   Fournisseur
   Zone facultative qui indique le créateur du modèle de application de charge de travail. Elle peut être utile pour faire savoir aux utilisateurs de application de charge de travail qui a créé le modèle et qui l'a fourni. La longueur est limitée à 120 caractères.
5. Dans la vue Détails, cliquez avec le bouton droit de la souris sur le modèle de application de charge de travail, puis cliquez sur **Ajouter un flot de travaux** pour ajouter des flots de travaux à celui-ci.

6. Dans la boîte de dialogue de recherche, sélectionnez les flots de travaux que vous souhaitez ajouter. En même temps que les flots de travaux, les dépendances correspondantes sont également automatiquement ajoutées au modèle de application de charge de travail.
7. Cliquez sur **Sauvegarder** pour sauvegarder le modèle de application de charge de travail dans la base de données.
8. Cliquez avec le bouton droit de la souris le modèle de application de charge de travail et cliquez sur **Exporter** pour produire un fichier compressé, nommé ***nom de modèle application de charge de travail*.zip**, contenant tous les fichiers et informations requises pour permettre à la charge de travail de s'exécuter également dans un autre environnement.

Le fichier compressé contient :

*Nom de modèle application de charge de travail*_**Definitions.UTF8.xml**
: Fichier XML qui contient les définitions de tous les objets exportés. Ces définitions seront déployées dans l'environnement cible pour remplir la base de données cible avec les mêmes objets qui existent dans l'environnement source. Les objets dans le fichier de définition peuvent rester tel quel ou vous pouvez les renommer. Si un objet n'a pas de définition dans le fichier de définition, par exemple, un poste de travail, alors, au moment de l'importation, l'objet correspondant ne sera pas créé dans l'environnement cible. Cet objet doit déjà être présent dans l'environnement cible, donc, pour ces types d'objets, vous devez les mapper dans le fichier de mappage.

*Nom de modèle application de charge de travail*_**Mapping.UTF8.properties**
: Fichier de mappage utilisé pour remplacer les noms des objets dans l'environnement source avec les noms de ces objets dans l'environnement cible. Les objets qui seront créés dans l'environnement cible peuvent être créés avec les mêmes noms que ceux de l'environnement source ou vous pouvez spécifier un autre nom dans ce fichier.

*Nom de modèle application de charge de travail*_**SourceEnv_reference.txt**
: Informations de référence contenant les définitions des postes de travail utilisés dans le modèle de application de charge de travail et toutes les autres informations qui peuvent être utiles pour mapper correctement l'environnement source dans l'environnement cible afin de permettre l'exécution de application de charge de travail.

Le package compressé doit ensuite être importé dans l'environnement cible où application de charge de travail sera déployé, ce qui créé tous les objets requis dans l'environnement cible. Dans l'environnement cible, le fichier *nom application de charge de travail*_Mapping.UTF8.properties doit être édité manuellement en indiquant les noms des objets tels qu'ils sont définis dans l'environnement cible (par exemple, les noms des postes de travail sur lesquels s'exécutent les flots de travaux). L'opération d'importation doit être effectuée dans l'environnement cible à l'aide de la ligne de commande. Pour plus de détails, voir les sections sur les applications de charge de travail et la commande wappman dans *Tivoli Workload Scheduler - Guide d'utilisation et de référence*.

Chapitre 9. Gestion des objets dans la base de données - composer

Le présent chapitre décrit comment utiliser le programme de ligne de commande **composer** afin de gérer les objets de planification dans la base de données Tivoli Workload Scheduler. Il comprend les sections suivantes :

- «Configuration du programme de ligne de commande composer»
- «Exécution de commandes depuis composer», à la page 305
- «Commandes Composer», à la page 310

Configuration du programme de ligne de commande composer

Le programme de ligne de commande **composer** gère les objets de planification dans la base de données.

Vous devez installer la fonction de client de ligne de commande de Tivoli Workload Scheduler sur des agents et sur des systèmes tolérants aux pannes à l'extérieur du réseau Tivoli Workload Scheduler pour pouvoir utiliser le programme de ligne de commande **composer**.

Configuration de l'environnement composer

La présente section décrit la configuration de votre environnement composer.

Sortie sur le terminal

Les variables de shell nommées ***MAESTROLINES*** et ***MAESTROCOLUMNS*** définissent la sortie de votre ordinateur. Si l'une ou l'autre de ces variables n'est pas définie, les variables de shell standard ***LINES*** et ***COLUMNS*** sont utilisées. A la fin de chaque page d'écran, le programme composer vous invite à poursuivre. Si une valeur nulle ou négative est attribuée à ***MAESTROLINES*** (ou ***LINES***), composer ne marque pas de pause à la fin de la page.

Selon la valeur attribuée à la variable locale ***MAESTROCOLUMNS***, deux ensembles d'informations s'affichent concernant l'objet sélectionné. Il existe deux possibilités :

- MAESTROCOLUMNS < 120 caractères
- MAESTROCOLUMNS >= 120 caractères

La valeur définie dans la variable locale ***MAESTROCOLUMNS*** ne peut pas être supérieure à 1024.

Voir tableau 60, à la page 327 et tableau 61, à la page 339 pour en savoir plus sur les différents formats de sortie.

Sortie offline

Dans les commandes du programme composer, l'option **;offline** permet d'imprimer la sortie d'une commande. Si vous la définissez, les variables suivantes contrôlent la sortie :

Variables Windows :

MAESTROLP
Indique le fichier dans lequel la sortie d'une commande est écrite. La valeur par défaut est **stdout**.

MAESTROLPLINES
Indique le nombre de lignes par page. La valeur par défaut est 60.

MAESTROLPCOLUMNS
Indique le nombre de caractères par ligne. La valeur par défaut est 132.

Variables UNIX :
Dans les commandes du programme composer, l'option **;offline** permet d'imprimer la sortie d'une commande. Lorsqu'elle est insérée, la sortie est contrôlée par les variables de shell ci-après :

MAESTROLP
Indique la destination de la sortie d'une commande. Vous pouvez indiquer :

> *fichier*
La sortie est redirigée vers un fichier dont le contenu est remplacé. Si le fichier n'existe pas, il est créé.

>> *fichier*
La sortie est redirigée vers un fichier et ajoutée à la fin de celui-ci. Si le fichier n'existe pas, il est créé.

| *commande*
La sortie est dirigée vers une commande système ou un processus. Le programme exécute la commande système, que la sortie soit générée ou non.

|| *commande*
La sortie est dirigée vers une commande système ou un processus. La commande système n'est pas exécutée en l'absence de résultat.

La valeur par défaut de ***MAESTROLP*** est **| lp -tCONLIST** qui dirige la sortie de la commande vers l'imprimante et insère le titre “CONLIST” sur la page bannière de la sortie imprimée.

MAESTROLPLINES
Indique le nombre de lignes par page. La valeur par défaut est **60**.

MAESTROLPCOLUMNS
Indique le nombre de caractères par ligne. La valeur par défaut est **132**.

Vous devez exporter les variables avant d'exécuter le programme **composer**.

Editeur Composer

Plusieurs commandes du programme composer permettent d'ouvrir automatiquement un éditeur de texte. Vous pouvez sélectionner l'éditeur que composer doit utiliser.

De plus, aussi bien sous Windows que sous UNIX, vous pouvez définir la variable d'environnement `XMLEDIT` pour pointer vers un éditeur XML de votre choix afin d'éditer les définitions de règles d'événement. L'éditeur XML s'ouvre automatiquement à chaque exécution des commandes `composer add`, `new` ou `modify` sur une règle d'événement.

Windows :

Sous Windows, **Notepad** est l'éditeur utilisé par défaut. Pour changer d'éditeur, définissez la variable d'environnement *EDITOR* avec le chemin d'accès et le nom du nouvel éditeur avant d'exécuter **composer**.

UNIX :
Plusieurs des commandes que vous pouvez émettre à partir de **composer** ouvrent automatiquement un éditeur de texte. Le type d'éditeur utilisé est déterminé par la valeur de deux variables du shell. Si la variable ***VISUAL*** est définie, elle définit l'éditeur ; sinon, c'est la variable ***EDITOR*** qui définit l'éditeur. Si aucune de ces variables n'est définie, l'éditeur **vi** apparaît.

Sélection de l'invite de commande sous UNIX

L'invite de commande **composer** est définie dans le fichier *racine_TWS*/`localopts`. L'invite de commande par défaut correspond à un tiret (-). Pour sélectionner une autre invite, éditez l'option ***composer prompt*** dans le fichier `localopts` et modifiez le tiret. L'invite peut comporter jusqu'à dix caractères, le signe dièse (#) de fin requis n'étant pas inclus :

```
#--------------------------------------------------------------------------
# Custom format attributes
#
date format =        1     # The possible values are 0-ymd, 1-mdy,
2-dmy, 3-NLS.
composer prompt =      -
conman prompt =       %
switch sym prompt =    <n>%
#--------------------------------------------------------------------------
```

Pour plus d'informations sur le fichier de configuration `localopts`, voir *IBM Tivoli Workload Scheduler - Guide d'administration*.

Exécution du programme Composer

Pour configurer l'environnement afin d'utiliser **composer**, définissez les variables *PATH* et *TISDIR_TWS* en exécutant l'un des scripts suivants :

Sous UNIX :

- `. ./`*racine_TWS*`/`**`tws_env.sh`** pour les interpréteurs de commandes Bourne et Korn
- `. ./`*racine_TWS*`/`**`tws_env.csh`** pour les interpréteurs de commandes C

Sous Windows :

- *racine_TWS*`\`**`tws_env.cmd`**

Utilisez ensuite la syntaxe suivante pour exécuter les commandes à partir de l'interface utilisateur **composer** :

composer [**-file** *filename*][***connection_parameters***] ["*command*[**&**[*command*]][...]"]

où :

-file *file_name*
: Indique un autre fichier de propriétés personnalisé contenant la configuration des paramètres de connexion, utilisé à la place des fichiers **`useropts`** et **`localopts`**.

paramètres_connexion
: Si vous utilisez **composer** à partir du gestionnaire de domaine maître, les

paramètres de connexion ont été configurés lors de l'installation et n'ont pas besoin d'être fournis, sauf si vous ne voulez pas utiliser les valeurs par défaut.

Si vous utilisez **composer** à partir du client de ligne de commande sur un autre poste de travail, les paramètres de connexion peuvent être fournis par l'une ou plusieurs de ces méthodes :

- Ils sont stockés dans le fichier `localopts`
- Ils sont stockés dans le fichier `useropts`
- Ils sont fournis à la commande dans un fichier de paramètres
- Ils sont fournis à la commande au sein d'une chaîne de commande

Pour une présentation de ces options, voir «Configuration des options pour l'utilisation des interfaces utilisateur», à la page 59. Pour des détails complets sur les paramètres de configuration, voir la rubrique relative à la configuration de l'accès du client de ligne de commande dans *Tivoli Workload Scheduler - Guide d'administration*.

Le programme de ligne de commande **composer** est automatiquement installé lors de l'installation du gestionnaire de domaine maître. Il doit être installé séparément sur un poste de travail d'agent Tivoli Workload Scheduler ou de façon autonome sur un noeud en dehors du réseau Tivoli Workload Scheduler. La fonction qui installe le programme de ligne de commande **composer** est nommée *client de ligne de commande*. Pour plus d'informations sur l'installation de la fonction *Command Line Client*, voir *IBM Tivoli Workload Scheduler - Guide de planification et d'installation*.

Une fois l'installation effectuée, vous pouvez utiliser la ligne de commande **composer** en mode de *traitement par lots* et *interactif*.

Lorsque vous exécutez **composer** en mode *interactif*, vous lancez d'abord le programme de ligne de commande **composer**, puis vous exécutez les commandes l'une après l'autre à partir de l'invite de ligne de commande **composer**, par exemple :

```
composer –username admin2 –password admin2pwd
   add myjobs.txt
   create myjobs.txt from jobs=@
```

Lorsque vous exécutez **composer** en mode de *traitement par lots*, vous lancez le programme de ligne de commande **composer** en spécifiant comme paramètre d'entrée la commande à émettre. Lorsqu'elle est exécutée, le programme de ligne de commande **composer** s'arrête comme dans l'exemple suivant :

```
composer –f "c:\TWS\network\mylocalopts" add myjobs.txt
```

Remarque : Si vous utilisez le mode de traitement par lots pour exécuter plusieurs commandes à partir de **composer**, assurez-vous que vous gérez le caractère ; de l'une des façons suivantes :

- Utilisation des guillemets, par exemple :

  ```
  composer "delete dom=ancien_domaine; noask"
  ```

- Utilisation d'un espace, par exemple :

  ```
  composer delete dom=ancien_domaine noask
  ```

- Utilisation d'une barre oblique inversée (\), puis d'un point-virgule (;) par exemple :

  ```
  composer delete dom=ancien_domaine \; noask
  ```

Voici d'autres exemples de l'utilisation de la commande avec les paramètres de connexion définis dans les scripts de la configuration locale :

- Exécute les commandes **print** et **version** avant de quitter :

  ```
  composer "p parms&v"
  ```

- Exécute les commandes **print** et **version** avant d'inviter l'utilisateur à taper une commande :

  ```
  composer "p parms&v&"
  ```

- Lit les commandes à partir de `cmdfile` :

  ```
  composer < cmdfile
  ```

- Transmet les commandes de `cmdfile` au programme **composer**:

  ```
  cat cmdfile | composer
  ```

Remarque : Sur les postes de travail Windows, si le contrôle de compte utilisateur (UAC) est activé et que la liste d'exceptions UAC ne contient pas le fichier `cmd.exe`, vous devez ouvrir l'interpréteur de commandes DOS avec l'option `"Exécuter en tant qu'administrateur"` pour exécuter **composer** sur votre poste de travail en tant qu'utilisateur générique différent de l'administrateur ou utilisateur Tivoli Workload Scheduler.

Caractères de commande

Vous pouvez taper les caractères de commande suivants en mode conversationnel pour interrompre le programme **Composer**, si les paramètres ***stty*** sont configurés à cette fin.

Ctrl+c Le programme **composer** interrompt l'exécution de la commande dès que l'étape courante le permet et renvoie une invite de commande.

Ctrl+d Le programme **composer** s'arrête après avoir exécuté la commande courante.

Exécution de commandes depuis composer

Les commandes du programme **composer** se décomposent de la manière suivante :

nom_commande sélection arguments

où :

nom_commande
Indique le nom de la commande.

sélection
Indique l'objet ou le groupe d'objets sur lequel sera exécutée la commande.

arguments
Indique les arguments de la commande.

Filtres et caractères génériques

Dans le programme **composer** de Tivoli Workload Scheduler, vous pouvez utiliser des caractères génériques et des filtres lorsque vous exécutez certaines commandes spécifiques afin de filtrer les objets de planification définis dans la base de données. vous pouvez utiliser les caractères génériques suivants à partir de **composer** :

@ Remplace un ou plusieurs caractères alphanumériques.

? Remplace un caractère alphanumérique.

Pour rechercher les occurrences de noms contenant soit @ soit ?, utilisez la barre oblique inversée \ devant @ ou ? de façon à ce qu'ils ne soient pas interprétés comme des caractères génériques. De la même façon, la barre oblique inversée doit être précédée d'une autre barre oblique inversée pour être interprétée comme une occurrence à trouver. Les exemples suivants expliquent ces règles qui s'appliquent également lorsque vous spécifiez des chaînes de recherche à l'aide du mot clé **filter**.

S@E Recherche de toutes les chaînes commençant par S et se terminant par E, quelle que soit leur longueur.

S?E Recherche de toutes les chaînes commençant par S, se terminant par E et dont la longueur est de trois caractères.

S\@E Recherche de la chaîne S@E.

S\?E Recherche de la chaîne S?E.

S\\E Recherche de la chaîne S\E.

Les commandes que vous pouvez exécuter à partir de composer et qui prennent en charge le filtrage sont :

- display
- create
- delete
- list
- lock
- modify
- print
- unlock

La syntaxe utilisée pour filtrer les objets lorsque vous émettez l'une de ces commandes est la suivante :

nom_commande type_objet=selection; [*option;*] [**filter** *mot_clé_filtre*=selection [...]]

Le tableau 54, à la page 307 montre les objets de planification que vous pouvez filtrer lorsque vous émettez les commandes répertoriées ci-dessus et, pour chaque objet, les zones qui peuvent être filtrées (en *italique*) ou la clé (en **gras)** utilisée pour filtrer les zones :

Tableau 54. Critères de filtrage des objets de planification

Objet de planification	**Mots clés de filtre ou zones qui peuvent être filtrées**	**Description**	**Exemple**
poste de travail	*nom_poste_travail*	Applique la commande aux postes de travail dont le nom respecte les critères.	list ws=p@
	domaine	Applique la commande aux postes de travail qui appartiennent à un domaine.	mod ws=@; filter **domain**=dom1
	vartable	Applique la commande aux postes de travail qui font référence à la table de variables spécifiée.	mod ws=@; filter **vartable**=table2
domain	*nom_domaine*	Applique la commande aux domaines dont le nom respecte les critères.	display dom=dom?
	parent	Applique la commande aux domaines dont le domaine parent respecte les critères.	list dom=@; filter **parent**=rome
prompt	*nom_invite*	Applique la commande aux invites globales dont le nom respecte les critères.	lock prompt=p@
user	*nom_poste_travail*# *nom_utilisateur*	Applique la commande aux utilisateurs dont l'identificateur respecte les critères.	list users=cpu1#operator?
resource	*nom_poste_travail*# *nom_ressource*	Applique la commande aux ressources dont l'identificateur respecte les critères.	print res=cpu?#operator?
Variable	*nom_variable*	Applique la commande aux paramètres dont le nom respecte les critères.	delete vb=mytable.myparm@
définition de travail	*nom_travail*	Applique la commande aux définitions de travail dont le nom respecte les critères.	mod jd=mycpu#myjob@
	RecoveryJob	Applique la commande aux travaux dont la définition contient la définition de travail de reprise spécifiée.	list jobdefinition=@; filter **RecoveryJob**=CPUA#job01

Tableau 54. Critères de filtrage des objets de planification (suite)

Objet de planification	Mots clés de filtre ou zones qui peuvent être filtrées	Description	Exemple
flot de travaux	*nom_poste_travail#nom_flot_travaux*	Applique la commande aux définitions de travail dont le nom respecte les critères.	mod js=mycpu#myjs@
	Calendar ou **FreeCalendar**	Applique la commande aux flots de travaux contenant l'agenda ou l'agenda des jours chômés spécifié dans le filtre.	list js=@#@; filter **Calendar**=cal1
	Jobdefinition	Applique la commande aux flots de travaux contenant la définition de travail spécifiée dans le filtre.	list js=@#@; filter **jobdefinition**=CPUA#job01
	Ressource	Applique la commande aux flots de travaux qui font référence à la ressource spécifiée dans le filtre.	list js=@#@; filter **Resource**=cpu1#disk1
	Prompt	Applique la commande aux flots de travaux qui font référence à l'invite spécifiée dans le filtre.	list js=@#@; filter **Prompt**=myprompt
	Vartable	Applique la commande aux flots de travaux qui font référence à la table de variables spécifiée dans le filtre. La table de variables peut être spécifiée dans la section du cycle d'exécution ou du le flot de travaux.	list js=@#@; filter **Vartable**=table1
	Rcvartable	Applique la commande aux cycles d'exécution des flots de travaux qui font référence à la table de variables spécifiée dans le filtre.	list js=@#@; filter **Rcvartable**=table1
	Jsvartable	Applique la commande aux flots de travaux qui font référence à la table de variables spécifiée dans le filtre, quels que soient les éléments spécifiés dans le cycle d'exécution.	list js=@#@; filter **Jsvartable**=table1
règle d'événement	*nom_règle_événement*	Applique la commande aux règles d'événement comprenant une action appliquée à un travail ou flot de travaux spécifique.	list er=@; filter js=accrecjs5

Tableau 54. Critères de filtrage des objets de planification (suite)

Objet de planification	Mots clés de filtre ou zones qui peuvent être filtrées	Description	Exemple
vartable	*nom_table_var*	Applique la commande aux tables de variables dont le nom satisfait les critères.	list vartable=A@
	isdefault	Applique la commande à la table de variables par défaut.	list vartable=A@; filter isdefault

Vous pouvez associer plusieurs filtres pour le même type d'objet (voir exemple suivant) :

```
list js=@#@; filter Calendar=cal1 FreeCalendar=VACATIONS jobdefinition=CPUA#job01
```

La commande génère une liste des flots de travaux utilisant l'agenda `cal1`, l'agenda des jours chômés `VACATIONS` et contenant un travail avec la définition `CPUA#job01`.

Délimiteurs et caractères spéciaux

Le tableau 55 répertorie les caractères qui ont une signification particulière dans les commandes composer.

Tableau 55. Délimiteurs et caractères spéciaux pour le programme composer

Caractère	Description
&	Délimiteur de commande. Voir «Exécution du programme Composer», à la page 303.
;	Délimiteur d'argument. Par exemple : `;info;offline`
=	Délimiteur de valeur. Par exemple : `sched=sked5`
: !	Préfixes de commande qui transmettent la commande au système. Ces préfixes sont facultatifs. Si le programme composer ne reconnaît pas une commande, elle est transmise automatiquement au système. Par exemple : `!ls ou :ls`
<< >>	Crochets de mise en commentaire. Des commentaires peuvent figurer sur une seule ligne n'importe où dans un flot de travaux. Par exemple : `schedule foo <<commentaire>> on everyday`

*	Préfixe de commentaire. Lorsque ce préfixe correspond au premier caractère d'une ligne, la ligne entière correspond à un commentaire. Lorsque ce préfixe suit une commande, seul le reste de la ligne correspond au commentaire. Par exemple :
	`*commentaire`
	ou
	`print& *commentaire`

>	La sortie de la commande est redirigée vers un fichier dont le contenu est remplacé. Si le fichier n'existe pas, il est créé. Par exemple : `display parms > parmlist`
>>	Dirige la sortie de la commande vers un fichier et celle-ci est ajoutée à la fin du fichier. Si le fichier n'existe pas, il est créé. Par exemple : `display parms >> parmlist`

Tableau 55. Délimiteurs et caractères spéciaux pour le programme composer (suite)

Caractère	Description
|	La sortie de la commande est dirigée vers une commande système ou un processus. Le programme exécute la commande système, que la sortie soit générée ou non. Par exemple : `display parms | grep alparm`
||	La sortie de la commande est dirigée vers une commande système ou un processus. La commande système n'est pas exécutée en l'absence de résultat. Par exemple : `display parms || grep alparm`

Codes retour de composer

Codes retour de composer

Gestion des codes retour de composer

Lorsque vous exécutez une commande composer, la ligne de commande peut afficher un code retour de résultat. Pour trouver le code retour, procédez comme suit :

Sur les systèmes Windows :

```
echo %ERRORLEVEL%
```

Sur les systèmes UNIX :

```
echo $?
```

La ligne de commande Composer renvoie les codes retour suivants :

0 Commande terminée.

4 La commande a été exécutée avec un avertissement.

8 La commande a été exécutée avec une erreur.

16 La commande échoue.

32 La commande comporte une erreur de syntaxe.

Commandes Composer

Le tableau 56, à la page 311 répertorie les commandes **composer**.

Remarque : Les noms de commande et les mots clés peuvent être entrés en majuscules ou en minuscules et peuvent être abrégés au nombre minimal de caractères suffisant pour les distinguer les uns des autres de façon unique. Certains noms de commande ont également une forme abrégée.

Cependant, certaines abréviations (par exemple **v** pour **version**) désignent une commande spécifique, même si elles n'identifient pas uniquement la commande dans la liste. Ce cas se produit lorsque l'abréviation est incorporée dans le produit, ce qui permet d'éviter l'appel d'une la commande incorrecte pouvant générer des problèmes de non-concordance.

Tableau 56. Liste des commandes composer

Commande	Nom abrégé	Description	Voir page
add	**a**	Ajoute une définition d'objets de planification à la base de données à partir d'un fichier texte.	«add», à la page 316
authenticate	**au**	Modifie les droits d'accès de l'utilisateur exécutant **composer**.	«authenticate», à la page 318
continue	**c**	Ignore l'erreur suivante.	«continue», à la page 319
create	**cr**	Extrait une définition d'objet à partir de la base de données et l'écrit dans un fichier texte. Synonyme de la commande **extract**.	«extract», à la page 330
delete	**de**	Supprime des objets de planification.	«delete», à la page 319
display	**di**	Affiche les détails de l'objet de planification spécifié.	«display», à la page 324
edit	**ed**	Permet d'éditer un fichier.	«edit», à la page 329
exit	**e**	Quitte le programme **composer**.	«exit», à la page 329
extract	**ext**	Extrait une définition d'objet à partir de la base de données et l'écrit dans un fichier texte.	«extract», à la page 330
help	**h**	Appelle l'aide en ligne d'une commande.	«help», à la page 334
list	**l**	Répertorie les objets de planification.	«list», à la page 335
lock	**lo**	Verrouille l'accès aux objets de la base de données.	«lock», à la page 341
modify	**m**	Modifie les objets de planification.	«modify», à la page 345
new		Crée un objet de planification à l'aide d'un fichier texte dans lequel la définition d'objet est insérée en ligne. Si vous indiquez le type d'objet de planification que vous souhaitez définir après la commande *new*, une définition d'objet prédéfinie est écrite dans le fichier texte.	«new», à la page 350
print	**p**	Imprime des objets de planification.	«display», à la page 324
redo	**red**	Modifie et exécute à nouveau la commande précédente.	«redo», à la page 352
rename	**rn**	Modifie le nom d'objet.	«rename», à la page 353
replace	**rep**	Permet de remplacer des objets de planification.	«replace», à la page 356
commande system		Appelle une commande de système d'exploitation.	«Commande système», à la page 357

Tableau 56. Liste des commandes composer (suite)

Commande	Nom abrégé	Description	Voir page
unlock	**u**	Déverrouille l'objet de planification défini dans la base de données.	«unlock», à la page 357
validate	**val**	Valide la syntaxe, la sémantique et l'intégrité des données d'une définition d'objet.	«validate», à la page 361
version	**v**	Affiche la bannière du programme de ligne de commande **composer**.	«version», à la page 362
wkldapp		Crée, remplace, supprime, affiche et répertorie une application de charge de travail.	«Commande wappman», à la page 369

Vérification de l'intégrité référentielle

Tivoli Workload Scheduler effectue automatiquement des vérifications référentielles pour préserver l'intégrité des définitions d'objet dans la base de données lorsque vous exécutez des commandes qui créent, modifient ou suppriment la définition d'un objet référencé. Le produit effectue les vérifications suivantes :

- Chaque fois que vous utilisez une commande qui crée un nouvel objet dans la base de données, Tivoli Workload Scheduler vérifie les points suivants :
  - Aucun objet du même type et avec le même identificateur n'existe déjà.
  - Les objets référencés par cet objet existent déjà dans la base de données.
- Chaque fois que vous exécutez une commande qui modifie une définition d'objet dans la base de données, Tivoli Workload Scheduler vérifie les points suivants :
  - La définition d'objet à modifier existe dans la base de données.
  - Les objets référencés par cet objet existent dans la base de données.
  - Pour éviter les incohérences dans l'intégrité, la définition d'objet ne s'affiche pas dans la définition d'un objet appartenant à la chaîne de ses ancêtres.
- Chaque fois que vous exécutez une commande qui supprime une définition d'objet dans la base de données, Tivoli Workload Scheduler vérifie les points suivants :
  - La définition d'objet à supprimer existe dans la base de données.
  - La définition d'objet à supprimer n'est pas référencée par d'autres objets définis dans la base de données.

Il est à noter que les règles d'événement ne font l'objet d'aucun contrôle d'intégrité référentielle.

Le tableau 57 montre, pour chaque type d'objet, les identificateurs utilisés pour identifier de manière unique l'objet dans la base de données lors de la création ou de la modification des définitions d'objet :

Tableau 57. Identificateurs d'objet pour chaque type d'objet défini dans la base de données

Type d'objet	Identificateurs d'objet
domaine	*nom_domaine*
poste de travail	*poste_de_travail* (vérifié sur les postes de travail et les classes de poste de travail)
classe de poste de travail	*nom_classe_poste_de_travail* (vérifié sur les postes de travail et les classes de postes de travail)

Tableau 57. Identificateurs d'objet pour chaque type d'objet défini dans la base de données (suite)

Type d'objet	Identificateurs d'objet
agenda	*nom_agenda*
définition de travail	*nom_poste_travail* et *nom_travail*
user	*nom_poste_travail* et *nom_utilisateur*
flot de travaux	*nom_poste_travail* et *nom_flot_travaux* et *début_validité* si ce paramètre est défini
travail dans un flot de travaux	*nom_poste_travail*, *nom_flot_travaux*, *nom_travail* et *début_validité* si ce paramètre est défini
ressource	*nom_poste_travail* et *nom_ressource*
invite	*nom_invite*
table de variables	*nom_table_variables*
Variable	*nom_table_variables.nom_variable*
Règle d'événement	*nom_règle_événement*

En règle générale, l'intégrité référentielle empêche la suppression des objets lorsqu'ils sont référencés par d'autres objets dans la base de données. Toutefois, dans certains cas où la suppression d'un objet (par exemple un poste de travail) implique uniquement la mise à jour d'un objet de référence (par exemple, une classe de postes de travail qui le comprend), la suppression peut être autorisée. Le tableau 58 montre tous les cas où un objet référencé peut être supprimé même si d'autres objets y font référence :

Tableau 58. Mise à jour de la définition d'objet lors de la suppression de l'objet référencé

Objet	Références	Lors de la suppression de l'objet référencé ...
Dépendance interréseau	Poste de travail	... suppression de la dépendance du travail ou du flot de travaux
Dépendance de prédécesseur/ successeur externe	Flot de travaux	... suppression de la dépendance du travail ou du flot de travaux
	Travail	... suppression de la dépendance du travail ou du flot de travaux
Dépendance interne	Travail	... suppression de la dépendance du travail ou du flot de travaux
Classe de poste de travail	Poste de travail	... suppression du poste de travail de la classe de postes de travail

Le tableau 59 décrit le comportement du produit lorsqu'il doit supprimer un objet désigné par un autre objet à l'aide d'une relation spécifique :

Tableau 59. Vérification de l'intégrité référentielle lors de la suppression d'un objet de la base de données

Objet à supprimer	Désigné par l'objet	Relation	Règle de suppression
Domaine A	Domaine B	Le domaine A est parent du domaine B	Un message d'erreur spécifiant la relation existante s'affiche.
	Poste de travail B	Le poste de travail B appartient au domaine A	Un message d'erreur spécifiant la relation existante s'affiche.

Tableau 59. Vérification de l'intégrité référentielle lors de la suppression d'un objet de la base de données (suite)

Objet à supprimer	Désigné par l'objet	Relation	Règle de suppression
Poste de travail A	Poste de travail B	Le poste de travail A est l'hôte du poste de travail B	Un message d'erreur spécifiant la relation existante s'affiche.
	Travail B	Le travail B est défini sur le poste de travail A	Un message d'erreur spécifiant la relation existante s'affiche.
	Flot de travaux B	Le flot de travaux B est défini sur le poste de travail A	Un message d'erreur spécifiant la relation existante s'affiche.
	Utilisateur B	L'utilisateur B est défini sur le poste de travail A	Un message d'erreur spécifiant la relation existante s'affiche.
	Flot de travaux B	Le poste de travail A fonctionne comme agent réseau pour les dépendances interréseaux définies dans le flot de travaux B	Le poste de travail A et la dépendance interréseau sont supprimés
	Flot de travaux B	Le flot de travaux B possède une dépendance de fichier à partir d'un fichier défini sur le poste de travail A	Le poste de travail A et la dépendance de fichier sont supprimés
	Travail B dans le flot de travaux B	Le poste de travail A fonctionne comme agent réseau pour les dépendances interréseaux définies dans le travail B	Le poste de travail A et la dépendance interréseau sont supprimés
	Travail B dans le flot de travaux B	Le travail B possède une dépendance de fichier à partir d'un fichier défini sur le poste de travail A	Le poste de travail A et la dépendance de fichier sont supprimés
	Ressource B	La ressource B est définie sur le poste de travail A	Un message d'erreur spécifiant la relation existante s'affiche.
	Fichier B	Le fichier B est défini sur le poste de travail A	Un message d'erreur spécifiant la relation existante s'affiche.
	Classe de postes de travail B	Le poste de travail A appartient à la classe de postes de travail B	Le poste de travail A et son entrée dans la classe de postes de travail B sont supprimés.
	Travail B dans le flot de travaux B	Le travail B contenu dans le flot de travaux B est défini sur le poste de travail A	Un message d'erreur spécifiant la relation existante s'affiche.

Tableau 59. Vérification de l'intégrité référentielle lors de la suppression d'un objet de la base de données (suite)

Objet à supprimer	Désigné par l'objet	Relation	Règle de suppression
Travail A	Travail B	Le travail A est un travail de reprise du travail B	Un message d'erreur spécifiant la relation existante s'affiche.
	Flot de travaux B	Le travail A est contenu dans le flot de travaux B	Un message d'erreur spécifiant la relation existante s'affiche.
	Flot de travaux B	Le flot de travaux B suit le travail A	Le travail A et la dépendance de prédécesseur/successeur dans le flot de travaux B sont supprimés.
	Travail B dans le flot de travaux B	Le flot de travaux B suit le travail A	Le travail A et la dépendance de prédécesseur/successeur dans le travail B sont supprimés.
	règle d'événement B	Le travail A est dans la définition d'action de la règle d'événement B (et ne fait appel à aucune substitution de variable)	Un message d'erreur spécifiant la relation existante s'affiche.
Agenda A	Flot de travaux B	Le flot de travaux B utilise l'agenda A	Un message d'erreur spécifiant la relation existante s'affiche.
Classe de postes de travail A	Travail B	Le travail B est défini sur la classe de postes de travail A	Un message d'erreur spécifiant la relation existante s'affiche.
	Flot de travaux B	Le flot de travaux B est défini sur la classe de postes de travail A	Un message d'erreur spécifiant la relation existante s'affiche.
	Ressource B	La ressource B est définie sur la classe de postes de travail A	Un message d'erreur spécifiant la relation existante s'affiche.
	Fichier B	Le fichier B est défini sur la classe de postes de travail A	Un message d'erreur spécifiant la relation existante s'affiche.
Ressource A	Flot de travaux B	Nécessite la dépendance définie dans le flot de travaux B	Un message d'erreur spécifiant la relation existante s'affiche.
	Travail B dans le flot de travaux B	Nécessite la dépendance définie dans le travail B	Un message d'erreur spécifiant la relation existante s'affiche.
Invite A	Flot de travaux B	Dépendance d'invite définie dans le flot de travaux B	Un message d'erreur spécifiant la relation existante s'affiche.
	Travail B dans le flot de travaux B	Dépendance d'invite définie dans le travail B	Un message d'erreur spécifiant la relation existante s'affiche.

Tableau 59. Vérification de l'intégrité référentielle lors de la suppression d'un objet de la base de données (suite)

Objet à supprimer	Désigné par l'objet	Relation	Règle de suppression
variable A	Flot de travaux B	variable A est utilisé dans le flot de travaux B dans : • le texte d'une invite ad hoc • ou le nom de fichier spécifié dans une dépendance de fichier	variable A est supprimé sans aucune vérification
	Travail B	variable A est utilisé dans le flot de travaux B dans : • le texte d'une invite ad hoc • ou le nom de fichier spécifié dans une dépendance de fichier • ou la valeur spécifiée pour connexion_flot • ou la valeur spécifiée pour nom_script	variable A est supprimé sans aucune vérification
	Invite B	variable A est utilisé dans le texte de l'invite B	variable A est supprimé sans aucune vérification
table de variables A	Flot de travaux B	table de variables A est référencé dans le flot de travaux B	table de variables A n'est pas supprimé
	Travail B	table de variables A est référencé dans le travail B	table de variables A n'est pas supprimé
	Invite B	table de variables A est référencé dans le texte de l'invite B	table de variables A n'est pas supprimé
Flot de travaux A	Flot de travaux B	Le flot de travaux B suit le flot de travaux A	Le flot de travaux A et la dépendance de prédécesseur/successeur dans le flot de travaux B sont supprimés.
	Travail B dans un flot de travaux B	Le travail B suit le flot de travaux A	Le flot de travaux A et la dépendance de prédécesseur/successeur dans le travail B sont supprimés.
	règle d'événement B	Le flot de travaux A est dans la définition d'action de la règle d'événement B (et ne fait appel à aucune substitution de variable)	Un message d'erreur spécifiant la relation existante s'affiche.

add

Ajoute ou met à jour les objets de planification dans la base de données.

Autorisation

Vous disposer des droits d'accès ***add*** pour ajouter un nouvel objet de planification. Si l'objet existe déjà dans la base de données, vous devez avoir les droits suivants :

- droit d'accès ***modify*** sur l'objet si ce dernier n'est pas verrouillé.
- droit d'accès ***modify*** et ***unlock*** sur l'objet s'il est verrouillé par un autre utilisateur.

Syntaxe

{**add** | **a**} *file_name* [**;unlock**]

Arguments

file_name
: Indique le nom du fichier texte qui contient les définitions d'objet. Pour les règles d'événement, *file_name* spécifie le nom du fichier XML contenant les définitions des règles d'événement que vous souhaitez ajouter (voir «Définition de règle d'événement», à la page 284 pour les références XML et «Editeur Composer», à la page 302 pour des informations sur la configuration d'un éditeur XML).

;unlock
: Indique que les définitions d'objet doivent être déverrouillées si elles ont été verrouillées par le même utilisateur dans la même session. Si vous n'avez pas verrouillé l'objet et que vous utilisez l'option ;unlock, vous recevez un message d'erreur et l'objet n'est pas remplacé.

Commentaires

Le fichier texte est validé sur le client et, s'il est correct, les objets sont insérés dans la base de données sur le gestionnaire de domaine maître. Le programme **composer** transforme les définitions d'objet en définition XML utilisée sur le serveur. Sinon, la commande est interrompue et un message d'erreur s'affiche. Ceci ne s'applique pas aux définitions de règles d'événement, car elles sont fournies directement au format XML.

Avec la commande **add**, le programme vous demande si vous souhaitez remplacer les objets existants. Ce comportement ne concerne pas les définitions de travail existantes dans les flots de travaux et les définitions de travail sont automatiquement mises à jour sans aucun message d'invite. Vous pouvez utiliser l'option **unlock** pour mettre à jour des objets existants que vous avez précédemment verrouillés à l'aide d'une seule commande. Pour tous les nouveaux objets insérés, l'option est ignorée. Si vous modifiez le nom d'un objet, il sera interprété par **composer** comme un nouvel objet et il sera inséré. Une commande **rename** est recommandée dans ce cas.

La commande add vérifie la présence de dépendances de boucle à l'intérieur des flots de travaux. Par exemple, si `job1` suit `job2` et si `job2` suit `job1`, il dépendance de boucle. Lorsqu'une dépendance de boucle à l'intérieur d'un flot de travaux est détectée, une erreur est affichée. La commande add ne vérifie pas les dépendances de boucle entre les flots de travaux parce que selon la complexité des activités de planification, cette vérification pourrait prendre trop de temps et de ressources d'UC.

Remarque : Pour importer dans la base de données les données enregistrées dans les fichiers à plat créés à l'aide de la commande **composer** dans une autre instance de Tivoli Workload Scheduler version 8.3 ou ultérieure, utilisez l'utilitaire **datamigrate** du gestionnaire de domaine maître au lieu de la commande**composer**, afin de gérer les fichiers à plat extraits. Pour plus d'informations sur l'utilitaire **datamigrate**, voir «datamigrate», à la page 550.

Exemples

Pour ajouter les travaux du fichier `myjobs`, exécutez la commande suivante :

```
add myjobs
```

Pour ajouter les flots de travaux du fichier `mysked`, exécutez la commande suivante :

```
a mysked
```

Pour ajouter les postes de travail, les classes de postes de travail et les domaines du fichier `cpus.src`, exécutez la commande suivante :

```
a cpus.src
```

Pour ajouter les définitions d'utilisateur du fichier `users_nt`, exécutez la commande suivante :

```
a users_nt
```

Pour ajouter les définitions de règles d'événement, vous éditez un fichier nommé `newrules.xml`, exécutez :

```
a newrules.xml
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer les mêmes tâches que celles décrites dans :

le manuel Dynamic Workload Console - Guide d'utilisation.

- Pour ajouter des postes de travail, voir

  le manuel Dynamic Workload Console - Guide d'utilisation, section "Création de postes de travail distribués".
- Pour ajouter des règles d'événement, voir

  le manuel Dynamic Workload Console - Guide d'utilisation, section Création d'une règle d'événement.
- Pour ajouter d'autres objets, voir

  le manuel Dynamic Workload Console - Guide d'utilisation, section "Conception de votre charge de travail".

authenticate

Bascule vers d'autres droits d'accès d'utilisateur lors de l'exécution de **composer**.

Autorisation

Tout utilisateur autorisé à exécuter **composer** est autorisé à émettre cette commande.

Syntaxe

{**authenticate** | **au**} [**username=***nom_utilisateur* **password=***mot_de_passe*]

Arguments

username=*nom_utilisateur*
Nom d'utilisateur de l'utilisateur vers lequel vous souhaitez basculer.

password=*mot_de_passe*
Mot de passe de l'utilisateur vers lequel vous souhaitez basculer.

Commentaires

Un message indique l'échec ou la réussite de l'authentification. Cette commande est uniquement utilisée en mode interactif.

Exemples

Pour basculer vers l'utilisateur **utilisateur_tws1** avec le mot de passe **mon_mot_de_passe1** à partir du programme de ligne de commande **composer**, exécutez la commande suivante :

```
au username=tws_user1 password=mypasswd1
```

continue

Indique que l'erreur de commande suivante doit être ignorée.

Autorisation

Tout utilisateur autorisé à exécuter composer est autorisé à émettre cette commande.

Syntaxe

{**continue** | **c**}

Commentaires

Cette commande est utile lorsque plusieurs commandes sont entrées sur la ligne de commande ou redirigées à partir d'un fichier. Elle indique au programme **composer** de continuer à exécuter les commandes même si celle qui a été lancée après **continue** génère une erreur. Elle n'est pas nécessaire lorsque vous entrez des commandes de manière interactive dans la mesure où le programme **composer** ne s'arrête pas sur une erreur.

Exemples

Si vous souhaitez que le programme composer continue à traiter la commande **print** si la commande **delete** génère une erreur, exécutez la commande suivante :

```
composer "c&delete cpu=site4&print cpu=@"
```

delete

Permet de supprimer des définitions d'objet dans la base de données.

Autorisation

Vous devez détenir les droits d'accès ***delete*** aux objets en cours de suppression.

Syntaxe

{**delete** | **de**}
{[**calendars** | **calendar** | **cal**=*nom_agenda*] |
[**domain** | **dom**=*nom_domaine*] |
[**eventrule** | **erule** | **er**=*nom_règle_événement*] |
[**parms** | **parm** | **vb**=[*nom_table.*]*nom_variable*] |
[**prompts** | **prom**=*nom_invite*] |
[**resources** | **resource** | **res**=[*poste_de_travail*#]*nom_ressource*] |
[**runcyclegroup** | **rcg**=*nom_groupe_cycle_exécution*] |
[**vartable** | **vt**=*nom_table*] |
| [**wat**=*workloadapplicationtemplatename*]
[**unité centrale**={*workstationame* [**;force**] | *workstationclassname* [**;force**]| *domainame*}]
[**workstation** | **ws**=*workstationame*] [**;force**] |
[**workstationclass** | **wscl**=*workstationclassname*] [**;force**] |
[**jobs** | **jobdefinition** | **jd**=[*nom_poste_travail*#]*nom_travail*] |
[**sched**|**jobstream**|**js**= [*nom_poste_travail*#]*nom_flot_travaux*
*[**valid from** date|**valid to** date |**valid in** date date]*
] |
[**users** | **user**=[*nom_poste_travail*#]*nom_utilisateur*]}
[**;noask**]

Arguments

calendars | calendar | cal
Si aucun argument ne suit, permet de supprimer toutes les définitions d'agenda.

Si l'argument *nom_agenda* suit, permet de supprimer l'agenda spécifié. Les caractères génériques sont acceptés.

domain | dom
Si aucun argument ne suit, permet de supprimer toutes les définitions de domaine.

Si l'argument *nom_domaine* suit, permet de supprimer le domaine spécifié. Les caractères génériques sont acceptés.

eventrule | erule | er
Si aucun argument ne suit, permet de supprimer toutes les définitions de règle d'événement.

Si l'argument *nom_règle_événement* suit, permet de supprimer la règle d'événement spécifiée. Les caractères génériques sont acceptés.

parms|parm|vb
Si aucun argument ne suit, permet de supprimer toutes les définitions de variable globale trouvées dans la table de variables par défaut.

Si l'argument *nom_table.nom_variable* suit, permet de supprimer la variable *nom_variable* de la table *nom_table*. Si *nom_table* est ignoré, **composer** recherche la définition de variable dans la table de variables par défaut. Les caractères génériques sont acceptés pour *nom_table* et *nom_variable*. Par exemple :

```
delete parms=@.@
```

Permet de supprimer toutes les variables de toutes les tables.

```
delete parms=@
```

Permet de supprimer toutes les variables de la table par défaut.

```
delete parms=@.acct@
```

Permet de supprimer toutes les variables dont le nom commence par `acct` de toutes les tables existantes.

A faire : Lors de la suppression d'une variable, la table de variables la contenant est verrouillée. Cela implique que, lorsque la table est verrouillée, aucun autre utilisateur ne peut exécuter d'autres commandes de verrouillage sur elle ou sur les variables qu'elle contient.

prompts | prom
Si aucun argument ne suit, permet de supprimer toutes les définitions d'invite.

Si l'argument *nom_invite* suit, permet de supprimer l'invite spécifiée. Les caractères génériques sont acceptés.

resources | resource | res
Si aucun argument ne suit, permet de supprimer toutes les définitions de ressource.

Si l'argument *nom_poste_travail#nom_ressource* suit, permet de supprimer la ressource *nom_ressource* du poste de travail *nom_poste_travail* sur lequel la ressource est définie. Si *nom_poste_travail* est ignoré, la valeur par défaut est le poste de travail sur lequel **composer** est exécuté. Les caractères génériques sont acceptés pour *nom_poste_travail* et *nom_ressource*.

runcyclegroup | rcg
Si aucun argument ne suit, permet de supprimer toutes les définitions de groupe de cycle d'exécution.

Si l'argument *nom_groupe_cycle_exécution* suit, permet de supprimer le groupe de cycle d'exécution spécifié. Les caractères génériques sont acceptés.

vartable | vt
Si aucun argument ne suit, permet de supprimer toutes les définitions de table de variables.

Si la table de variables *nom_table* de l'argument suit, permet de supprimer la table de variables spécifiée. Les caractères génériques sont acceptés.

wat Si aucun argument ne suit, permet de supprimer toutes les définitions de modèle d'application de charge de travail.

Si l'argument *modèle_application_charge_travail* suit, permet de supprimer le modèle d'application de charge de travail spécifié. Les caractères génériques sont acceptés.

cpu Supprime les postes de travail, les classes de postes de travail ou les domaines.

poste de travail
Nom du poste de travail. Les caractères génériques sont acceptés. Si vous indiquez l'argument **force**, la définition de poste de travail est supprimée de la base de données Tivoli Workload Scheduler.

workstationclass
Nom de la classe de poste de travail. Les caractères génériques sont acceptés. Si vous indiquez l'argument **force**, la classe de poste de travail est supprimée de la base de données Tivoli Workload Scheduler.

domaine
Nom du domaine. Les caractères génériques sont acceptés.

workstation | ws
Si aucun argument ne suit, permet de supprimer toutes les définitions de poste de travail.

Si l'argument *nom_poste_travail* suit, permet de supprimer le poste de travail spécifié. Les caractères génériques sont acceptés. Si vous indiquez l'argument **force**, la définition de poste de travail est supprimée de la base de données Tivoli Workload Scheduler.

workstationclass | wscl
Si aucun argument ne suit, permet de supprimer toutes les définitions de classe de poste de travail.

Si l'argument *nom_classe_poste_travail* suit, permet de supprimer la classe de poste de travail spécifiée. Les caractères génériques sont acceptés. Si vous indiquez l'argument **force**, la définition de classe de poste de travail est supprimée de la base de données Tivoli Workload Scheduler.

jobs | jobdefinition | jd
Si aucun argument ne suit, permet de supprimer toutes les définitions de travail.

Si l'argument *nom_poste_travail#nom_travail* suit, permet de supprimer le travail *nom_travail* du poste de travail *nom_poste_travail* sur lequel le travail s'exécute. Si *nom_poste_travail* est ignoré, la valeur par défaut est le poste de travail sur lequel **composer** est exécuté. Les caractères génériques sont acceptés pour *nom_poste_travail* et *nom_travail*.

sched | jobstream | js
Si aucun argument ne suit, permet de supprimer toutes les définitions de flot de travaux.

Si l'argument *nom_poste_travail#nom_flot_travaux* suit, permet de supprimer le flot de travaux *nom_flot_travaux* du poste de travail *nom_poste_travail* sur lequel le flot de travaux est défini. Si *nom_poste_travail* est ignoré, la valeur par défaut est le poste de travail sur lequel **composer** est exécuté. Les caractères génériques sont acceptés pour *nom_poste_travail* et *nom_flot_travaux*.

valid from
date Limite la sélection aux flots de travaux qui ont une date *valid from* égale à la valeur indiquée. Le format et *mm/jj/aaaa*.

valid to
date Limite la sélection aux flots de travaux qui ont une date *valid to* égale à la valeur indiquée. Le format et *mm/jj/aaaa*.

valid in
date date Intervalle de temps d'exécution du flot de travaux. Le format est *mm/jj/aaaa - mm/jj/aaaa*. L'une des deux dates peut être représentée par @.

users | user
Si aucun argument ne suit, permet de supprimer toutes les définitions d'utilisateur.

Si l'argument *nom_poste_travail#nom_utilisateur* suit, permet de supprimer l'utilisateur *nom_utilisateur* du poste de travail *nom_poste_travail* sur lequel l'utilisateur est défini. Si *nom_poste_travail* est ignoré, la valeur par défaut est le poste de travail sur lequel **composer** est exécuté. Les caractères génériques sont acceptés pour *nom_poste_travail* et *nom_utilisateur*. La zone du mot de passe n'est pas copiée pour des raisons de sécurité.

;noask Indique que l'utilisateur ne doit pas être invité à confirmer avant d'effectuer une opération sur chaque objet éligible.

Commentaires

Si vous utilisez des caractères génériques pour spécifier plusieurs définitions, le programme **composer** demande confirmation avant de supprimer une définition répondant aux critères. Une confirmation est requise avant la suppression de chaque définition correspondante si vous ne spécifiez pas l'option **noask**.

Les objets à supprimer ne doivent pas être verrouillés. Si certains objets correspondants sont verrouillés lors du traitement de la commande, un message d'erreur s'affiche avec la liste de ces objets.

Exemples

Pour supprimer le travail `job3` qui a été lancé sur le poste de travail `site3`, exécutez la commande suivante :

```
delete jobs=site3#job3
```

Pour supprimer tous les postes de travail dont le nom commence par `ux`, exécutez la commande suivante :

```
de cpu=ux@
```

Pour supprimer tous les flots de travaux dont le nom commence par `test` sur tous les postes de travail, exécutez la commande suivante :

```
de sched=@#test@
```

Pour supprimer toutes les règles d'événement nommées de `rulejs320` à `rulejs329`, exécutez la commande suivante :

```
de erule=rulejs32?
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer les mêmes tâches que celles décrites dans :

le manuel Dynamic Workload Console - Guide d'utilisation.

- Pour afficher, éditer ou supprimer des postes de travail, voir
  le manuel Dynamic Workload Console - Guide d'utilisation, section Edition de définitions de poste de travail.
- Pour afficher, éditer ou supprimer des règles d'événement, voir
  le manuel Dynamic Workload Console - Guide d'utilisation, section Edition d'une règle d'événement.

- Pour afficher, éditer ou supprimer tous les autres objets, voir
  le manuel Dynamic Workload Console - Guide d'utilisation, section Liste des définitions d'objet dans la base de données.

display

Permet d'afficher les détails d'une ou plusieurs définitions d'objet du même type enregistrées dans la base de données. La définition complète de l'objet s'affiche.

Autorisation

Vous devez avoir un droit d'accès ***display*** à l'objet qui s'affiche. Si vous souhaitez utiliser le mot clé **full**, vous devez également avoir le droit d'accès ***display*** aux travaux contenus dans la définition du flot de travaux. Si vous ne possédez pas l'accès requis, le composeur ne parvient pas à trouver les objets.

Syntaxe

{**display** | **di**}
{[**calendars** | **calendar** | **cal**=*nom_agenda*] |
[**eventrule** | **erule** | **er**=*nom_règle_événement*] |
[**parms** | **parm** | **vb**=*nom_variable.*]*nom_variable*] |
[**vartable** | **vt**=*nom_table*] |
[**prompts** | **prom**=*nom_invite*] |
[**resources** | **resource** | **res**=[*poste_de_travail*#]*nom_ressource*] |
[**runcyclegroup** | **rcg**=*nom_groupe_cycle_exécution*] |
[**cpu**={*nom_poste_travail* | *nom_classe_poste_de_travail* | *nom_domaine*}]
[**workstation** | **ws**=*nom_poste_travail*] |
[**workstationclass** | **wscl**=*nom_classe_poste_de_travail*] |
[**domain** | **dom**=*nom_domaine*] |
[**jobs** | **jobdefinition** | **jd**=[*nom_poste_travail*#]*nom_travail*] |
[**sched**|**jobstream**|**js**= [*nom_poste_travail*#]*nom_flot_travaux*
*[**valid from** date|**valid to** date |**valid in** date date]*
*[**;full**]*] |
[**users** | **user**=[*nom_poste_travail*#]*nom_utilisateur*]}
[**;offline**]

Arguments

calendars | calendar | cal
Si aucun argument ne suit, permet d'afficher toutes les définitions d'agenda.

Si l'argument *nom_agenda* suit, permet d'afficher l'agenda *nom_agenda*. Les caractères génériques sont acceptés.

eventrule | erule | er
Si aucun argument ne suit, permet d'afficher toutes les définitions de règle d'événement.

Si l'argument *nom_règle_événement* suit, permet d'afficher la règle d'événement *nom_règle_événement*. Les caractères génériques sont acceptés.

parms | parm | vb
Si aucun argument ne suit, permet d'afficher toutes les définitions de variable globale trouvées dans la table de variables par défaut.

Si l'argument *nom_table.nom_variable* suit, permet d'afficher la variable *nom_variable* de la table spécifiée. Si la table de variables *nom_table* est

ignorée, **composer** recherche la définition de variable dans la table de variables par défaut. Les caractères génériques sont utilisés dans la table de variables *nom_table* et dans la variable *nom_variable*. Par exemple :

```
display parms=@.@
```

Permet d'afficher toutes les variables dans toutes les tables.

```
display parms=@
```

Permet d'afficher toutes les variables dans la table par défaut.

```
display parms=@.acct@
```

Permet d'afficher toutes les variables dont le nom commence par `acct` dans toutes les tables existantes.

vartable | vt
Si aucun argument ne suit, permet d'afficher toutes les définitions de table de variables.

Si la table de variables *nom_table* de l'argument suit, permet d'afficher la table de variables *nom_table*. Les caractères génériques sont acceptés.

prompts | prom
Si aucun argument ne suit, permet d'afficher toutes les définitions d'invite.

Si l'argument *nom_invite* suit, permet d'afficher l'invite *nom_invite*. Les caractères génériques sont acceptés.

resources | resource | res
Si aucun argument ne suit, permet d'afficher toutes les définitions de ressource.

Si l'argument *nom_poste_travail#nom_ressource* suit, permet d'afficher la ressource *nom_ressource* du poste de travail *nom_poste_travail* sur lequel la ressource est définie. Si *nom_poste_travail* est ignoré, la valeur par défaut est le poste de travail sur lequel **composer** est exécuté. Les caractères génériques sont acceptés pour *nom_poste_travail* et *nom_ressource*.

runcyclegroup | rcg
Si aucun argument ne suit, permet d'afficher toutes les définitions de groupe de cycle d'exécution.

Si l'argument *runcyclegroupname* suit, permet d'afficher le groupe de cycle d'exécution *runcyclegroupname*. Les caractères génériques sont acceptés.

cpu Affiche les postes de travail, les classes de postes de travail ou les domaines.

poste de travail
Nom du poste de travail. Les caractères génériques sont acceptés.

workstationclass
Nom de la classe de poste de travail. Les caractères génériques sont acceptés.

domaine
Nom du domaine. Les caractères génériques sont acceptés.

workstation | ws
Si aucun argument ne suit, permet d'afficher toutes les définitions de poste de travail.

Si l'argument *nom_poste_travail* suit, permet d'afficher le poste de travail *nom_poste_travail*. Les caractères génériques sont acceptés.

domain | dom
Si aucun argument ne suit, permet d'afficher toutes les définitions de domaine.

Si l'argument *nom_domaine* suit, permet d'afficher le domaine *nom_domaine*. Les caractères génériques sont acceptés.

workstationclass | wscl
Si aucun argument ne suit, permet d'afficher toutes les définitions de classe de poste de travail.

Si l'argument *nom_classe_poste_de_travail* suit, permet d'afficher la classe de poste de travail *nom_classe_poste_de_travail*. Les caractères génériques sont acceptés.

jobs | jobdefinition | jd
Si aucun argument ne suit, permet d'afficher toutes les définitions de travail.

Si l'argument *nom_poste_travail#nom_travail* suit, permet d'afficher le travail *nom_travail* du poste de travail *nom_poste_travail* sur lequel le travail s'exécute. Si *nom_poste_travail* est ignoré, la valeur par défaut est le poste de travail sur lequel **composer** est exécuté. Les caractères génériques sont acceptés pour *nom_poste_travail* et *nom_travail*.

sched | jobstream | js
Si aucun argument ne suit, permet d'afficher toutes les définitions de flot de travaux.

Si l'argument *nom_poste_travail#nom_flot_travaux* suit, permet d'afficher le flot de travaux *nom_flot_travaux* du poste de travail *nom_poste_travail* sur lequel le flot de travaux est défini. Si *nom_poste_travail* est ignoré, la valeur par défaut est le poste de travail sur lequel **composer** est exécuté. Les caractères génériques sont acceptés pour *nom_poste_travail* et *nom_flot_travaux*.

valid from
date Limite la sélection aux flots de travaux qui ont une date *valid from* égale à la valeur indiquée. Le format et *mm/jj/aaaa*.

valid to
date Limite la sélection aux flots de travaux qui ont une date *valid to* égale à la valeur indiquée. Le format et *mm/jj/aaaa*.

valid in
date date Intervalle de temps d'exécution du flot de travaux. Le format est *mm/jj/aaaa - mm/jj/aaaa*. L'une des deux dates peut être représentée par @.

full Affiche également toutes les définitions de travail contenues dans le flot de travaux.

users | user
Si aucun argument ne suit, permet d'afficher toutes les définitions d'utilisateur.

Si l'argument *nom_poste_travail#nom_utilisateur* suit, permet d'afficher l'utilisateur *nom_utilisateur* du poste de travail *nom_poste_travail* sur lequel l'utilisateur est défini. Si *nom_poste_travail* est ignoré, la valeur par défaut

est le poste de travail sur lequel **composer** est exécuté. Les caractères génériques sont acceptés pour *nom_poste_travail* et *nom_utilisateur*.

;offline
Envoie la sortie de la commande au périphérique de sortie de **composer**. Pour plus d'informations sur ce périphérique, voir «Variables UNIX», à la page 302.

Résultats

La commande **display** vous renvoie les informations suivantes sur l'objet à afficher :

- une ligne récapitulative contenant des informations sur l'objet sélectionné
- la définition d'objet sélectionnée

En fonction de la valeur définie dans la variable locale ***MAESTROCOLUMNS***, la ligne récapitulative montre différents ensembles d'information sur l'objet sélectionné.

Le tableau 60 illustre la sortie produite en fonction de la valeur définie pour la variable ***MAESTROCOLUMNS***.

Tableau 60. Formats de sortie de l'affichage des objets de planification

Type d'objet	Format de sortie si MAESTROCOLUMNS<120	Format de sortie si MAESTROCOLUMNS ≥ 120
Agenda	"CalendarName : UpdatedOn : UpdatedBy : LockedBy"	"CalendarName : UpdatedBy : UpdatedOn : LockedBy : LockedOn"
Domaine	"DomainName : ParentDomain : Master : UpdatedOn : LockedBy"	"DomainName : ParentDomain : Master : UpdatedBy : UpdatedOn : LockedBy : LockedOn"
Règle d'événement	"EventRuleName : Type : Draft : Status : UpdatedOn : LockedBy"	"EventRuleName : Type : Draft : Status : UpdatedOn : LockedBy : LockedOn"
Travail	"Workstation : JobDefinitionName : UpdatedOn : LockedBy"	"Workstation : JobDefinitionName : TaskType : UpdatedBy : LockedBy : LockedOn"
Flot de travaux	"Workstation : JobstreamName : Validfrom : UpdatedOn : LockedBy"	"Workstation : JobstreamName : Draft : ValidFrom : ValidTo : UpdatedBy : UpdatedOn : LockedBy : LockedOn"
Paramètre	"VariableTableName : VariableName : UpdatedOn : LockedBy"	"VariableTableName : VariableName : UpdatedBy : UpdatedOn : LockedBy : LockedOn"
Invite	"PromptName : UpdatedOn : LockedBy "	"PromptName : UpdatedBy : UpdatedOn : LockedBy : LockedOn"
Ressource	"Workstation : ResourceName : Quantity : UpdatedOn : LockedBy "	"Workstation : ResourceName : Quantity : UpdatedBy : UpdatedOn : LockedBy : LockedOn"
Table de variables	"VariableTableName : Default : UpdatedOn : LockedBy "	"VariableTableName : Default : UpdatedBy : UpdatedOn : LockedBy : LockedOn"
User	"Workstation : UserName : UpdatedOn : LockedBy"	"UserName : Workstation : UpdatedBy : UpdatedOn : LockedBy : LockedOn"

Tableau 60. Formats de sortie de l'affichage des objets de planification (suite)

Type d'objet	Format de sortie si MAESTROCOLUMNS<120	Format de sortie si MAESTROCOLUMNS ≥ 120
Poste de travail	"WorkstationName : Type : Domain : Ignored : UpdatedOn : LockedBy"	"WorkstationName : Type : Domain : OsType : Ignored : UpdatedBy : UpdatedOn : LockedBy : LockedOn"
Classe de poste de travail	"WorkstationClassName : Ignored : UpdatedOn : LockedBy"	"WorkstationClassName : Ignored : UpdatedBy : UpdatedOn : LockedBy : LockedOn"

Voir «Sortie offline», à la page 301 pour plus d'informations sur la définition de ***MAESTROCOLUMNS*.**

Exemples

Pour afficher tous les agendas, exécutez la commande suivante :

```
display calendars=@
```

Voici un exemple de sortie :

```
Calendar Name     Updated On  Locked By
---------------  ----------  --------------------
HOLIDAYS     12/31/2005 tws83
HOLIDAYS
  01/01/2006 02/15/2006 05/31/2006


Calendar Name     Updated On  Locked By
---------------  ----------  --------------------
MONTHEND     01/01/2006 -

MONTHEND
  "Month end dates 1st half 2006"
  01/31/2006 02/28/2006 03/31/2006 04/30/2006 05/31/2006 06/30/2006

Calendar Name     Updated On  Locked By
---------------  ----------  --------------------
PAYDAYS      01/02/2006 -

PAYDAYS
  01/15/2006 02/15/2006 03/15/2006 04/15/2006 05/14/2006 06/15/2006
```

Pour imprimer la sortie de la commande d'affichage sur tous les flots de travaux lancés sur le poste de travail `site2`, exécutez la commande suivante :

```
di sched=site2#@;offline
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer les mêmes tâches que celles décrites dans :

le manuel Dynamic Workload Console - Guide d'utilisation.

- Pour afficher, éditer ou supprimer des postes de travail, voir
  le manuel Dynamic Workload Console - Guide d'utilisation, section Edition de définitions de poste de travail.
- Pour afficher, éditer ou supprimer des règles d'événement, voir

le manuel Dynamic Workload Console - Guide d'utilisation, section Edition d'une règle d'événement.

- Pour afficher, éditer ou supprimer tous les autres objets, voir
  le manuel Dynamic Workload Console - Guide d'utilisation, section Liste des définitions d'objet dans la base de données.

edit

Permet d'éditer un fichier.

Autorisation

Tout utilisateur autorisé à exécuter composer est autorisé à émettre cette commande.

Syntaxe

{**edit** | **ed**} *file_name*

Arguments

file_name
Nom du fichier à modifier.

Commentaires

Il convient de lancer un éditeur et d'ouvrir le fichier à modifier dans ce programme. Pour plus d'informations, voir «Editeur Composer», à la page 302.

Exemples

Pour modifier le fichier `mytemp`, exécutez la commande suivante :

```
edit mytemp
```

Pour modifier le fichier `resfile`, exécutez la commande suivante :

```
ed resfile
```

exit

Quitte le programme de ligne de commande **composer**.

Autorisation

Tout utilisateur autorisé à exécuter composer est autorisé à émettre cette commande.

Syntaxe

{**exit** | **e**}

Commentaires

Si vous exécutez le programme de ligne de commande **composer** en mode Aide, la commande active le programme **composer** en mode Saisie de commande.

Exemples

Pour quitter le programme de ligne de commande **composer**, exécutez la commande suivante :

```
exit
```

ou :

```
e
```

extract

Crée un fichier texte contenant les définitions d'objet extraites de la base de données.

Autorisation

Vous devez avoir le droit d'accès ***display*** sur les objets copiés et également le droit d'accès ***modify*** si vous souhaitez utiliser le mot clé **;lock**. Si vous ne possédez pas l'accès requis, le composeur ne parvient pas à trouver les objets.

Syntaxe

{**create** | **cr** | **extract** | **ext**} *file_name* **from**
{[**calendars** | **calendar** | **cal**=*nom_agenda*] |
[**eventrule** | **erule** | **er**=*nom_règle_événement*] |
[**parms** | **parm** | **vb**=[*nom_table.*]*nom_variable*] |
[**vartable** | **vt**=*nom_table*] |
[**prompts** | **prom**=*nom_invite*] |
[**resources** | **resource** | **res**=[*poste_de_travail*#]*nom_ressource*] |
[**runcyclegroup** | **rcg**=*nom_groupe_cycle_exécution*] |
[**cpu**={*nom_poste_travail* | *classe_poste_de_travail* | *nom_domaine*}] |
[**workstation** | **ws**=*nom_poste_travail*] |
[**workstationclass** | **wscl**=*nom_classe_poste_de_travail*] |
[**domain** | **dom**=*nom_domaine*] |
[**jobs** | **jobdefinition** | **jd**=[*nom_poste_travail*#]*nom_travail*] |
[**sched**|**jobstream**|**js**= [*nom_poste_travail*#]*nom_flot_travaux*
*[**valid from** date|**valid to** date |**valid in** date date]*
*[**;full**]*] |
[**users** | **user**=[*nom_poste_travail*#]*nom_utilisateur* [;password]]}
[**;lock**]

Arguments

file_name
Indique le nom du fichier qui contient les définitions d'objet.

calendars | calendar | cal
Si aucun argument ne suit, permet de copier toutes les définitions d'agenda dans le fichier.

Si l'argument *nom_agenda* suit, permet de copier l'agenda *nom_agenda* dans le fichier. Les caractères génériques sont acceptés.

eventrule | erule | er
Si aucun argument ne suit, permet de copier toutes les définitions de règle d'événement dans le fichier XML.

Si l'argument *nom_règle_événement* suit, permet de copier la règle d'événement *nom_règle_événement* dans le fichier. Les caractères génériques sont acceptés.

parms | parm | vb
Si aucun argument ne suit, permet de copier toutes les définitions de variable globale trouvées dans la table de variables par défaut dans le fichier.

Si l'argument *nom_table.nom_variable* suit, permet de copier la variable *nom_variable* de la variable *nom_table* spécifiée dans le fichier. Si la table de variables *nom_table* est ignorée, **composer** recherche la définition de variable dans la table de variables par défaut. Les caractères génériques sont acceptés dans la table de variables *nom_table* et dans la variable *nom_variable*.

Par exemple :

```
create parmfile from parms=@.@
```

Permet de copier toutes les variables à partir de toutes les tables.

```
create parmfile from parms=@
```

Permet de copier toutes les variables à partir de la table par défaut.

```
create parmfile from parms=@.acct@
```

Permet de copier toutes les variables dont le nom commence par `acct` à partir de toutes les tables existantes.

A faire : L'utilisation de l'option **;lock** sur une variable permet de verrouiller la table de variables qui la contient. Cela implique que, lorsque la table est verrouillée, aucun autre utilisateur ne peut exécuter d'autres commandes de verrouillage sur elle ou sur les variables qu'elle contient.

vartable | vt
Si aucun argument ne suit, permet de copier toutes les définitions de table de variables dans le fichier.

Si la table de variables *nom_table* de l'argument suit, permet de copier la table de variables *nom_table* dans le fichier. Les caractères génériques sont acceptés.

prompts | prom
Si aucun argument ne suit, permet de copier toutes les définitions d'invite dans le fichier.

Si l'argument *nom_invite* suit, permet de copier l'invite *nom_invite* dans le fichier. Les caractères génériques sont acceptés.

resources | resource | res
Si aucun argument ne suit, permet de copier toutes les définitions de ressource dans le fichier.

Si l'argument *nom_poste_travail#nom_ressource* suit, permet de copier la ressource *nom_ressource* du poste de travail *nom_poste_travail* sur lequel la ressource est définie dans le fichier. Si *nom_poste_travail* est ignoré, la valeur par défaut est le poste de travail sur lequel **composer** est exécuté. Les caractères génériques sont acceptés pour *nom_poste_travail* et *nom_ressource*.

runcyclegroup | rcg
Si aucun argument ne suit, permet de copier toutes les définitions de groupe de cycle d'exécution dans le fichier.

Si l'argument *runcyclegroupname* suit, permet de copier le groupe de cycle d'exécution *runcyclegroupname*. Les caractères génériques sont acceptés.

cpu Copie les postes de travail, les classes de postes de travail ou les domaines dans le fichier.

poste de travail
Nom du poste de travail. Les caractères génériques sont acceptés.

workstationclass
Nom de la classe de poste de travail. Les caractères génériques sont acceptés.

domaine
Nom du domaine. Les caractères génériques sont acceptés.

workstation | ws
Si aucun argument ne suit, permet de copier toutes les définitions de poste de travail dans le fichier.

Si l'argument *nom_poste_travail* suit, permet de copier le poste de travail *nom_poste_travail* dans le fichier. Les caractères génériques sont acceptés.

domain | dom
Si aucun argument ne suit, permet de copier toutes les définitions de domaine dans le fichier.

Si l'argument *nom_domaine* suit, permet de copier le domaine *nom_domaine* dans le fichier. Les caractères génériques sont acceptés.

workstationclass | wscl
Si aucun argument ne suit, permet de copier toutes les définitions de classe de poste de travail dans le fichier.

Si l'argument *nom_classe_poste_de_travail* suit, permet de copier la classe de poste de travail *nom_classe_poste_de_travail* dans le fichier. Les caractères génériques sont acceptés.

jobs | jobdefinition | jd
Si aucun argument ne suit, permet de copier toutes les définitions de travail dans le fichier.

Si l'argument *nom_poste_travail#nom_travail* suit, permet de copier le travail *nom_travail* du poste de travail *nom_poste_travail* sur lequel le travail s'exécute dans le fichier. Si *nom_poste_travail* est ignoré, la valeur par défaut est le poste de travail sur lequel **composer** est exécuté. Les caractères génériques sont acceptés pour *nom_poste_travail* et *nom_travail*.

sched | jobstream | js
Si aucun argument ne suit, permet de copier toutes les définitions de flot de travaux dans le fichier.

Si l'argument *nom_poste_travail#nom_flot_travaux* suit, permet de copier le flot de travaux *nom_flot_travaux* du poste de travail *nom_poste_travail* sur lequel le flot de travaux est défini dans le fichier. Si *nom_poste_travail* est ignoré, la valeur par défaut est le poste de travail sur lequel **composer** est exécuté. Les caractères génériques sont acceptés pour *nom_poste_travail* et *nom_flot_travaux*.

valid from
date Limite la sélection aux flots de travaux qui ont une date *valid from* égale à la valeur indiquée. Le format et *mm/jj/aaaa*.

valid to
date Limite la sélection aux flots de travaux qui ont une date *valid to* égale à la valeur indiquée. Le format et *mm/jj/aaaa*.

valid in
date date Intervalle de temps d'exécution du flot de travaux. Le format est *mm/jj/aaaa - mm/jj/aaaa*. L'une des deux dates peut être représentée par @.

full Copie également toutes les définitions de travail contenues dans le flot de travaux.

users | user
Si aucun argument ne suit, permet de copier toutes les définitions d'utilisateur dans le fichier.

Si l'argument *nom_poste_travail#nom_utilisateur* suit, permet de copier l'utilisateur *nom_utilisateur* du poste de travail *nom_poste_travail* sur lequel l'utilisateur est défini dans le fichier. Si *nom_poste_travail* est ignoré, la valeur par défaut est le poste de travail sur lequel **composer** est exécuté. Les caractères génériques sont acceptés pour *nom_poste_travail* et *nom_utilisateur*.

Si vous n'ajoutez pas l'option **;password**, le mot de passe défini pour l'utilisateur est enregistré dans le fichier de sortie sous la forme d'une séquence de 10 astérisques (*) et ne peut pas être réutilisé.

Si vous ajoutez l'option **;password**, le mot de passe défini pour l'utilisateur est chiffré et enregistré dans le fichier de sortie. Il peut être ainsi réimporté et utilisé à nouveau.

;lock Indique que l'objet sélectionné doit toujours être verrouillé.

Commentaires

Vous pouvez utiliser cette commande pour créer un fichier contenant des définitions de paramètres à importer dans la base de données de paramètres définie localement sur un poste de travail. Pour plus d'informations sur la façon d'importer les définitions de paramètres localement, voir «parms», à la page 568.

Vous pouvez appeler la commande avec l'ancien nom “create” ou le nouveau nom “extract”. Sans option **lock**, le verrouillage de la base de données n'est pas vérifié et tous les objets correspondants sont extraits dans le fichier. Après avoir créé un fichier, vous pouvez le modifier à l'aide de la commande **edit** et ajouter ou mettre à jour la base de données à l'aide de la commande **add** ou **replace**.

Avec l'option **lock**, vous pouvez préciser si l'utilisateur de la base de données doit laisser verrouillés les objets qui répondent aux critères sélectionnés. Au cours de l'extraction, si **composer** détecte que certains de ces objets sont déjà verrouillés par un autre utilisateur, ces objets ne sont pas insérés dans le fichier et un message *stdout* s'affiche pour chaque objet verrouillé.

Exemples

Pour créer un fichier nommé `caltemp` contenant tous les agendas, exécutez la commande suivante :

```
create caltemp from calendars=@
```

Pour créer un fichier nommé `stemp` contenant tous les flots de travaux définis sur le poste de travail sur lequel s'exécute **composer**, exécutez la commande suivante :

```
cr stemp from jobstream=@
```

Pour créer un fichier nommé `alljobs.txt` contenant toutes les définitions de travail, exécutez la commande suivante :

```
extract alljobs.txt from jd=@#@
```

Pour créer un fichier nommé `allrules.xml` contenant toutes les définitions de règles d'événement, exécutez la commande suivante :

```
ext allrules.xml from erule=@
```

Pour créer un fichier intitulé `dbmainadm.txt` avec la définition de l'utilisateur `princeps` du poste de travail `dbserv349`, y compris le mot de passe chiffré, exécutez la commande :

```
composer extract c:\dbmainadm.txt from user=dbserv349#princeps;password
```

Le contenu du fichier `dbmainadm.txt` sera :

```
USERNAME princeps
  PASSWORD "ENCRYPT:EIu7PP+gvS8="
END
```

help

Permet d'afficher l'aide en ligne d'une commande ou la liste des commandes qui peuvent être exécutées à partir de **composer**. Non disponible sous Windows.

Autorisation

Tout utilisateur autorisé à exécuter composer est autorisé à émettre cette commande.

Syntaxe

{**help** | **h**} {*commande*|*mot_clé*}

Arguments

commande

Indique le nom d'une commande **composer** ou d'une commande système. Pour les commandes **composer**, entrez le nom complet de la commande ; les abréviations et les formats abrégés ne sont pas pris en charge.

mot_clé

Vous pouvez également entrer les mots clés suivants :

COMMANDS
Répertorie toutes les commandes composer.

RUNCONPOSER
Comment exécuter composer.

SETUPCOMPOSER
Décrit comment configurer la commande composer.

SPECIALCHAR
Décrit les caractères génériques, les délimiteurs et les autres caractères spéciaux que vous pouvez utiliser.

Exemples

Pour afficher une liste des toutes les commandes **composer**, exécutez la commande suivante :

```
help commands
```

Pour afficher des informations sur la commande **add**, exécutez la commande suivante :

```
help add
```

Pour afficher des informations relatives aux caractères spéciaux que vous pouvez utiliser, exécutez la commande suivante :

```
h specialchar
```

list

Permet de répertorier ou d'imprimer les informations récapitulatives sur des objets définis dans la base de données Tivoli Workload Scheduler. La commande list vous fournit la liste des noms d'objets et de leurs attributs. La commande Print envoie la liste des noms d'objets et de leurs attributs au périphérique ou au fichier défini dans la variable locale ***MAESTROLP***. La commande print peut être utilisée pour envoyer la sortie à une imprimante locale, à condition que la variable ***MAESTROLP*** soit définie en conséquence.

Autorisation

Si l'option globale ***enListSecChk*** est définie sur **yes** sur le gestionnaire de domaine maître, vous devez, pour afficher ou imprimer un objet, posséder un accès de type ***list*** et/ou ***list*** et ***display***.

Syntaxe

{**list** | **l**}
{[**calendars** | **calendar** | **cal**=*nom_agenda*] |
[**eventrule** | **erule** | **er**=*nom_règle_événement*] |
[**parms** | **parm** | **vb**=[*nom_table*.]*nom_variable*] |
[**vartable** | **vt**=*nom_table*] |
[**prompts** | **prom**=*nom_invite*] |
[**resources** | **resource** | **res**=[*poste_de_travail*#]*nom_ressource*] |
[**runcyclegroup** | **rcg**=*nom_groupe_cycle_exécution*] |
[**cpu**={*nom_poste_travail* | *nom_classe_poste_de_travail* | *nom_domaine*}]
[**workstation** | **ws**=*nom_poste_travail*] |
[**workstationclass** | **wscl**=*nom_classe_poste_de_travail*] |
[**domain** | **dom**=*nom_domaine*] |
[**jobs** | **jobdefinition** | **jd**=[*nom_poste_travail*#]*nom_travail*] |
[**sched**|**jobstream**|**js**= [*nom_poste_travail*#]*nom_flot_travaux*
*[**valid from** date*|
valid to** date* |valid in** date date]* |
[**users** | **user**=[*nom_poste_travail*#]*nom_utilisateur*]}
[**;offline**]

Arguments

calendars | calendar | cal

Si aucun argument ne suit, permet de répertorier ou d'imprimer toutes les définitions d'agenda.

Si l'argument *nom_agenda* suit, permet de répertorier ou d'imprimer l'agenda *nom_agenda*. Les caractères génériques sont acceptés.

eventrule | erule | er

Si aucun argument ne suit, permet de répertorier ou d'imprimer toutes les définitions de règle d'événement.

Si l'argument *nom_règle_événement* suit, permet de répertorier ou d'imprimer la règle d'événement *nom_règle_événement*. Les caractères génériques sont acceptés.

parms | parm | vb

Si aucun argument ne suit, permet de répertorier ou d'imprimer toutes les définitions de variable globale trouvées dans la table de variables par défaut.

Si l'argument *nom_table.nom_variable* suit, permet de répertorier ou d'imprimer la variable *nom_variable* de la table *nom_table*. Si *nom_table* est ignoré, **composer** recherche la définition de variable dans la table de variables par défaut. Les caractères génériques sont utilisés dans *nom_table* et *nom_variable*. Par exemple :

```
list parms=@.@
```

Répertorie toutes les variables de toutes les tables.

```
list parms=@
```

Répertorie toutes les variables de la table par défaut.

```
list parms=@.acct@
```

Répertorie toutes les variables dont le nom commence par `acct` dans toutes les tables existantes.

vartable | vt

Si aucun argument ne suit, permet de répertorier ou d'imprimer toutes les définitions de table de variables.

Si la variable *nom_table* de l'argument suit, permet de répertorier ou d'imprimer la table de variables *nom_table*. Les caractères génériques sont acceptés.

prompts | prom

Si aucun argument ne suit, permet de répertorier ou d'imprimer toutes les définitions d'invite.

Si l'argument *nom_invite* suit, permet de répertorier ou d'imprimer l'invite *nom_invite*. Les caractères génériques sont acceptés.

resources | resource | res

Si aucun argument ne suit, permet de répertorier ou d'imprimer toutes les définitions de ressource.

Si l'argument *nom_poste_travail#nom_ressource* suit, permet de répertorier ou d'imprimer la ressource *nom_ressource* du poste de travail *nom_poste_travail* sur lequel la ressource est définie. Si *nom_poste_travail* est ignoré, la valeur

par défaut est le poste de travail sur lequel **composer** est exécuté. Les caractères génériques sont acceptés pour *nom_poste_travail* et *nom_ressource*.

runcyclegroup | rcg
Si aucun argument ne suit, permet de répertorier ou d'imprimer tous les groupes de cycle d'exécution.

Si l'argument *runcyclegroupname* suit, permet de répertorier ou d'imprimer le groupe de cycle d'exécution *runcyclegroupname*. Les caractères génériques sont acceptés.

cpu Répertorie ou imprime les postes de travail, les classes de postes de travail ou les domaines.

poste de travail
Nom du poste de travail. Les caractères génériques sont acceptés.

workstationclass
Nom de la classe de poste de travail. Les caractères génériques sont acceptés.

domaine
Nom du domaine. Les caractères génériques sont acceptés.

workstation | ws
Si aucun argument ne suit, permet de répertorier ou d'imprimer toutes les définitions de poste de travail.

Si l'argument *nom_poste_travail* suit, permet de répertorier ou d'imprimer le poste de travail *nom_poste_travail*. Les caractères génériques sont acceptés.

domain | dom
Si aucun argument ne suit, permet de répertorier ou d'imprimer toutes les définitions de domaine.

Si l'argument *nom_domaine* suit, permet de répertorier ou d'imprimer le domaine *nom_domaine*. Les caractères génériques sont acceptés.

workstationclass | wscl
Si aucun argument ne suit, permet de répertorier ou d'imprimer toutes les définitions de classe de poste de travail.

Si l'argument *nom_classe_poste_de_travail* suit, permet de répertorier ou d'imprimer la classe de poste de travail *nom_classe_poste_de_travail*. Les caractères génériques sont acceptés.

jobs | jobdefinition | jd
Si aucun argument ne suit, permet de répertorier ou d'imprimer toutes les définitions de travail.

Si l'argument *nom_poste_travail#nom_travail* suit, permet de répertorier ou d'imprimer le travail *nom_travail* du poste de travail *nom_poste_travail* sur lequel le travail s'exécute. Si *nom_poste_travail* est ignoré, la valeur par défaut est le poste de travail sur lequel **composer** est exécuté. Les caractères génériques sont acceptés pour *nom_poste_travail* et *nom_travail*.

sched | jobstream | js
Si aucun argument ne suit, permet de répertorier ou d'imprimer toutes les définitions de flot de travaux.

Si l'argument *nom_poste_travail#nom_flot_travaux* suit, permet de répertorier ou d'imprimer le flot de travaux *nom_flot_travaux* du poste de travail *nom_poste_travail* sur lequel le flot de travaux est défini. Si *nom_poste_travail*

est ignoré, la valeur par défaut est le poste de travail sur lequel **composer** est exécuté. Les caractères génériques sont acceptés pour *nom_poste_travail* et *nom_flot_travaux*.

valid from
date Limite la sélection aux flots de travaux qui ont une date *valid from* égale à la valeur indiquée. Le format et *mm/jj/aaaa*.

valid to
date Limite la sélection aux flots de travaux qui ont une date *valid to* égale à la valeur indiquée. Le format et *mm/jj/aaaa*.

valid in
date date Intervalle de temps d'exécution du flot de travaux. Le format est *mm/jj/aaaa - mm/jj/aaaa*. L'une des deux dates peut être représentée par @.

users | user
Si aucun argument ne suit, permet de répertorier ou d'imprimer toutes les définitions d'utilisateur.

Si l'argument *nom_poste_travail#nom_utilisateur* suit, permet de répertorier ou d'imprimer l'utilisateur *nom_utilisateur* du poste de travail *nom_poste_travail* sur lequel l'utilisateur est défini. Si *nom_poste_travail* est ignoré, la valeur par défaut est le poste de travail sur lequel **composer** est exécuté. Les caractères génériques sont acceptés pour *nom_poste_travail* et *nom_utilisateur*.

Remarque : Si vous répertoriez les utilisateurs Windows au format UPN `nom_utilisateur@domaine_internet`, insérez le caractère d'échappement '\' avant le caractère '@' dans la valeur `nom_utilisateur@domaine_internet`. Par exemple, si vous faites référence à l'utilisateur `administrator@bvt.com`, exécutez la commande suivante :

```
list users=administrator\@bvt.com
```

;offline
Envoie la sortie de la commande au périphérique de sortie de **composer**. Pour plus d'informations sur ce périphérique, voir «Variables UNIX», à la page 302. La commande **list ..... ;offline** est équivalente à la commande **print**.

Résultats

La commande list vous fournit la liste des noms d'objets et de leurs attributs. La commande print envoie la liste des noms d'objets et de leurs attributs au périphérique ou au fichier défini dans la variable locale *MAESTROLP*. La commande print peut être utilisée pour envoyer la sortie vers une imprimante locale si vous avez attribué une valeur à *MAESTROLP* en conséquence. Assurez-vous que la variable *MAESTROLP* est définie dans votre environnement avant d'exécuter la commande print.

En fonction de la valeur définie dans la variable locale ***MAESTROCOLUMNS***, les différents ensembles d'informations sur l'objet sélectionné peuvent être affichés.

Le tableau 61, à la page 339 illustre la sortie produite en fonction de la valeur définie pour la variable ***MAESTROCOLUMNS***.

Tableau 61. Formats de sortie de l'affichage des objets de planification

Type d'objet	Format de sortie si MAESTROCOLUMNS<120	Format de sortie si MAESTROCOLUMNS ≥ 120
Agenda	"CalendarName : UpdatedOn : UpdatedBy : LockedBy"	"CalendarName : UpdatedBy : UpdatedOn : LockedBy : LockedOn"
Domaine	"DomainName : ParentDomain : Master : UpdatedOn : LockedBy"	"DomainName : ParentDomain : Master : UpdatedBy : UpdatedOn : LockedBy : LockedOn"
Règle d'événement	"EventRuleName : Type : Draft : Status : UpdatedOn : LockedBy"	"EventRuleName : Type : Draft : Status : UpdatedOn : LockedBy : LockedOn"
Travail	"Workstation : JobDefinitionName : UpdatedOn : LockedBy"	"Workstation : JobDefinitionName : TaskType : UpdatedBy : LockedBy : LockedOn"
Flot de travaux	"Workstation : JobstreamName : Validfrom : UpdatedOn : LockedBy"	"Workstation : JobstreamName : Draft : ValidFrom : ValidTo : UpdatedBy : UpdatedOn : LockedBy : LockedOn"
Paramètre	"VariableTableName : VariableName : UpdatedOn : LockedBy"	"VariableTableName : VariableName : UpdatedBy : UpdatedOn : LockedBy : LockedOn"
Invite	"PromptName : UpdatedOn : LockedBy "	"PromptName : UpdatedBy : UpdatedOn : LockedBy : LockedOn"
Ressource	"Workstation : ResourceName : Quantity : UpdatedOn : LockedBy "	"Workstation : ResourceName : Quantity : UpdatedBy : UpdatedOn : LockedBy : LockedOn"
Groupe de cycle d'exécution	"RunCycleGroupName : UpdatedOn : LockedBy "	"RunCycleGroupName : UpdatedBy : UpdatedOn : LockedBy : LockedOn"
Table de variables	"VariableTableName : Default : UpdatedOn : LockedBy "	"VariableTableName : Default : UpdatedBy : UpdatedOn : LockedBy : LockedOn"
Utilisateur	"Workstation : UserName : UpdatedOn : LockedBy"	"UserName : Workstation : UpdatedBy : UpdatedOn : LockedBy : LockedOn"
Poste de travail	"WorkstationName : Type : Domain : Ignored : UpdatedOn : LockedBy"	"WorkstationName : Type : Domain : OsType : Ignored : UpdatedBy : UpdatedOn : LockedBy : LockedOn"
Classe de poste de travail	"WorkstationClassName : Ignored : UpdatedOn : LockedBy"	"WorkstationClassName : Ignored : UpdatedBy : UpdatedOn : LockedBy : LockedOn"

Voir «Sortie offline», à la page 301 pour plus d'informations sur la définition de ***MAESTROLP***.

Exemples

- Pour répertorier tous les agendas, exécutez la commande suivante :

```
list calendars=@
```

  Voici un exemple de sortie :

```
Calendar Name     Updated On  Locked By
---------------  ----------  --------------------
HOLIDAYS         03/02/2010
```

```
PAYDAYS           03/02/2010
HOLIDAYS          03/02/2010
  01/01/2010 02/15/2010 05/31/2010


Calendar Name     Updated On  Locked By
----------------  ----------  --------------------
MONTHEND          01/01/2010  -

MONTHEND
  "Month end dates 1st half 2010"
  01/31/2010 02/28/2010 03/31/2010 04/30/2010 05/31/2010 06/30/2010

Calendar Name     Updated On  Locked By
----------------  ----------  --------------------
PAYDAYS           01/02/2010  -

PAYDAYS
  01/15/2010 02/15/2010 03/15/2010 04/15/2010 05/14/2010 06/15/2010
```

- Pour répertorier toutes les règles d'événement définies, exécutez la commande suivante :

  ```
  list er=@
  ```

  Si MAESTROCOLUMNS=80, la sortie est similaire à ceci :

  ```
  Event Rule Name                          Type      Draft Status    Updated On  Locked By
  ---------------  ---------  -----  ---------  ----------  -------------
  EVENT-MULTIPLE1         filter          active   06/06/2009 -
  EVENT-MULTIPLE2         filter          active   06/06/2009 -
  EVENT-MULTIPLE3         filter          active   06/06/2009 -
  M_SUCC_12_S             sequence   Y    inactive 06/07/2009 -
  M_SUCC_12_S_A           filter          active   06/07/2009 -
  M_SUCC_12_S_B           filter     Y    inactive 06/07/2009 -
  NEWEVENTRULE            filter          active   06/01/2009 administrator
  ```

  Si MAESTROCOLUMNS≥120 la sortie est similaire à ceci :

  ```
  Event Rule Name                          Type      Draft Status    Updated On  Locked By
  --------------------   --------- ----- -------- ---------- ---------
  EVENT-MULTIPLE1         filter          active   06/06/2009 -
  EVENT-MULTIPLE2         filter          active   06/06/2009 -
  EVENT-MULTIPLE3         filter          active   06/06/2009 -
  M_SUCC_12_S             sequence   Y    inactive 06/07/2009 -
  M_SUCC_12_S_A           filter          active   06/07/2009 -
  M_SUCC_12_S_B           filter     Y    inactive 06/07/2009 -
  NEWEVENTRULE            filter          active   06/01/2009 administrator
  ```

- Pour afficher les propriétés du poste de travail NC1150691 de l'agent, exécutez la commande suivante :

  ```
  list ws=NC1150691
  ```

  Le résultat affiché est du type suivant :

  ```
  Workstation Name  Type     Domain            Ignored  Updated On  Locked By
  ----------------  -------  ----------------  -------  ----------  ----------
  NC1150691         agent    -                           03/31/2010  -

  CPUNAME NC1150691
    DESCRIPTION "This workstation was automatically created at agent
                 installation time."
    OS WNT
    NODE nc115069.romelab.it.ibm.com SECUREADDR 22114
    TIMEZONE GMT+1
    FOR MAESTRO HOST NC115069_DWB
      TYPE AGENT
      PROTOCOL HTTPS
  END
  ```

- Pour afficher les propriétés du poste de travail du pool POOL_A, y compris de tous ses membres, exécutez la commande suivante :

  ```
  list ws=POOL_A
  ```

  Le résultat affiché est du type suivant :

  ```
  Workstation Name  Type     Domain            Ignored  Updated On  Locked By
  ----------------  -------  ----------------  -------  ----------  ----------------
  POOL_A            pool     -                          03/31/2010  -

  CPUNAME POOL_A
    DESCRIPTION "This is a manually created pool"
    VARTABLE TABLE1
    OS OTHER
    TIMEZONE America/Argentina/Buenos_Aires
    FOR MAESTRO HOST NC115069_DWB
      TYPE POOL
    MEMBERS
      NC1150691
      NC1150692
  END
  ```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer les mêmes tâches que celles décrites dans :

le manuel Dynamic Workload Console - Guide d'utilisation.

- Pour afficher, éditer ou supprimer des postes de travail, voir

  le manuel Dynamic Workload Console - Guide d'utilisation, section Edition de définitions de poste de travail.
- Pour afficher, éditer ou supprimer des règles d'événement, voir

  le manuel Dynamic Workload Console - Guide d'utilisation, section Edition d'une règle d'événement.
- Pour afficher, éditer ou supprimer tous les autres objets, voir

  le manuel Dynamic Workload Console - Guide d'utilisation, section Liste des définitions d'objet dans la base de données.

lock

Permet de verrouiller l'accès aux définitions des objets de planification dans la base de données.

Autorisation

Vous devez avoir un accès ***modify*** à l'objet.

Syntaxe

{**lock** | **lo**}
{[**calendars** | **calendar** | **cal**=*nom_agenda*] |
[**eventrule** | **erule** | **er**=*nom_règle_événement*]
[**parms** | **parm** | **vb**=[*nom_table.*]*nom_variable*] |
[**vartable** | **vt**=*nom_table*] |
[**prompts** | **prom**=*nom_invite*] |
[**resources** | **resource** | **res**=[*poste_de_travail*#]*nom_ressource*] |
[**runcyclegroup** | **rcg**=*nom_groupe_cycle_exécution*] |
[**cpu**={*nom_poste_travail* | *nom_classe_poste_de_travail* | *nom_domaine*}]
[**workstation** | **ws**=*nom_poste_travail*] |

[**workstationclass** | **wscl**=*nom_classe_poste_de_travail*] |
[**domain** | **dom**=*nom_domaine*] |
[**jobs** | **jobdefinition** | **jd**=[*nom_poste_travail*#]*nom_travail*] |
[**sched**|**jobstream**|**js**= [*nom_poste_travail*#]*nom_flot_travaux*
*[**début de validité** date|**fin de validité** date |**valid in** date date]*] |
[**users** | **user**=[*nom_poste_travail*#]*nom_utilisateur*]}

Arguments

agendas
Verrouille toutes les définitions d'agenda.

calendars | calendar | cal
Si aucun argument ne suit, permet de verrouiller toutes les définitions d'agenda.

Si l'argument *nom_agenda* suit, permet de verrouiller l'agenda *nom_agenda*. Les caractères génériques sont acceptés.

eventrule | erule | er
Si aucun argument ne suit, permet de verrouiller toutes les définitions de règle d'événement.

Si l'argument *nom_règle_événement* suit, permet de verrouiller la règle d'événement *nom_règle_événement*. Les caractères génériques sont acceptés.

parms | parm | vb
Si aucun argument ne suit, permet de verrouiller toute la table de variables par défaut.

Si l'argument *nom_table.nom_variable* suit, permet de verrouiller toute la table contenant la variable *nom_variable*. Si *nom_table* est ignoré, **composer** verrouille toute la table de variables par défaut.

Remarque : Le verrouillage d'une table verrouille également toutes la table de variables qui la contient. Cela implique que, lorsque la table est verrouillée, aucun autre utilisateur ne peut exécuter d'autres commandes de verrouillage sur elle.
Les caractères génériques sont utilisés dans *nom_table* et *nom_variable*. Par exemple :

```
lock parms=@.@
```

Permet de verrouiller toutes les variables sur toutes les tables. En conséquence, toutes les tables de variables sont verrouillées.

```
lock parms=@
```

Permet de verrouiller toutes les variables de la table par défaut. En conséquence, la table de variables est verrouillée.

```
lock parms=@.acct@
```

Permet de verrouiller toutes les variables dont le nom commence par `acct` dans toutes les tables existantes. En conséquence, toutes les tables de variables contenant au moins une variable nommée ainsi sont verrouillées.

vartable | vt
Si aucun argument ne suit, permet de verrouiller toutes les définitions de table de variables.

Si la table de variables *nom_table* de l'argument suit, permet de verrouiller la table de variables *nom_table*. Les caractères génériques sont acceptés.

prompts | prom
Si aucun argument ne suit, permet de verrouiller toutes les définitions d'invite.

Si l'argument *nom_invite* suit, permet de verrouiller l'invite *nom_invite*. Les caractères génériques sont acceptés.

resources | resource | res
Si aucun argument ne suit, permet de verrouiller toutes les définitions de ressource.

Si l'argument *nom_poste_travail#nom_ressource* suit, permet de verrouiller la ressource *nom_ressource* du poste de travail *nom_poste_travail* sur lequel la ressource est définie. Si *nom_poste_travail* est ignoré, la valeur par défaut est le poste de travail sur lequel **composer** est exécuté. Les caractères génériques sont acceptés pour *nom_poste_travail* et *nom_ressource*.

runcyclegroup | rcg
Si aucun argument ne suit, permet de verrouiller toutes les définitions de groupe de cycle d'exécution.

Si l'argument *runcyclegroupname* suit, permet de verrouiller le groupe de cycle d'exécution *runcyclegroupname*. Les caractères génériques sont acceptés.

cpu Verrouille les postes de travail, les classes de postes de travail ou les domaines.

poste de travail
Nom du poste de travail. Les caractères génériques sont acceptés.

workstationclass
Nom de la classe de poste de travail. Les caractères génériques sont acceptés.

domaine
Nom du domaine. Les caractères génériques sont acceptés.

workstation | ws
Si aucun argument ne suit, permet de verrouiller toutes les définitions de pote de travail.

Si l'argument *nom_poste_travail* suit, permet de verrouiller le poste de travail *nom_poste_travail*. Les caractères génériques sont acceptés.

domain | dom
Si aucun argument ne suit, permet de verrouiller toutes les définitions de domaine.

Si l'argument *nom_domaine* suit, permet de verrouiller le domaine *nom_domaine*. Les caractères génériques sont acceptés.

workstationclass | wscl
Si aucun argument ne suit, permet de verrouiller toutes les définitions de classe de poste de travail.

Si l'argument *nom_classe_poste_de_travail* suit, permet de verrouiller la classe de poste de travail *nom_classe_poste_de_travail*. Les caractères génériques sont acceptés.

jobs | jobdefinition | jd
Si aucun argument ne suit, permet de verrouiller toutes les définitions de travail.

Si l'argument *nom_poste_travail#nom_travail* suit, permet de verrouiller le travail *nom_travail* du poste de travail *nom_poste_travail* sur lequel le travail s'exécute. Si *nom_poste_travail* est ignoré, la valeur par défaut est le poste de travail sur lequel **composer** est exécuté. Les caractères génériques sont acceptés pour *nom_poste_travail* et *nom_travail*.

sched | jobstream | js
Si aucun argument ne suit, permet de verrouiller toutes les définitions de flot de travaux.

Si l'argument *nom_poste_travail#nom_flot_travaux* suit, permet de verrouiller le flot de travaux *nom_flot_travaux* du poste de travail *nom_poste_travail* sur lequel le flot de travaux est défini. Si *nom_poste_travail* est ignoré, la valeur par défaut est le poste de travail sur lequel **composer** est exécuté. Les caractères génériques sont acceptés pour *nom_poste_travail* et *nom_flot_travaux*.

valid from
date Limite la sélection aux flots de travaux qui ont une date *valid from* égale à la valeur indiquée. Le format et *mm/jj/aaaa*.

valid to
date Limite la sélection aux flots de travaux qui ont une date *valid to* égale à la valeur indiquée. Le format et *mm/jj/aaaa*.

valid in
date date Intervalle de temps d'exécution du flot de travaux. Le format est *mm/jj/aaaa - mm/jj/aaaa*. L'une des deux dates peut être représentée par @.

users | user
Si aucun argument ne suit, permet de verrouiller toutes les définitions d'utilisateur.

Si l'argument *nom_poste_travail#nom_utilisateur* suit, permet de verrouiller l'utilisateur *nom_utilisateur* du poste de travail *nom_poste_travail* sur lequel l'utilisateur est défini. Si *nom_poste_travail* est ignoré, la valeur par défaut est le poste de travail sur lequel **composer** est exécuté. Les caractères génériques sont acceptés pour *nom_poste_travail* et *nom_utilisateur*.

Commentaires

Les objets sont verrouillés pour que les définitions de la base de données ne soient pas écrasées par différents utilisateurs qui accèdent simultanément aux mêmes objets.

Avec cette commande, l'utilisateur acquiert de façon explicite les serrures des objets de base de données. Lorsqu'un objet de l'utilisateur est verrouillé, tout autre utilisateur possède le droit d'accès en lecture seule jusqu'à ce que l'objet soit libéré ou déverrouillé de façon explicite par l'administrateur. Si un utilisateur essaie de verrouiller un objet déjà verrouillé par un autre utilisateur, un message d'erreur s'affiche.

Les verrous sur les objets de base de données sont acquis par l'utilisateur en utilisant `username` et `session`, où `session` est une chaîne qui peut être définie dans la variable d'environnement **TWS_SESSION** identifiant cette session utilisateur spécifique.

Ceci signifie que sur un ordinateur, l'identificateur **TWS_SESSION** est différent pour :

- un utilisateur connecté dans deux shells différents sur le programme de ligne de commande **composer**.
- un utilisateur connecté, déconnecté, puis reconnecté à la ligne de commande **composer** à partir du même shell.

Si aucune valeur n'est attribuée à la variable **TWS_SESSION**, la valeur par défaut identifiant la session est définie comme suit :

- Si vous utilisez **composer** en mode de traitement par lots, la valeur par défaut est le *nom_utilisateur* utilisé par l'utilisateur lorsqu'il se connecte au gestionnaire de domaine maître.
- Si vous utilisez **composer** en mode interactif, la valeur par défaut correspond à une chaîne alphanumérique que le produit a automatiquement créée.

Remarque : Dans la base de données, le *nom_utilisateur* de l'utilisateur verrouillant une définition d'objet est enregistré en majuscules.

Exemples

Pour verrouiller l'agenda `Holidays`, exécutez la commande suivante :

```
lock calendar=HOLIDAYS
```

Voir aussi

Dans Dynamic Workload Console, les objets sont automatiquement verrouillés tant que vous, ou un autre utilisateur, ne les aurez pas ouverts à l'aide du bouton **Editer**. Les objets ne sont pas verrouillés si vous, ou un autre utilisateur, les ouvrez avec le bouton **Afficher**.

modify

Permet de modifier ou d'ajouter des objets de planification. Lorsque vous modifiez des objets, la commande **modify** extrait uniquement les objets qui peuvent être verrouillés par l'utilisateur en cours.

Autorisation

Vous devez avoir le droit d'accès **add** pour ajouter un nouvel objet de planification. Si l'objet existe déjà dans la base de données, vous devez avoir le droit d'accès **modify** sur cet objet ; sinon, le composeur ne parvient pas à trouver les objets.

Syntaxe

{**modify** | **m**}
{[**calendars** | **calendar** | **cal**=*nom_agenda*] |
[**eventrule** | **erule** | **er**=*nom_règle_événement*] |
[**parms** | **parm** | **vb**=[*nom_table*.]*nom_variable*] |
[**vartable** | **vt**=*nom_table*] |
[**prompts** | **prom**=*nom_invite*] |

[**resources** | **resource** | **res**=[*poste_de_travail*#]*nom_ressource*] |
[**runcyclegroup** | **rcg**=*nom_groupe_cycle_exécution*] |
[**cpu**={*nom_poste_travail* | *nom_classe_poste_de_travail* | *nom_domaine*}]
[**workstation** | **ws**=*nom_poste_travail*] |
[**workstationclass** | **wscl**=*nom_classe_poste_de_travail*] |
[**domain** | **dom**=*nom_domaine*] |
[**jobs** | **jobdefinition** | **jd**=[*nom_poste_travail*#]*nom_travail*] |
[**sched**|**jobstream**|**js**= [*nom_poste_travail*#]*nom_flot_travaux*
*[**valid from** date|**valid to** date |**valid in** date date]*
*[;**full**]*] |
[**users** | **user**=[*nom_poste_travail*#]*nom_utilisateur*]}

Arguments

calendars | calendar | cal
Si aucun argument ne suit, permet de modifier toutes les définitions d'agenda.

Si l'argument *nom_agenda* suit, permet de modifier l'agenda *nom_agenda*. Les caractères génériques sont acceptés.

eventrule | erule | er
Si aucun argument ne suit, permet de modifier toutes les définitions de règle d'événement.

Si l'argument *nom_règle_événement* suit, permet de modifier la règle d'événement *nom_règle_événement*. Les caractères génériques sont acceptés.

parms|parm|vb
Si aucun argument ne suit, permet de modifier toutes les définitions de variable globale trouvées dans la table de variables par défaut.

Si l'argument *nom_table.nom_variable* suit, permet de modifier la variable spécifiée de la table *nom_table*. Si *nom_table* est ignoré, **composer** recherche la définition de variable *nom_variable* dans la table de variables par défaut. Les caractères génériques sont utilisés dans *nom_table* et *nom_variable*. Par exemple :

```
modify parms=@.@
```

Modifie toutes les variables de toutes les tables.

```
modify parms=@
```

Modifie toutes les variables de la table par défaut.

```
modify parms=@.acct@
```

Modifie toutes les variables dont le nom commence par `acct` dans toutes les tables existantes.

A faire : La modification ou l'ajout d'une variable permet de verrouiller la table de variables qui la contient. Cela implique que, lorsque la table est verrouillée, aucun autre utilisateur ne peut exécuter d'autres commandes de verrouillage sur elle ou sur les variables qu'elle contient.

vartable | vt
Si aucun argument ne suit, permet de modifier toutes les définitions de table de variables.

Si la table de variables *nom_table* de l'argument suit, permet de modifier la table de variables *nom_table*. Les caractères génériques sont acceptés.

prompts | prom
Si aucun argument ne suit, permet de modifier toutes les définitions d'invite.

Si l'argument *nom_invite* suit, permet de modifier l'invite *nom_invite*. Les caractères génériques sont acceptés.

resources | resource | res
Si aucun argument ne suit, permet de modifier toutes les définitions de ressource.

Si l'argument *nom_poste_travail#nom_ressource* suit, permet de modifier la ressource *nom_ressource* du poste de travail *nom_poste_travail* sur lequel la ressource est définie. Si *nom_poste_travail* est ignoré, la valeur par défaut est le poste de travail sur lequel **composer** est exécuté. Les caractères génériques sont acceptés pour *nom_poste_travail* et *nom_ressource*.

runcyclegroup | rcg
Si aucun argument ne suit, permet de modifier toutes les définitions de groupe de cycle d'exécution.

Si l'argument *runcyclegroupname* suit, permet de modifier le groupe de cycle d'exécution *runcyclegroupname*. Les caractères génériques sont acceptés.

cpu Modifie les postes de travail, les classes de postes de travail ou les domaines.

poste de travail
Nom du poste de travail. Les caractères génériques sont acceptés.

workstationclass
Nom de la classe de poste de travail. Les caractères génériques sont acceptés.

domaine
Nom du domaine. Les caractères génériques sont acceptés.

workstation | ws
Si aucun argument ne suit, permet de modifier toutes les définitions de poste de travail.

Si l'argument *nom_poste_travail* suit, permet de modifier le poste de travail *nom_poste_travail*. Les caractères génériques sont acceptés.

domain | dom
Si aucun argument ne suit, permet de modifier toutes les définitions de domaine.

Si l'argument *nom_domaine* suit, permet de modifier le domaine *nom_domaine*. Les caractères génériques sont acceptés.

workstationclass | wscl
Si aucun argument ne suit, permet de modifier toutes les définitions de classe de poste de travail.

Si l'argument *nom_classe_poste_de_travail* suit, permet de modifier la classe de poste de travail *nom_classe_poste_de_travail*. Les caractères génériques sont acceptés.

jobs | jobdefinition | jd
Si aucun argument ne suit, permet de modifier toutes les définitions de travail.

Si l'argument *nom_poste_travail#nom_travail* suit, permet de modifier le travail *nom_travail* du poste de travail *nom_poste_travail* sur lequel le travail s'exécute. Si *nom_poste_travail* est ignoré, la valeur par défaut est le poste de travail sur lequel **composer** est exécuté. Les caractères génériques sont acceptés pour *nom_poste_travail* et *nom_travail*.

sched | jobstream | js
Si aucun argument ne suit, permet de modifier toutes les définitions de flot de travaux.

Si l'argument *nom_poste_travail#nom_flot_travaux* suit, permet de modifier le flot de travaux *nom_flot_travaux* du poste de travail *nom_poste_travail* sur lequel le flot de travaux est défini. Si *nom_poste_travail* est ignoré, la valeur par défaut est le poste de travail sur lequel **composer** est exécuté. Les caractères génériques sont acceptés pour *nom_poste_travail* et *nom_flot_travaux*.

valid from
date Limite la sélection aux flots de travaux qui ont une date *valid from* égale à la valeur indiquée. Le format et *mm/jj/aaaa*.

valid to
date Limite la sélection aux flots de travaux qui ont une date *valid to* égale à la valeur indiquée. Le format et *mm/jj/aaaa*.

valid in
date date Intervalle de temps d'exécution du flot de travaux. Le format est *mm/jj/aaaa - mm/jj/aaaa*. L'une des deux dates peut être représentée par @.

full Modifie toutes les définitions de travail contenues dans le flot de travaux.

users | user
Si aucun argument ne suit, permet de modifier toutes les définitions d'utilisateur.

Si l'argument *nom_poste_travail#nom_utilisateur* suit, permet de modifier l'utilisateur *nom_utilisateur* du poste de travail *nom_poste_travail* sur lequel l'utilisateur est défini. Si *nom_poste_travail* est ignoré, la valeur par défaut est le poste de travail sur lequel **composer** est exécuté. Les caractères génériques sont acceptés pour *nom_poste_travail* et *nom_utilisateur*.

Commentaires

La commande **modify** effectue la séquence d'actions suivante :

1. Verrouille les objets dans la base de données.
2. Copie les définitions d'objet dans un fichier temporaire.
3. Modifie le fichier.
4. Retransmet la définition contenue dans le fichier temporaire dans la base de données.
5. Si la commande **modify** échoue sur un sous-ensemble d'objets sélectionnés, **composer** demande *"Voulez-vous la rééditer ?"* Le fichier enregistré auparavant est rouvert pour être modifié et les étapes suivantes de la séquence sont répétées.
6. Déverrouille les objets dans la base de données.

Les définitions de règle d'événement sont ouvertes avec un éditeur XML (voir «Définition de règle d'événement», à la page 284 pour la référence XML et voir «Editeur Composer», à la page 302 pour des détails sur la configuration d'un éditeur XML).

Si vous modifiez avec la même commande **modify** deux objets ou plus liés par une relation (par exemple un travail remplaçant et le travail remplacé), l'ordre dans lequel les objets sont répertoriés dans le fichier temporaire a parfois son importance pour la réussite de la commande **modify**. Ceci se produit parce que la commande **modify** lit en séquence les objets contenus dans le fichier temporaire ; par conséquent si l'objet référençant est affiché avant l'objet référencé la commande modify peut échouer sur l'objet référençant.

Par exemple, si la commande :

```
modify FTA1#@PROVA
```

produit le fichier temporaire suivant :

```
SCHEDULE FTA1#PROVA VALIDFROM 08/31/2005
MATCHING SAMEDAY
:
FTA2#MY-JOB
 FOLLOWS FTA1#COPYOFPROVA.MY-JOB06
END

SCHEDULE FTA1#COPYOFPROVA VALIDFROM 08/31/2005
MATCHING SAMEDAY
:
FTA1#MY-JOBO6
END
```

et vous changez le nom du travail prédécesseur de `FTA1#MY-JOB06` à `FTA1#MY-JOB05`dans les deux flots de travaux `FTA1#PROVA` et `FTA1#COPYOFPROVA`, puis la commande **modify** :

1. essaie d'abord de changer la définition du flot de travaux `FTA1#PROVA`. Elle échoue parce qu'elle trouve une dépendance de prédécesseur/successeur provenant d'un travail `FTA1#MY-JOB05` qui est encore inconnu.
2. elle essaie ensuite de modifier la définition de `FTA1#COPYOFPROVA` et elle réussit.

A la seconde exécution de **modify** pour faire passer le nom du travail précédent de `FTA1#MY-JOB06` à `FTA1#MY-JOB05` dans le flot de travaux `FTA1#PROVA`, la commande est exécutée correctement car le travail précédent `FTA1#MY-JOB05` existe maintenant dans la base de données.

Si le flot de travaux `FTA1#COPYOFPROVA` a été répertorié dans le fichier temporaire avant `FTA1#PROVA`, alors la commande **modify** s'exécute correctement la première fois parce que le nom du travail prédécesseur a été modifié avant de changer la définition de dépendance dans le travail successeur.

Pour les définitions d'utilisateur, si la zone de mot de passe conserve la valeur "*******" lorsque vous quittez l'éditeur, l'ancien mot de passe est conservé. Pour définir un mot de passe vide, insérez deux guillemets ("").

La commande modify vérifie la présence de dépendances de boucle à l'intérieur des flots de travaux. Par exemple, si `job1` suit `job2` et si `job2` suit `job1`, il dépendance de boucle. Lorsqu'une dépendance de boucle est détectée à l'intérieur d'un flot de travaux, une erreur est affichée. La commande modify ne vérifie pas les dépendances de boucle entre les flots de travaux parce que selon la complexité

des activités de planification, cette vérification pourrait prendre trop de temps et de ressources d'UC.

Exemples

Pour modifier tous les agendas, exécutez la commande suivante :

```
modify calendars=@
```

Pour modifier le flot de travaux `sked9` qui a été lancé sur le poste de travail `site1`, exécutez la commande suivante :

```
m sched=site1#sked9
```

Pour modifier toutes les règles d'événement comprenant une action appliquée à l'objet `DPJOB10`, exécutez la commande :

```
mod er=@;filter job=DPJOB10
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer les mêmes tâches que celles décrites dans :

le manuel Dynamic Workload Console - Guide d'utilisation.

- Pour afficher, éditer ou supprimer des postes de travail, voir
  le manuel Dynamic Workload Console - Guide d'utilisation, section Edition de définitions de poste de travail.
- Pour afficher, éditer ou supprimer des règles d'événement, voir
  le manuel Dynamic Workload Console - Guide d'utilisation, section Edition d'une règle d'événement.
- Pour afficher, éditer ou supprimer tous les autres objets, voir
  le manuel Dynamic Workload Console - Guide d'utilisation, section Liste des définitions d'objet dans la base de données.

new

Permet d'ajouter une nouvelle définition d'objet dans la base de données.

Autorisation

Vous devez avoir le droit d'accès ***add*** pour ajouter un nouvel objet de planification. Si l'objet existe déjà dans la base de données, vous devez avoir le droit d'accès ***modify*** à l'objet.

Syntaxe

new
[calendar |
domain |
eventrule |
job |
jobstream |
parameter |
prompt |
resource |
runcyclegroup |

user |
vartable |
workstation |
workstation_class]

Arguments

L'objet que vous souhaitez définir : un agenda, un domaine, une règle d'événement, un travail, un flot de travaux, une variable, une invite, une ressource, un utilisateur, une table de variables, un poste de travail ou une classe de poste de travail.

Commentaires

La commande ouvre un modèle prédéfini qui vous aide à modifier la définition d'objets et l'ajoute à la base de données lorsque vous l'enregistrez.

Les modèles d'objets sont situés dans le sous-dossier `templates` du répertoire d'installation Tivoli Workload Scheduler. Ils peuvent être personnalisés pour tenir compte de vos préférences.

Les définitions de règle d'événement sont ouvertes avec un éditeur XML (voir «Définition de règle d'événement», à la page 284 pour la référence XML et voir «Editeur Composer», à la page 302 pour des détails sur la configuration d'un éditeur XML).

Lors de la création d'une variable, la table de variables de destination est verrouillée. Cela implique que, lorsque la table est verrouillée, aucun autre utilisateur ne peut exécuter d'autres commandes de verrouillage sur elle.

Exemples

Pour créer une nouvelle définition d'utilisateurs, exécutez la commande :

```
new user
```

Pour créer une nouvelle définition d'invite, exécutez la commande :

```
new prompt
```

Pour créer une nouvelle définition de règle d'événement, exécutez la commande :

```
new erule
```

Pour créer une définition de table de variables, exécutez la commande suivante :

```
new vartable
```

Pour créer une définition de variable, exécutez la commande suivante :

```
new parameter
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer les mêmes tâches que celles décrites dans :

le manuel Dynamic Workload Console - Guide d'utilisation.

- Pour créer des postes de travail, voir

le manuel Dynamic Workload Console - Guide d'utilisation, section "Création de postes de travail distribués".

- Pour créer des règles d'événement, voir

  le manuel Dynamic Workload Console - Guide d'utilisation, section Création d'une règle d'événement.
- Pour créer tous les autres objets, voir

  le manuel Dynamic Workload Console - Guide d'utilisation, section "Conception de votre charge de travail".

print

Il s'agit d'un synonyme de la commande **list**. Voir «list», à la page 335 pour plus de détails.

redo

Modifie et exécute à nouveau la commande précédente.

Remarque : Si la commande précédente était **authenticate**, la commande **redo** n'affiche pas le mot de passe défini.

Autorisation

Tout utilisateur autorisé à exécuter composer est autorisé à émettre cette commande.

Syntaxe

{**redo** | **red**}

Contexte

Si vous exécutez la commande **redo**, le programme composer affiche la dernière commande que vous pouvez modifier et réexécuter à votre guise. Utilisez la barre d'espacement pour placer le curseur sous le caractère à modifier, puis entrez les instructions ci-dessous.

Instructions

d[*rép***]** Supprime le caractère situé au-dessus de la lettre **d**. Cette commande peut être suivie par d'autres instructions.

i*texte* Insère du texte avant le caractère situé au-dessus de la lettre **i**.

r*texte* Remplace un ou plusieurs caractères par du texte en commençant par le caractère au-dessus du **r**. Le remplacement est implicite si aucune autre instruction n'est entrée.

>*texte* Ajoute du texte à la fin de la ligne.

>d[*rép* | *texte* **]**
Supprime les caractères en fin de ligne. Cette commande peut être suivie d'une autre instruction ou d'un autre texte.

>r*texte* Remplace des caractères à la fin de la ligne par du texte.

Exemples d'instruction

ddd Supprime les trois caractères au-dessus des **d**.

iabc Insère **abc** avant le caractère situé au-dessus de la lettre **i**.

rabc Remplace les trois caractères, en commençant par celui qui est situé au-dessus du **r**, par **abc**.

abc Remplace les trois caractères situés au-dessus de **abc**, par **abc**.

d diabc
Supprime le caractère au-dessus du premier **d**, ignore un caractère, supprime le caractère au-dessus du second **d**, puis le remplace par **abc**.

>abc Ajoute **abc** à la fin de la ligne.

>ddabc
Supprime les deux derniers caractères de la ligne, puis les remplace par **abc**.

>rabc Remplace les trois derniers caractères de la ligne par **abc**.

Exemples

Pour insérer un caractère, exécutez la commande suivante :

```
redo
dislay site1#sa@
   ip
display site1#sa@
```

Pour remplacer trois caractères, par exemple, pour remplacer `site` par `serv` en remplaçant `ite` par `erv`, exécutez la commande suivante :

```
redo
display site1#sa@
         rerv
display serv1#sa@
```

rename

Permet de renommer un objet de planification existant dans la base de données. Le nouveau nom ne doit pas identifier un objet déjà défini dans la base de données.

Autorisation

Vous devez avoir le droit d'accès ***delete*** à l'objet avec l'ancien nom ainsi qu'un droit d'accès ***add*** à l'objet avec le nouveau nom.

Syntaxe

{**rename** | **rn**}
{**calendars** | **calendar** | **cal** |
parms | **parm** | **vb** |
vartable | **vt** |
prompts | **prom** |
resorces | **resource** | **res** |
runcyclegroup | **rcg** |
workstation | **ws** |
workstationclass | **wscl** |
domain | **dom** |
jobs | **jobdefinition** | **jd** |
jobsched | **jb** |
eventrule | **erule** | **er**
sched | **jobstream** | **js** |

users | user }
ancien_identificateur_objet nouvel_identificateur_objet

Arguments

ancien_identificateur_objet
Spécifie l'ancienne clé externe qui identifie l'objet de planification, par exemple le nom d'agenda `cal1` comme identificateur d'un objet d'agenda défini pour être renommé.

nouvel_identificateur_objet
Spécifie la nouvelle clé externe qui identifie l'objet de planification, par exemple le nom d'agenda `cal2` comme le nouvel identificateur à attribuer à l'objet d'agenda auparavant nommé `cal1`.

En ce qui concerne les travaux, flots de travaux, ressources et utilisateurs, les valeurs ***ancien_identificateur_objet*** et ***nouvel_identificateur_objet*** possèdent les formats suivants :

[*nom_poste_travail***#]***nom_travail*
La commande s'applique à cette définition de travail. Ce format est utilisé avec la clé **jobs | jobdefinition | jd**.

[*nom_poste_travail***#]***nom_flot_travaux*
La commande s'applique à toutes les versions de ce flot de travaux. Ce format est utilisé avec la clé **sched | jobstream | js**.

[*nom_poste_travail***#]***nom_flot_travaux***+valid from** *date*
La commande s'applique uniquement à cette version du flot de travaux. Ce format est utilisé avec la clé **sched | jobstream | js**.

[*nom_poste_travail***#]***nom_flot_travaux.nom_travail*
La commande s'applique à cette instance de travail définie dans ce flot de travaux. Voir le mot-clé **js** dans la syntaxe «Définition de flot de travaux», à la page 236 pour des détails supplémentaires. Ce format est utilisé avec la clé **jobsched | jb**.

[*nom_poste_travail***#]***nom_ressource*
La commande s'applique à cette définition de ressource. Ce format est utilisé avec la clé **resources | resource | res**.

[*nom_poste_travail***#][***domaine***]***nom_utilisateur*
La commande s'applique à cette définition d'utilisateur. Ce format est utilisé avec la clé **users | user**.

En ce qui concerne les variables (paramètres globaux) :

ancien_identificateur_objet
Doit être spécifié au format *nom_table.nom_variable*. Si *nom_table* est ignoré, composer recherche la définition de variable dans la table de variables par défaut.

nouvel_identificateur_objet
Doit être spécifié au format *nom_variable*. L'ajout du nom de table ici génère une erreur.

Commentaires

Pour pouvoir être renommé, l'objet doit être déverrouillé ou verrouillé par l'utilisateur qui exécute la commande rename.

La table de variables contenant la variable est verrouillée, alors que la variable est renommée. Cela implique que, lorsque la table est verrouillée, aucun autre utilisateur ne peut exécuter d'autres commandes de verrouillage sur elle.

Si l'objet nommé comme dans la zone *ancien_identificateur_objet* n'existe pas dans la base de données, un message d'erreur s'affiche.

L'utilisation de caractères génériques n'est pas autorisée avec cette commande.

Lorsque ***nom_poste_travail*** n'est pas spécifié pour les objets qui ont le nom du poste de travail dans leur identificateur d'objet (par exemple définitions de travail ou de flot de travaux), le planificateur utilise l'un des éléments suivants pour ***nom_poste_travail*** :

- Le poste de travail par défaut spécifié dans le fichier *localopts*
- Le gestionnaire de domaine maître si le programme de ligne de commande **composer** s'exécute sur un noeud à l'extérieur du réseau Tivoli Workload Scheduler. Dans ce cas en effet, le poste de travail par défaut défini dans le fichier *localopts* est le gestionnaire de domaine maître.

La commande **rename** est utilisée pour attribuer les nouveaux noms aux objets déjà existants dans la base de données. Le nouveau nom attribué à l'objet est immédiatement utilisé dans la base de données alors qu'il est utilisé dans le plan après la réexécution du script **JnextPlan**. Cela peut entraîner des absurdités lors de la soumission des travaux adhoc avant la régénération du plan de production.

Exemples

Pour remplacer le nom de l'objet de domaine `DOMAIN1` par `DOMAIN2`, exécutez la commande suivante :

```
rename dom=DOMAIN1 DOMAIN2
```

Pour renommer le flot de travaux de `LABJST1` en `LABJST2` sur le poste de travail `CPU1`, exécutez la commande suivante :

```
rename js=CPU1#LABJST1 CPU1#LABJST2
```

Pour renommer la variable `ACCTOLD` (définie dans la table `ACCTAB`) en `ACCTNEW`, exécutez la commande suivante :

```
rename parm=ACCTAB.ACCTOLD ACCTNEW
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer les mêmes tâches que celles décrites dans :

le manuel Dynamic Workload Console - Guide d'utilisation.

- Pour afficher, éditer ou supprimer des postes de travail, voir

  le manuel Dynamic Workload Console - Guide d'utilisation, section Edition de définitions de poste de travail.
- Pour afficher, éditer ou supprimer des règles d'événement, voir

  le manuel Dynamic Workload Console - Guide d'utilisation, section Edition d'une règle d'événement.
- Pour afficher, éditer ou supprimer tous les autres objets, voir

le manuel Dynamic Workload Console - Guide d'utilisation, section Liste des définitions d'objet dans la base de données.

replace

Remplace les définitions d'objets de planification dans la base de données.

Autorisation

Vous devez avoir le droit d'accès ***add*** pour ajouter un nouvel objet de planification. Si l'objet existe déjà dans la base de données, vous devez avoir les droits suivants :

- droit d'accès ***modify*** sur l'objet si ce dernier n'est pas verrouillé.
- droits d'accès ***modify*** et ***unlock*** aux objets si vous souhaitez utiliser l'option **;unlock** sur des objets verrouillés par d'autres utilisateurs.

Syntaxe

{**replace** | **rep**} *file_name* [**;unlock**]

Arguments

file_name
Indique le nom d'un fichier qui contient les définitions d'objet à remplacer. Le fichier peut contenir tous les types de définition d'objet de planification.

unlock
Permet de mettre à jour les objets auparavant verrouillés, puis de les déverrouiller. Une erreur s'affiche si les objets n'étaient pas auparavant verrouillés. Pour tous les nouveaux objets insérés, cette option est ignorée.

Commentaires

La commande **replace** ressemble à la commande **add** mais elle n'affiche aucune invite de confirmation du remplacement des objets existants. Pour plus d'informations, voir «add», à la page 316.

La commande replace vérifie la présence de dépendances de boucle à l'intérieur des flots de travaux. Par exemple, si `job1` suit `job2` et si `job2` suit `job1`, il dépendance de boucle. Lorsqu'une dépendance de boucle est détectée à l'intérieur d'un flot de travaux, une erreur est affichée. La commande replace ne vérifie pas les dépendances de boucle entre les flots de travaux parce que selon la complexité des activités de planification, cette vérification pourrait prendre trop de temps et de ressources d'UC.

Exemples

Pour remplacer les travaux du fichier `myjobs`, exécutez la commande suivante :

```
replace myjobs
```

Pour remplacer toutes les ressources par celles du fichier `myres`, exécutez la commande suivante :

```
rep myres
```

Vous souhaitez modifier certaines définitions de règles d'événement existantes dans la base de données. Vous souhaitez également ajouter certaines nouvelles définitions. Vous utilisez cette commande de la façon suivante :

1. Vous écrivez les définitions complètes dans un fichier XML que vous nommez `2Q07rules.xml`.
2. Vous exécutez :

   ```
   rep 2Q07rules.xml
   ```

Commande système

Exécute une commande système.

Syntaxe

[**:** | **!**] *commande système*

Arguments

commande système
: Désigne une commande système valide. Le préfixe deux-points (:) ou le point d'exclamation (!) sont requis uniquement lorsque la commande est orthographiée comme une commande Composer.

Exemples

Pour exécuter une commande **ps** sous UNIX, exécutez la commande suivante :

```
ps -ef
```

Pour exécuter une commande **dir** sous Windows, exécutez la commande suivante :

```
dir \bin
```

unlock

Permet de déverrouiller l'accès aux objets de planification dans la base de données. Par défaut, pour déverrouiller un objet, ce dernier doit avoir été verrouillé par le même utilisateur lors de la même session.

Autorisation

Vous devez avoir le droit d'accès ***unlock*** pour déverrouiller les objets verrouillés par d'autres utilisateurs.

Syntaxe

{**unlock** | **u**}
{[**calendars** | **calendar** | **cal**=*nom_agenda*] |
[**eventrule** | **erule** | **er**=*nom_règle_événement*] |
[**parms** | **parm** | **vb**=[*nom_table.*]*nom_variable*] |
[**vartable** | **vt**=*nom_table*] |
[**prompts** | **prom**=*nom_invite*] |
[**resources** | **resource** | **res**=[*poste_de_travail*#]*nom_ressource*] |
[**runcyclegroup** | **rcg**=*nom_groupe_cycle_exécution*] |
[**cpu**={*nom_poste_travail* | *nom_classe_poste_de_travail* | *nom_domaine*}]
[**workstation** | **ws**=*nom_poste_travail*] |
[**workstationclass** | **wscl**=*nom_classe_poste_de_travail*] |
[**domain** | **dom**=*nom_domaine*] |
[**jobs** | **jobdefinition** | **jd**=[*nom_poste_travail*#]*nom_travail*] |
[**sched**|**jobstream**|**js**= [*nom_poste_travail*#]*nom_flot_travaux*
*[**début de validité** date|**fin de validité** date |**valid***

***in** date date*]] |
[**users** | **user**=[*nom_poste_travail*#]*nom_utilisateur*]}
[**;forced**]

Arguments

calendars | calendar | cal
Si aucun argument ne suit, permet de déverrouiller toutes les définitions d'agenda.

Si l'argument *nom_agenda* suit, permet de déverrouiller l'agenda *nom_agenda*. Les caractères génériques sont acceptés.

eventrule | erule | er
Si aucun argument ne suit, permet de déverrouiller toutes les définitions de règle d'événement.

Si l'argument *nom_règle_événement* suit, permet de déverrouiller la règle d'événement *nom_règle_événement*. Les caractères génériques sont acceptés.

parms | parm | vb
Si aucun argument ne suit, permet de déverrouiller la table de variables par défaut.

Si l'argument *nom_table.nom_variable* suit, permet de déverrouiller toute la table contenant la variable *nom_variable*. Si *nom_table* est ignoré, permet de déverrouiller la table de variables par défaut. Les caractères génériques sont utilisés dans *nom_table* et *nom_variable*. Par exemple :

```
unlock parms=@.@
```

Déverrouille toutes les tables.

```
unlock parms=@
```

Déverrouille la table par défaut.

```
unlock parms=@.acct@
```

Permet de déverrouiller toutes les tables contenant les variables dont le nom commence par `acct`.

```
unlock parms=acct@
```

Déverrouille la table par défaut.

A faire : L'action effectuée sur une seule variable permet de déverrouiller la table de variables la contenant.

vartable | vt
Si aucun argument ne suit, permet de déverrouiller toutes les définitions de table de variables.

Si la table de variables *nom_table* de l'argument suit, permet de déverrouiller la table de variables *nom_table*. Les caractères génériques sont acceptés.

prompts | prom
Si aucun argument ne suit, permet de déverrouiller toutes les définitions d'invite.

Si l'argument *nom_invite* suit, permet de déverrouiller l'invite *nom_invite*. Les caractères génériques sont acceptés.

resources | resource | res
Si aucun argument ne suit, permet de déverrouiller toutes les définitions de ressource.

Si l'argument *nom_poste_travail#nom_ressource* suit, permet de déverrouiller la ressource *nom_ressource* du poste de travail *nom_poste_travail* sur lequel la ressource est définie. Si *nom_poste_travail* est ignoré, la valeur par défaut est le poste de travail sur lequel **composer** est exécuté. Les caractères génériques sont acceptés pour *nom_poste_travail* et *nom_ressource*.

runcyclegroup | rcg
Si aucun argument ne suit, permet de déverrouiller toutes les définitions de groupe de cycle d'exécution.

Si l'argument *runcyclegroupname* suit, permet de déverrouiller le groupe de cycle d'exécution *runcyclegroupname*. Les caractères génériques sont acceptés.

cpu Déverrouille les postes de travail, les classes de postes de travail ou les domaines.

poste de travail
Nom du poste de travail. Les caractères génériques sont acceptés.

workstationclass
Nom de la classe de poste de travail. Les caractères génériques sont acceptés.

domaine
Nom du domaine. Les caractères génériques sont acceptés.

workstation | ws
Si aucun argument ne suit, permet de déverrouiller toutes les définitions de poste de travail.

Si l'argument *nom_poste_travail* suit, permet de déverrouiller le poste de travail *nom_poste_travail*. Les caractères génériques sont acceptés.

domain | dom
Si aucun argument ne suit, permet de déverrouiller toutes les définitions de domaine.

Si l'argument *nom_domaine* suit, permet de déverrouiller le domaine *nom_domaine*. Les caractères génériques sont acceptés.

workstationclass | wscl
Si aucun argument ne suit, permet de déverrouiller toutes les définitions de classe de poste de travail.

Si l'argument *nom_classe_poste_de_travail* suit, permet de déverrouiller la classe de poste de travail *nom_classe_poste_de_travail*. Les caractères génériques sont acceptés.

jobs | jobdefinition | jd
Si aucun argument ne suit, permet de déverrouiller toutes les définitions de travail.

Si l'argument *nom_poste_travail#nom_travail* suit, permet de déverrouiller le travail *nom_travail* du poste de travail *nom_poste_travail* sur lequel le travail s'exécute. Si *nom_poste_travail* est ignoré, la valeur par défaut est le poste de travail sur lequel **composer** est exécuté. Les caractères génériques sont acceptés pour *nom_poste_travail* et *nom_travail*.

sched | jobstream | js
Si aucun argument ne suit, permet de déverrouiller toutes les définitions de flot de travaux.

Si l'argument *nom_poste_travail#nom_flot_travaux* suit, permet de déverrouiller le flot de travaux *nom_flot_travaux* du poste de travail *nom_poste_travail* sur lequel le flot de travaux est défini. Si *nom_poste_travail* est ignoré, la valeur par défaut est le poste de travail sur lequel **composer** est exécuté. Les caractères génériques sont acceptés pour *nom_poste_travail* et *nom_flot_travaux*.

valid from
date Limite la sélection aux flots de travaux qui ont une date *valid from* égale à la valeur indiquée. Le format et *mm/jj/aaaa*.

valid to
date Limite la sélection aux flots de travaux qui ont une date *valid to* égale à la valeur indiquée. Le format et *mm/jj/aaaa*.

valid in
date date Intervalle de temps d'exécution du flot de travaux. Le format est *mm/jj/aaaa - mm/jj/aaaa*. L'une des deux dates peut être représentée par @.

users | user
Si aucun argument ne suit, permet de déverrouiller toutes les définitions d'utilisateur.

Si l'argument *nom_poste_travail#nom_utilisateur* suit, permet de déverrouiller l'utilisateur *nom_utilisateur* du poste de travail *nom_poste_travail* sur lequel l'utilisateur est défini. Si *nom_poste_travail* est ignoré, la valeur par défaut est le poste de travail sur lequel **composer** est exécuté. Les caractères génériques sont acceptés pour *nom_poste_travail* et *nom_utilisateur*.

forced Autorise l'utilisateur qui a verrouillé l'objet à le déverrouiller, quelle que soit la session.

Si cette option est utilisée par le *superutilisateur*, la commande **unlock** peut fonctionner quels que soient l'utilisateur et la session utilisés pour verrouiller l'objet.

Commentaires

Si un utilisateur autre que le *superutilisateur* essaie de déverrouiller un objet verrouillé par un autre utilisateur, un message d'erreur s'affiche.

Exemples

Pour déverrouiller la définition de travail `JOBDEF1`, exécutez la commande suivante :

```
unlock jd=@#JOBDEF1
```

Pour déverrouiller la définition de règle d'événement `ERJS21`, exécutez la commande suivante :

```
unlock erule=ERJS21
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer les mêmes tâches que celles décrites dans :

le manuel Dynamic Workload Console - Guide d'utilisation.

- Pour afficher, éditer ou supprimer des postes de travail, voir

  le manuel Dynamic Workload Console - Guide d'utilisation, section Edition de définitions de poste de travail.
- Pour afficher, éditer ou supprimer des règles d'événement, voir

  le manuel Dynamic Workload Console - Guide d'utilisation, section Edition d'une règle d'événement.
- Pour afficher, éditer ou supprimer tous les autres objets, voir

  le manuel Dynamic Workload Console - Guide d'utilisation, section Liste des définitions d'objet dans la base de données.

validate

Exécute la validation des définitions d'objets contenus dans un fichier utilisateur.

Autorisation

Vous n'avez pas besoin d'une autorisation spécifique sur les objets pour exécuter cette commande.

Syntaxe

{**validate** | **val**} *file_name* [**;syntax**]

Arguments

file_name
: Désigne le nom d'un fichier qui contient des agendas, des postes de travail, des classes de postes de travail, des domaines, des travaux, des paramètres, des invites, des ressources, des flots de travaux, des règles d'événement ou des tables de variables. Pour les définitions de règles d'événement, le fichier doit être en XML. Voir «Définition de règle d'événement», à la page 284 pour plus de détails sur l'écriture des définitions de règles d'événement.

syntaxe
: Détecte les erreurs de syntaxe dans le fichier.

Commentaires

La sortie de la commande **validate** peut être redirigée vers un fichier de la manière suivante :

```
composer "validate filename" > outfile
```

Pour insérer des messages d'erreur dans le fichier de sortie, utilisez la commande suivante :

```
composer "validate filename" > outfile 2>&1
```

Exemples

Pour vérifier la syntaxe d'un fichier contenant les définitions de poste de travail, exécutez la commande suivante :

```
validate mycpus;syntax
```

version

Affiche la bannière du programme de ligne de commande **composer**.

Autorisation

Tout utilisateur autorisé à exécuter composer est autorisé à émettre cette commande.

Syntaxe

{**version** | **v**}

Exemples

Pour afficher la bannière du programme de ligne de commande **composer**, exécutez la commande suivante :

```
version
```

ou :

```
v
```

Chapitre 10. Gestion des applications de charge de travail

Les applications de charge de travail peuvent être créées puis exportées de sorte à pouvoir être partagées avec d'autres environnements Tivoli Workload Scheduler. Dans le nouvel environnement, l'application de charge de travail peut être mise à jour, remplacée ou supprimée par la suite.

Le cycle de vie d'une application de charge de travail commence avec la définition du modèle d'application de charge de travail. Le modèle est ensuite exporté puis, avant de pouvoir être déployé dans l'environnement cible, une personnalisation manuelle est requise.

Le processus d'exportation génère un fichier compressé contenant trois fichiers :

*nom de l'application de charge de travail*_**Definitions.UTF8.xml**
: Un fichier au format XML contenant une définition de tous les objets référencés dans l'application de charge de travail. Les définitions sont déployées dans l'environnement cible pour remplir la base de données cible avec les mêmes objets qui existent dans l'environnement source. N'éditez pas ce fichier.

*nom de l'application de charge de travail*_**Mapping.UTF8.properties**
: Fichier de mappage modifié par l'utilisateur cible, qui remplace les noms des objets dans l'environnement source par les futurs noms des objets dans l'environnement cible.

*nom de l'application de charge de travail*_**SourceEnv_reference.txt**
: Fichier contenant des informations de référence sur les postes de travail utilisés dans l'application de charge de travail ainsi que d'autres informations pouvant être utiles pour mapper correctement l'environnement source à l'environnement cible.

Pour utiliser l'application de charge de travail dans un nouvel environnement, modifiez le fichier de mappage de sorte à refléter l'environnement de destination à l'aide des informations fournies dans le fichier de référence, puis exécutez une opération d'importation. L'opération d'importation est réalisée en transmettant le fichier de mappage et le fichier de définition sous la forme d'une entrée à la commande wappman.

La commande wappman peut être utilisée pour importer, remplacer, répertorier, afficher et supprimer une application de charge de travail. Voir «Commande wappman», à la page 369 pour obtenir la syntaxe et l'utilisation complète de la ligne de commande afin de réaliser ces actions sur une application de charge de travail et pour prendre connaissance des éventuelles remarques associées.

Résolution du fichier de mappage

Le fichier de mappage généré par le processus d'exportation d'une application de charge de travail, contient une liste d'éléments, dont certains sont dépendants de la topologie de l'environnement dans lequel ils sont utilisés. Ces éléments doivent être personnalisés pour refléter l'environnement cible.

Au cours du processus d'exportation, les objets contenus dans l'application de charge de travail sont extraits dans le fichier de définitions avec la même définition que dans l'environnement source. Le fichier de définitions peut contenir une

définition d'objet complète, ou dans certains cas, un nom ou une référence à l'objet extrait uniquement. Des références simples et une définition d'objet incomplète sont extraites pour ces objets qui doivent être mappés à un objet déjà présent dans l'environnement cible. Pour certains objets extraits par référence, la définition d'objet est écrite dans le fichier de mappage qui nécessite une personnalisation manuelle afin de mapper les objets dans l'environnement source Tivoli Workload Scheduler à l'environnement dans lequel l'application de charge de travail sera déployée.

Le fichier de mappage peut être consulté et édité à l'aide d'un éditeur de texte. Il est organisé en sections et contient des commentaires permettant de vous aider à attribuer les valeurs correctes aux éléments.

Le tableau ci-dessous présente tous les objets qui peuvent se trouver dans une application de charge de travail ou qui sont référencés par un autre élément dans l'application de charge de travail et la manière dont le processus d'exportation les gère.

Tableau 62. Objets extraits au cours du processus d'exportation

Type d'objet	**Quels éléments sont exportés vers le fichier de définitions ?**	**Quels éléments nécessitent d'être personnalisés dans le fichier de mappage ?**	**Que se passe-t-il au cours de l'importation ?**
Flot de travaux	Définition d'objet	Poste de travail Nom du flot de travaux Alias du travail	Un flot de travaux est créé dans la base de données. **Restriction :** Si le flot de travaux possède une dépendance à un flot de travaux ou un travail externe à l'application de charge de travail, le fichier de mappage contient une référence au nom du flot de travaux ou du travail externe et à la définition de poste de travail relative. Toutefois, le fichier de définitions ne contient pas la définition de flot de travaux ou de travail. Le nom du fichier de mappage doit être mappé à un flot de travaux ou un travail existant dans l'environnement cible afin d'importer correctement l'application de charge de travail.

Tableau 62. Objets extraits au cours du processus d'exportation (suite)

Type d'objet	Quels éléments sont exportés vers le fichier de définitions ?	Quels éléments nécessitent d'être personnalisés dans le fichier de mappage ?	Que se passe-t-il au cours de l'importation ?
Travail	Définition d'objet	Poste de travail Nom du travail Affinité Variables de mot de passe trouvés dans JSDL	Un travail est créé dans la base de données. **Restriction :** Si le travail possède une dépendance à un travail qui est défini dans un flot de travaux externe à l'application de charge de travail, le fichier de mappage contient une référence aux objets suivants : nom du flot de travaux externe, travail défini dans le flot de travaux externe, définitions de postes de travail dans le flot de travaux et définitions de poste de travail du travail. Cependant, le fichier de définitions ne contient pas la définition du travail. Le nom du fichier de mappage doit être mappé à un flot de travaux ou un travail existant dans l'environnement cible afin d'importer correctement l'application de charge de travail. Les relations d'affinité déclenchent l'exécution de travaux sur le même poste de travail. Le poste de travail sur lequel le premier travail s'exécute est choisi de façon dynamique, le travail ou les travaux d'affinité s'exécutant sur le même poste de travail. Les travaux doivent appartenir au même flot de travaux. Lorsqu'un travail avec une affinité est exporté, le nom du travail est ajouté au fichier de mappage. Les variables du JSDL utilisent le format ${password:ws#user}. Seuls les postes de travail sont représentés de façon générique. La zone de l'utilisateur est copiée telle quelle dans l'environnement cible. Les variables doivent être utilisées pour les noms d'utilisateur.
Tables de variable de cycle d'exécution Tables de variable du flot de travaux	Définition d'objet	Nom de la table Nom de la variable	Une table de variable est créée dans la base de données. Les tables de poste de travail et de variable par défaut sont extraites par référence et écrites dans le fichier de mappage. La valeur associée à la variable peut être modifiée, pas les noms de variable. Evitez d'associer la table de variable par défaut aux flots de travaux et aux cycles d'exécution.
Utilisateur	Aucune action. Aucune définition d'objet ou référence n'est établie dans le fichier de référence.		L'utilisateur doit exister dans l'environnement cible. Des variables doivent être utilisées comme références aux utilisateurs afin de rendre l'application de charge de travail plus flexible pour les futures utilisations.

Tableau 62. Objets extraits au cours du processus d'exportation (suite)

Type d'objet	Quels éléments sont exportés vers le fichier de définitions ?	Quels éléments nécessitent d'être personnalisés dans le fichier de mappage ?	Que se passe-t-il au cours de l'importation ?
Agenda	Définition d'objet	Nom de l'agenda	Un agenda est créé dans la base de données
Cycle d'exécution	Définition d'objet	Non de cycle d'exécution	Un cycle d'exécution est créé dans la base de données
Groupe de cycle d'exécution	Définition d'objet	Nom du groupe de cycle d'exécution	Un groupe de cycle d'exécution est créé dans la base de données
Dépendances interréseaux	Le travail ou le flot de travaux référencé n'est pas exporté car il appartient à un moteur différent.	Nom du poste de travail de l'agent de réseau qui gère les dépendances de prédécesseur/successeur entre le réseau local et le réseau distant.	Une dépendance interréseau est ajoutée au travail ou au flot de travaux.
Dépendances externes	Le travail ou le flot de travaux référencé est exporté uniquement s'il appartient au modèle d'application de charge de travail.	Si le travail ou le flot de travaux référencé n'appartient pas au modèle d'application de charge de travail, attribuez un nom au travail ou au flot de travaux qui correspond à un travail ou un flot de travaux qui existe déjà dans l'environnement cible.	Une dépendance externe est ajoutée au travail ou au flot de travaux
Ressources	Définition d'objet	Poste de travail Nom de la ressource	Des ressources sont créées dans la base de données
Invites globales	Définition d'objet	Variables utilisées dans la définition	Des invites globales sont créées dans la base de données. Vous pouvez utiliser des variables. Etant donné qu'elles sont résolues à l'aide de la table par défaut, la variable utilisée dans une invite globale peut être mappée à une variable dans l'environnement cible.
Postes de travail	Référence uniquement	Nom	Non importés. Les postes de travail sont extraits dans le fichier de définition sous la forme d'une référence. Une définition n'est pas importée car les postes de travail sont déjà définis dans l'environnement cible. Toutefois, leurs noms n'ont pas besoin d'être mappés.
Classe de poste de travail	Référence uniquement	Nom	Non importée. Les classes de poste de travail sont extraites dans le fichier de définition sous la forme d'une référence. Une définition n'est pas importée car les classes de poste de travail sont déjà définies dans l'environnement cible. Toutefois, leurs noms n'ont pas besoin d'être mappés.

Tableau 62. Objets extraits au cours du processus d'exportation (suite)

Type d'objet	Quels éléments sont exportés vers le fichier de définitions ?	Quels éléments nécessitent d'être personnalisés dans le fichier de mappage ?	Que se passe-t-il au cours de l'importation ?
Variable	Définition d'objet	Valeur	Importée. Les variables sont utilisées à plusieurs emplacements dans une définition de flot de travaux. Une référence est ajoutée au fichier de définitions.

Exemple

L'exemple ci-dessous décrit comment les informations des fichiers contenus dans le fichier compressé créé par l'exportation de l'application de charge de travail à partir de Dynamic Workload Console sont utilisé pour préparer les fichiers au déploiement dans l'environnement cible.

Tableau 63. Résolution du fichier de mappage

Fichier de définitions

```
<model:JobStream carryforward=
  "true" draft="false"
 iskey="false" limit="55" name=
  "$JOBSTREAM_BADPBN34_JS1_1I$"
 onrequest="false" priority="100"
 workstation="$WORKSTATION_
  BADPBN34_WC1$">
     <model:runcycles/>
     <model:matching>
       <model:sameDay/>
     </model:matching>
     <model:restrictions/>
     <model:dependencies/>
     <model:jobs>
     <model:job confirmed="false"
definition=
   "$WORKSTATION_MDM112097$
  #$JOB_BADPBN34_J1$"
 isCritical="false" iskey="false"
  name="$JOB_BADPBN34_J1$"
priority="10">
       <model:restrictions/>
       <model:dependencies>
        <model:predecessor target=
         "$WORKSTATION_BADPBN34_
         WC1$#$JOBSTREAM_BADPBN34_
         JS2$.@">
          <model:matching>
           <model:previous>
          </model:matching>
       </model:predecessor>
      </model:dependencies>
     </model:job>
    </model:jobs>
</model:JobStream>
```

Fichier de mappage

```
#Noms des postes de travail
#Remplacez la valeur par le nom
 d'un poste de travail qui existe
 déjà dans l'environnement cible.
#Voir le fichier MAIN_TEMPLATE_
  SourceEnv_reference.txt pour
  obtenir des détails sur le
  poste de travail.
#
#Ce poste de travail est de type
  Gestionnaire
WORKSTATION_MDM112097=MDM112097
#Ce poste de travail est de type
  Agent
WORKSTATION_MDM112097_1=
  MDM112097_1
#Ce poste de travail est de type
  Courtier
WORKSTATION_MDM112097_DWB=
  MDM112097_DWB
```

Fichier d'informations de référence

```
CPUNAME $WORKSTATION_MDM112097$
  DESCRIPTION "Exemple de
   gestionnaire de domaine maître
   de sauvegarde"
  OS UNIX
  NODE MDM112097.romelab.it.ibm.com
   TCPADDR 35111
  TIMEZONE Europe/Rome
  DOMAIN MASTERDM
  FOR MAESTRO
    TYPE MDM
    AUTOLINK ON
    BEHINDFIREWALL ON
    FULLSTATUS ON
    SERVER A
END
```

Tableau 63. Résolution du fichier de mappage (suite)

Fichier de définitions	Fichier de mappage	Fichier d'informations de référence
Le fichier d'informations de référence indique que le poste de travail nommé MDM112097 est de type gestionnaire de domaine maître s'exécutant sur un système d'exploitation UNIX. Le fichier de définitions contient des références au nom du poste de travail. L'entrée dans le fichier de mappage `WORKSTATION_MDM112097=MDM112097` doit ainsi être mise à jour. Remplacez `MDM112097` par le nom d'un poste de travail qui existe déjà dans l'environnement cible et qui est doté des mêmes caractéristiques que dans le fichier d'informations de référence.		

Fichier de définitions

```
<model:JobStream carryforward="true" draft="false"
 iskey="false" limit="55" name="$JOBSTREAM_BADPBN34_JS1_1I$"
 onrequest="false" priority="100"
 workstation="$WORKSTATION_BADPBN34_WC1$">
     <model:runcycles/>
     <model:matching>
       <model:sameDay/>
     </model:matching>
     <model:restrictions/>
     <model:dependencies/>
     <model:jobs>
     <model:job confirmed="false"
defintion="$WORKSTATION_MDM112097$#$JOB_BADPBN34_J1$"
 isCritical="false" iskey="false" name="$JOB_BADPBN34_J1$"
priority="10">
       <model:restrictions/>
       <model:dependencies>
         <model:predecessor target="$WORKSTATION_BADPBN34_WC1$#$JOBSTREAM_BADPBN34_JS2$.@">
          <model:matching>
           <model:previous>
          </model:matching>
       </model:predecessor>
      </model:dependencies>
     </model:job>
    </model:jobs>
</model:JobStream>
```

Déploiement d'un application de charge de travail

Le déploiement d'une application de charge de travail se compose de deux étapes. Il commence par la personnalisation du fichier de mappage et se termine par l'importation du fichier de mappage et du fichier de définitions dans le nouvel environnement Tivoli Workload Scheduler.

1. Extrayez le contenu du modèle d'application de charge de travail contenant le fichier de définitions, le fichier de mappage et le fichier d'informations de référence à partir du fichier compressé créé par l'opération d'exportation depuis Dynamic Workload Console.
2. Personnalisez le fichier de mappage. Attribuez à chaque objet répertorié dans le fichier de mappage le nom d'un objet existant dans l'environnement cible, ou renommez-le avec le nom de votre choix dans l'environnement cible. Voir «Résolution du fichier de mappage», à la page 363 pour plus d'informations sur la personnalisation du fichier de mappage.
3. A partir de la ligne de commande, soumettez la commande suivante, en indiquant les noms du fichier de définitions et du fichier de mappage personnalisé :

   ```
   wappman -import <fichier_XML_définitions> <fichier_propriétés_mappage>
   ```

   Voir «Commande wappman», à la page 369 pour obtenir des détails sur l'utilisation et la syntaxe de la commande.

Tous les objets Tivoli Workload Scheduler définis dans *fichier_XML_définitions* sont créés dans l'environnement cible, s'ils n'existent pas déjà. Vous mettez à jour le fichier de mappage afin de résoudre les références aux objets qui doivent être personnalisés pour refléter l'environnement cible dans lequel l'application de charge de travail sera déployée.

Vous pouvez mettre à jour une application de charge de travail ultérieurement. Il existe deux cas dans lesquels l'application de charge de travail peut être mise à jour ou modifiée :

Modification du modèle dans l'environnement source
Une version mise à jour du modèle peut être déployée à nouveau dans l'environnement cible. Tout objet déjà présent dans la base de données Tivoli Workload Scheduler de l'environnement cible est remplacé par les versions mises à jour. Au contraire, tout objet qui n'existe pas déjà dans l'environnement cible est créé et les objets sont supprimés de l'environnement cible si la définition d'objet a été supprimée de l'application de charge de travail mise à jour. Le même fichier de mappage utilisé pour déployer initialement l'application de charge de travail peut être utilisé pour la mettre à jour, en personnalisant tout nouvel objet en cours de déploiement avec la mise à jour.

Modification de l'instance dans l'environnement cible
Après avoir déployé une application de charge de travail, vous pouvez ajouter un nouveau travail à un flot de travaux, modifier une définition de travail ou supprimer un travail ou un flot de travaux. Ces modifications ne sont toutefois pas conservées si l'application de charge de travail est mise à jour avec un modèle d'application de charge de travail révisé.

Commande wappman

Crée, remplace, supprime, affiche et répertorie une application de charge de travail.

Remarque : Sous Windows 2012, la commande n'est pas prise en charge sur Windows PowerShell.

Autorisation

Vous devez avoir le droit d'accès ***add*** pour créer une application de charge de travail. Si l'objet existe déjà dans la base de données, vous devez avoir le droit d'accès ***modify*** à l'objet.

Syntaxe

wappman[*paramètres_connexion*]
[-u]|[-V]
| **{-import|-replace}** *<fichier_XML_définitions>* *<fichier_propriétés_mappage*
-list
-delete *<nom_application_charge_travail>*
-display *<nom_application_charge_travail>*
]

Arguments

[*paramètres_connexion***]**
Fichier contenant les paramètres de connexion à utiliser pour se connecter

au serveur. Vous pouvez utiliser les paramètres de connexion pour remplacer les valeurs spécifiées dans les fichiers useropts et localopts. Pour déterminer les propriétés de connexion à utiliser, une vérification est réalisée dans l'ordre suivant :

1. Paramètres spécifiés dans la chaîne de commande.
2. Paramètres spécifiés dans le fichier des propriétés personnalisées.
3. Fichier useropts.
4. Fichier localopts.

Les valeurs admises sont les suivantes :

[-file <*fichier_propriétés_personnalisées***>]**

[-host <*nom_hôte***>]**

[-port <*numéro_port***>]**

[-protocol {http | https}]

[-password <*mot_de_passe***>]**

[-username <*nom_utilisateur***>]**
où *nom_utilisateur* est un utilisateur Tivoli Workload Scheduler doté de privilèges suffisants pour effectuer l'opération.

-u Affiche des informations sur la syntaxe de la commande et quitte l'application.

-V Affiche la version de la commande et quitte l'application.

{-import|-replace} <*fichier_XML_définitions***> <***fichier_propriétés_mappage***>**
Nom du fichier de définitions et du fichier de mappage à utiliser lors de l'importation ou du remplacement d'une application de charge de travail. L'opération d'importation d'une application de charge de travail est identique au déploiement d'une application de charge de travail.

import
Après avoir personnalisé le fichier de mappage, celui-ci et le fichier de définitions sont transmis en tant qu'entrée à la commande **wappman -import** afin de déployer l'application de charge de travail dans un environnement Tivoli Workload Scheduler différent de l'environnement source dans lequel le modèle d'application de charge de travail a été initialement créé.

replace
Une version mise à jour du modèle d'application de charge de travail est réimportée dans l'environnement cible. Les objets qui étaient initialement créés dans la base de données Tivoli Workload Scheduler au premier déploiement du modèle d'application de charge de travail, sont mis à jour s'ils sont toujours présents dans la base de données, créés s'ils ne s'y trouvent plus et supprimés s'ils n'existent plus dans le modèle d'application de charge de travail mis à jour.

L'opération replace échoue si un objet externe à l'application de charge de travail référence un objet dans l'application de charge de travail. Suite à l'opération replace, l'objet référencé est supprimé car il n'existe plus dans l'application de charge de travail mise à jour.

Si des références ont été établies de l'application de charge de travail à un élément externe, la référence est supprimée à l'aide de l'action replace.

Le même fichier de mappage utilisé pour déployer initialement l'application de charge de travail peut être utilisé pour la mettre à jour, en effectuant les modifications nécessaires pour refléter l'application de charge de travail mise à jour. Le fichier de mappage et le fichier de définitions sont transmis en tant qu'entrée à la commande **wappman -replace**.

-list Répertorie toutes les applications de charge de travail présentes dans l'environnement.

-delete <*nom_application_charge_travail***>**
Supprime l'application de charge de travail spécifiée et tous les objets qui ont été ajoutés à l'environnement lorsque l'application de charge de travail a été initialement déployée.

-display <*nom_application_charge_travail***>**
Affiche le contenu de l'application de charge de travail spécifiée.

Affiche tous les objets présents dans l'application de charge de travail qui ont été créés au cours du processus d'importation.

Journaux et fichiers de trace

Vous pouvez configurer la ligne de commande **wappman** en modifiant les paramètres dans le fichier `CLI.ini` situé à l'emplacement *rép_base_TWA*`/TWS/ITA/cpa/config`. Ce fichier contient les paramètres des journaux de messages et des fichiers de trace associés aux applications de charge de travail. Vous ne devriez modifier les paramètres que si la documentation Tivoli Workload Scheduler le recommande ou si le service de support logiciel IBM vous invite à le faire.

Exemples

Pour importer une application de charge de travail dans un nouvel environnement, exécutez :

```
wappman -import <fichier_XML_définitions> <fichier_propriétés_mappage>
```

Pour remplacer une application de charge de travail existante, exécutez :

```
wappman -replace <fichier_XML_définitions> <fichier_propriétés_mappage>
```

Pour supprimer une application de charge de travail, exécutez :

```
wappman -delete <nom_application_charge_travail>
```

Pour répertorier toutes les applications de charge de travail disponibles dans un environnement, exécutez :

```
wappman -list
```

Pour afficher une application de charge de travail spécifique, exécutez :

```
wappman -display <nom_application_charge_travail>
```

Voir aussi

Pour plus d'informations sur la définition d'un modèle d'application de charge de travail dans l'environnement source, voir «Création d'un modèle d'application de charge de travail», à la page 298.

Voir «Déploiement d'un application de charge de travail», à la page 368 pour plus d'informations sur la gestion d'une application de charge de travail dans l'environnement cible.

Chapitre 11. Gestion des objets dans le plan - conman

L'environnement du plan de production Tivoli Workload Scheduler est géré à l'aide du programme de ligne de commande **conman**. Le programme **conman** est utilisé pour démarrer et arrêter le traitement, modifier et afficher le plan de production Symphony et contrôler les liens entre les postes de travail dans un réseau. Il peut être utilisé à partir du gestionnaire de domaine maître ou de tout agent tolérant aux pannes du réseau Tivoli Workload Scheduler. Le présent chapitre comprend les sections suivantes :

Configuration du programme de ligne de commande conman

Le programme de ligne de commande **conman** gère l'environnement du plan de production.

Vous pouvez utiliser le programme **conman** à partir du gestionnaire de domaine maître et de tout poste de travail agent tolérant aux pannes du réseau Tivoli Workload Scheduler.

Configuration de l'environnement Conman

Vous trouverez dans la présente section des informations sur la configuration que vous pouvez choisir pour votre environnement **conman**.

Remarque : Sur les systèmes Windows, avant d'exécuter **conman**, assurez-vous que la page de codes et les polices de caractères sont définies correctement dans le shell DOS afin d'éviter toute conversion incorrecte des caractères. Pour plus d'informations sur les paramètres obligatoires, voir la section *Remarques relatives à l'internationalisation* du document *IBM Tivoli Workload Scheduler: Release Notes*, à l'adresse http://www-01.ibm.com/support/docview.wss?rs=672 &uid=swg27041032.

Sortie sur le terminal

La sortie sur votre ordinateur est déterminée par les variables shell nommées ***MAESTROLINES*** et ***MAESTROCOLUMNS***. Si l'une ou l'autre de ces variables n'est pas définie, les variables du shell standard ***LINES*** et ***COLUMNS*** sont utilisées. Les variables peuvent être définies comme suit :

MAESTROLINES
: Indique le nombre de lignes par page d'écran. La valeur par défaut est **24**. A la fin de chaque page d'écran, **conman** vous invite à poursuivre. Si la variable ***MAESTROLINES*** (ou ***LINES***) a pour valeur zéro ou une valeur négative, le programme **conman** ne marque pas de pause à la fin de chaque page.

Il est recommandé d'utiliser MAESTROLINES puisque LINES est une variable du système d'exploitation shell et que dans la plupart des systèmes d'exploitation elle est automatiquement réinitialisée par le système lui-même.

MAESTROCOLUMNS
Indique le nombre de caractères par ligne. Les options suivantes sont disponibles :

- Inférieur à 120
- Supérieur ou égal à 120

MAESTRO_OUTPUT_STYLE
Indique la façon dont les noms d'objets sont affichés. Si la valeur définie est **LONG**, les noms complets sont affichés. Si la valeur est définie par une valeur différente de **LONG**, les noms longs sont tronqués à huit caractères suivis du signe (+).

Sortie offline

Dans les commandes **conman**, l'option **;offline** sert généralement à imprimer la sortie d'une commande. Lorsqu'elle est insérée, la sortie est contrôlée par les variables de shell ci-après :

MAESTROLP
Indique la destination de la sortie d'une commande. Vous pouvez indiquer :

> *fichier*
La sortie est dirigée vers un fichier dont elle remplace le contenu. Si le fichier n'existe pas, il est créé.

>> *fichier*
La sortie est dirigée vers un fichier et est ajoutée à la fin de celui-ci. Si le fichier n'existe pas, il est créé.

| *commande*
La sortie est dirigée vers une commande système ou un processus. Le programme exécute la commande système, que la sortie soit générée ou non.

|| *commande*
La sortie est dirigée vers une commande système ou un processus. La commande système n'est pas exécutée en l'absence de résultat.

La valeur par défaut de ***MAESTROLP*** est **| lp -tCONLIST** qui transmet la sortie de la commande à l'imprimante et insère le titre “CONLIST” sur la page bannière de la sortie imprimée.

MAESTROLPLINES
Indique le nombre de lignes par page. La valeur par défaut est **60**.

MAESTROLPCOLUMNS
Indique le nombre de caractères par ligne. La valeur par défaut est **132**.

Les variables doivent être exportées avant l'exécution de **conman**.

Sélection de l'invite conman sous UNIX

Par défaut, l'invite de commande **conman** est figurée par le signe %. Elle est définie dans le fichier *racine_TWS*/`localopts`. L'invite de commande par défaut correspond à un tiret (-). Pour sélectionner une autre invite, éditez l'option d'invite de **conman** dans le fichier `localopts` et modifiez le tiret. Le message d'invite peut comporter jusqu'à 10 caractères, le signe dièse (#) de fin requis non inclus.

```
#--------------------------------------------------------------------------
# Custom format attributes
#
date format =        1     # The possible values are 0-ymd, 1-mdy,
2-dmy, 3-NLS.
composer prompt =      -
conman prompt =       %
switch sym prompt =    <n>%
#--------------------------------------------------------------------------
```

Exécution de conman

Pour configurer l'environnement en vue d'utiliser **conman**, définissez les variables *PATH* et *TWS_TISDIR* en exécutant l'un des scripts suivants :

Sous UNIX :

- `. ./`*racine_TWS*`/`**tws_env.sh** pour les interpréteurs de commandes Bourne et Korn
- `. ./`*racine_TWS*`/`**tws_env.csh** pour les interpréteurs de commandes C

Sous Windows :

- *racine_TWS*`\`**tws_env.cmd**

Utilisez ensuite la syntaxe suivante pour exécuter des commandes à partir de l'interface utilisateur **conman** :

conman [*paramètres_connexion*] [**"***commande*[**&**[*commande*]...] [**&**]**"**]

où :

paramètres_connexion

Si vous utilisez **conman** à partir du gestionnaire de domaine maître, les paramètres de connexion ont été configurés lors de l'installation et n'ont pas besoin d'être fournis, à moins que vous ne vouliez pas utiliser les valeurs par défaut.

Si vous utilisez **conman** à partir du client de ligne de commande sur un autre poste de travail, les paramètres de connexion peuvent être fournis par l'une ou plusieurs des méthodes suivantes :

- Ils sont stockés dans le fichier `localopts`
- Ils sont stockés dans le fichier `useropts`
- Ils sont fournis à la commande dans un fichier de paramètres
- Ils sont fournis à la commande au sein d'une chaîne de commande

Pour une présentation de ces options, voir «Configuration des options pour l'utilisation des interfaces utilisateur», à la page 59. Pour des détails complets sur les paramètres de configuration, voir la rubrique relative à la configuration de l'accès du client de ligne de commande dans *Tivoli Workload Scheduler - Guide d'administration*.

Vous pouvez appeler la ligne de commande **conman** en mode de *traitement par lots* et en mode *interactif*.

Lorsque vous exécutez **conman** en mode *interactif*, vous lancez d'abord le programme de ligne de commande **conman**, puis vous exécutez les commandes l'une après l'autre à partir de l'invite de la ligne de commande **conman**, par exemple :

```
conman –username admin2 –password admin2pwd
   ss @+state=hold;deps
   dds sked5;needs=2 tapes
```

Lorsque vous exécutez **conman** en mode de *traitement par lots*, vous lancez d'abord le programme de ligne de commande **conman** en indiquant comme paramètre d'entrée la commande à émettre. Une fois la commande traitée, le programme de ligne de commande **conman** s'arrête comme dans l'exemple suivant :

```
conman"sj&sp"
```

Lors de l'émission de commandes à partir de **conman** en mode de traitement par lots, assurez-vous de délimiter les commandes entre guillemets. Voici des exemples de l'utilisation du mode de traitement par lots pour émettre plusieurs commandes depuis **conman** :

- **conman** exécute les commandes **sj** et **sp** avant de s'arrêter :

  ```
  conman"sj&sp"
  ```

- **conman** exécute les commandes **sj** et **sp**, puis invite l'utilisateur à taper une commande :

  ```
  conman "sj&sp&"
  ```

- **conman** lit les commandes à partir de **cfile** :

  ```
  conman < cfile
  ```

- les commandes du fichier **cfile** sont dirigées vers **conman** :

  ```
  cat cfile | conman
  ```

Remarque : Sur les postes de travail Windows, si le contrôle de compte utilisateur (UAC) est activé et que la liste d'exceptions UAC ne contient pas le fichier `cmd.exe`, vous devez ouvrir l'interpréteur de commandes DOS avec l'option `"Exécuter en tant qu'administrateur"` pour exécuter **conman** sur votre poste de travail en tant qu'utilisateur générique différent de l'administrateur ou utilisateur Tivoli Workload Scheduler.

Caractères de commande

Vous pouvez entrer les caractères de commande suivants pour arrêter **conman**.

Ctrl+c Le programme **conman** interrompt l'exécution de la commande dès que l'étape courante le permet et renvoie une invite de commande.

Ctrl+d **conman** s'arrête après avoir exécuté la commande courante, uniquement sur les postes de travail UNIX.

Exécution des commandes système

Lorsque vous entrez une commande système avec une barre verticale ou un préfixe de commande système (: ou !), elle est exécutée par un processus enfant. Pour garantir la sécurité, l'ID utilisateur effectivement associé au processus enfant est celui de l'utilisateur qui exécute **conman**.

Invites utilisateur

Si vous utilisez des caractères génériques pour sélectionner les objets soumis à l'action d'une commande, **conman** vous demande confirmation après avoir localisé chacun des objets recherchés. Si vous répondez **oui** l'action est effectuée sur l'objet, et si vous répondez **non** l'objet est ignoré et aucune action n'est effectuée.

Lorsque vous exécutez **conman** en mode interactif, les invites de confirmation sont exécutées sur votre ordinateur. La sélection de la touche **Retour** en réponse à une invite est interprétée comme étant une réponse **négative**. La fonction d'invite peut être désactivée dans une commande à l'aide de l'option **;noask**.

Bien qu'aucune invite de confirmation ne soit générée lorsque **conman** est exécuté en mode interactif, celui-ci se comporte comme si la réponse avait été **négative** dans chacun des cas et aucune action n'est effectuée sur les objets. Par conséquent, il est important d'inclure l'option **;noask** dans les commandes lorsque **conman** ne s'exécute pas en mode interactif.

Exécution de commandes depuis Conman

Les commandes du programme **conman** se décomposent de la manière suivante :

nom_commande sélection arguments

où :

nom_commande
Indique le nom de la commande.

sélection
Indique l'objet ou le groupe d'objets sur lequel sera exécutée la commande.

arguments
Indique les arguments de la commande.

Voici un exemple de commande **conman** :

```
sj sked1(1100 03/05/2006).@+state=hold~priority=0;info;offline
```

où :

sj Forme abrégée de la commande **showjobs**.

sked1(1100 03/05/2006).@+state=hold~priority=0
Sélectionne tous les travaux du flot de travaux **sked1(1100 03/05/2006)** qui sont à l'état HOLD et dont la priorité est différente de zéro.

;info;offline
Arguments de la commande **showjobs**.

Caractères génériques

Les caractères génériques acceptés sont les suivants :

@ Remplace un ou plusieurs caractères alphanumériques.

? Remplace un caractère alphanumérique.

% Remplace un caractère numérique.

Délimiteurs et caractères spéciaux

Le tableau 64 répertorie les caractères qui ont une signification spéciale dans les commandes **conman** :

Tableau 64. Délimiteurs et caractères spéciaux pour la commande conman

Caractère	Description
&	Délimiteur de commande. Voir «Configuration du programme de ligne de commande conman», à la page 373.
+	Délimiteur servant à sélectionner les objets pour les commandes. Il permet d'ajouter un attribut dont un objet doit être doté. Par exemple : `sked1(1100 03/05/2006).@~priority=0`

Tableau 64. Délimiteurs et caractères spéciaux pour la commande conman (suite)

Caractère	Description
~	Délimiteur servant à sélectionner les objets pour les commandes. Il permet d'ajouter un attribut dont un objet doit être doté. Par exemple : `sked1(1100 03/05/2006).@~priority=0`
;	Délimiteur d'argument. Par exemple : `;info;offline`
,	Délimiteur de répétition et de fourchette. Par exemple : `state=hold,sked,pend`
=	Délimiteur de valeur. Par exemple : `state=hold`
:!	Préfixes de commande qui transmettent la commande au système. Ces préfixes sont facultatifs. Si **conman** ne reconnaît pas la commande, celle-ci est transmise automatiquement au système. Par exemple : `!ls ou :ls`

*	Préfixe de commentaire. Le préfixe doit être le premier caractère de la ligne de commande ou suivre un délimiteur de commande. Par exemple :
	`*commentaire`
	ou
	`sj& *commentaire`

>	Dirige la sortie de la commande vers un fichier dont elle remplace le contenu. Si le fichier n'existe pas, il est créé. Par exemple : `sj> joblist`
>>	Dirige la sortie de la commande vers un fichier et celle-ci est ajoutée à la fin du fichier. Si le fichier n'existe pas, il est créé. Par exemple : `sj>> joblist`
|	La sortie de la commande est dirigée vers une commande système ou un processus. Le programme exécute la commande système, que la sortie soit générée ou non. Par exemple : `sj| grep ABEND`
||	La sortie de la commande est dirigée vers une commande système ou un processus. La commande système n'est pas exécutée en l'absence de résultat. Par exemple : `sj || grep ABEND`

Traitement des commandes Conman

Le programme **conman** exécute les commandes qui modifient l'état des objets, telles que le démarrage ou l'arrêt d'un poste de travail, ainsi que les commandes qui modifient les objets du plan de façon *asynchrone*. Cela explique le fait que vous observerez sans doute un décalage entre le moment où vous soumettez la commande et celui où les informations stockées dans le fichier `Symphony` sont mises à jour avec le résultat de la commande.

Ce décalage est dû au fait que le programme **conman** ne met pas à jour les informations dans le fichier `Symphony` ; **conman** soumet les commandes à **batchman** qui est le seul processus capable d'accéder aux informations du fichier `Symphony` et de les mettre à jour. Pour cette raison, vous devez attendre que **batchman** traite la demande de modification d'objet émise par **conman** et mette à jour les

informations relatives à l'objet stockées dans le fichier Symphony avant de voir les informations mises à jour dans la sortie de la commande ***showobj***.

Tout changement effectué à l'aide du programme conman qui affecte le fichier Symphony s'applique également aux informations du plan répliquées dans la base de données.

Par exemple, si vous demandez la suppression d'une dépendance à l'aide de la commande **conman deldep**, **conman** soumet la commande **deldep** en publiant un événement dans la boîte aux lettres Mailman.msg. Le processus **mailman** reçoit les informations relatives à la demande de suppression à partir de Mailman.msg et les place dans la boîte aux lettres Intercom.msg sur le poste de travail auquel appartient la ressource dont vous supprimez la dépendance. Sur chaque poste de travail, **batchman** reçoit les événements dans sa boîte aux lettres Intercom.msg et les traite selon leur ordre de réception. Dans le cas où **batchman** est occupé, pour quelque raison que ce soit, les événements transmettant des demandes d'exécution de commandes **conman** sont mis en file d'attente dans le fichier Intercom.msg en attendant d'être lus et traités par **batchman**.

En outre, lorsque **batchman** traite l'événement, l'utilisateur ne reçoit aucune notification. En conclusion, il se peut que vous supprimiez une dépendance et que la suppression n'apparaisse pas parce que **batchman** était trop occupé pour exécuter immédiatement l'opération demandée. Si vous réexécutez la commande, il se peut que la suppression soit déjà effective bien qu'un message indiquant que la commande a été transmise à **batchman** s'affiche dans l'invite **conman**.

Sélection de travaux dans les commandes

Pour les commandes qui s'exécutent sur les travaux, les travaux cible sont sélectionnés au moyen d'attributs et de qualifiants. La syntaxe de sélection des travaux fournie ci-après est décrite dans les pages suivantes.

Syntaxe

[*poste_de_travail***#**]{*nom_flot_travaux*(*hhmm*[*date*]).*nom_travail* | *numéro_travail*} [{**+** | **~**}*qualifiant_travail*[...]]

ou

[*poste_de_travail***#**]*ID_flot_travaux.travail* [{**+** | **~**]*qualifiant_travail*[...]]**;schedid**

ou :

agent_réseau::[*poste_de_travail***#**]{*nom_flot_travaux*(*hhmm*[*date*]).*nom_travail* | *ID_flot_travaux.nom_travail***;schedid**}

Arguments

poste de travail
Utilisé avec le fichier *flot_travaux.travail*, cet argument indique le nom du poste de travail sur lequel est exécuté le flot de travaux. Utilisé avec l'argument *numéro_travail*, il indique le poste de travail sur lequel est exécuté le travail. Excepté lorsque schedid est également utilisé, les caractères génériques sont autorisés. Cet argument peut être obligatoire, selon le poste de travail sur lequel vous lancez la commande, comme suit :

- Si vous lancez la commande sur le poste de travail sur lequel les travaux cible se sont exécutés, l'argument *workstation* est facultatif.
- Si vous lancez la commande sur un poste de travail hébergé, l'argument *workstation* est obligatoire. Les postes de travail hébergés sont :
  - agents étendus
  - agents
  - les pools
  - les pools dynamiques

nom_flot_travaux
Indique le nom du flot de travaux dans lequel s'exécute le travail. Les caractères génériques sont acceptés.

(*hhmm* **[***date***])**
Indique l'heure et la date auxquelles l'instance du flot de travaux est située dans le plan de préproduction. La valeur *hhmm* correspond à la valeur attribuée au mot clé **schedtime** dans la définition du flot de travaux si aucune restriction temporelle **at** n'a été définie. Une fois que le traitement de l'instance du flot de travaux a démarré, la valeur de ***hhmm* [*date*]** est définie sur l'heure à laquelle le flot de travaux a démarré. L'utilisation de caractères génériques n'est pas autorisée dans cette zone. Lorsque vous émettez des commandes **conman** en ligne à partir de l'invite shell, mettez la commande **conman** entre guillemets **" "**. Par exemple, exécutez la commande suivante comme suit :

```
conman "sj my_workstation#my_js(2101 02/23).@"
```

ID_flot_travaux
Indique l'identificateur unique du flot de travaux. Voir «Arguments», à la page 389 pour plus d'informations sur les identificateurs de flot de travaux.

schedid Indique que l'identificateur du flot de travaux est utilisé pour sélectionner le flot de travaux.

nom_travail
Indique le nom du travail. Les caractères génériques sont acceptés.

numéro_travail
Indique le numéro du travail.

qualifiant_travail
Voir la section suivante .

agent_réseau
Indique le nom de l'agent de réseau Tivoli Workload Scheduler qui assure l'interface avec le réseau Tivoli Workload Scheduler distant sur lequel se trouve le travail cible. Les deux-points doubles (::) sont un délimiteur obligatoire. Les caractères génériques sont acceptés. Pour plus d'informations, voir Chapitre 18, «Gestion des dépendances interréseaux», à la page 671.

Remarque : Tivoli Workload Scheduler vous aide à identifier l'instance de flot de travaux correct lorsque la sélection de flot de travaux fournit un résultat ambigu si plus d'une instance répond à vos critères de sélection. Par exemple, lorsque plusieurs instances de `WK1#J1` sont incluses dans le plan de production et que, par conséquent, la sélection de flot de travaux fournit un résultat ambigu, l'invite suivante est générée automatiquement pour vous permettre de choisir l'instance appropriée :

```
Process WK1#J1[(0600 03/04/06),(0AAAAAAAAAAAABTB)]
        (tapez "y" pour oui, "n" pour non)?y
Commande réacheminée vers batchman pour WK1#J1[(0600 03/04/06),(0AAAAAAAAAAAABTB)]
Process WK1#J1[(1010 03/04/06),(0AAAAAAAAAAAABTC)] (tapez "y" pour oui, "n" pour non)?n
```

Dans le résultat, seule l'instance du flot de travaux planifiée le `(0600 03/04/06)` et ayant pour identificateur `0AAAAAAAAAAAABTB` est sélectionnée pour exécuter la commande.

Qualifiants de travaux

Les qualifiants de travaux indiquent les attributs de travaux sur lesquels une commande doit agir. Ils peuvent être précédés d'un signe + ou ~. Si un qualifiant de travail est précédé par un +, alors les travaux contenant cet attribut spécifique sont sélectionnés pour exécuter la commande. Si un qualifiant de travail est précédé par un ~, alors les travaux contenant cet attribut spécifique sont exclus de l'exécution de la commande.

Les mots clés des qualifiants de travaux peuvent être abrégés en conservant autant de premiers caractères que nécessaire pour les distinguer les uns des autres de façon unique.

at[=*heure* | *heure_min,* | **,***heure_max* | *heure_min*,*heure_max* **]**
Sélectionne ou exclut des travaux en fonction de l'heure spécifiée dans la dépendance **at**.

heure

Indique l'heure comme suit :

hhmm[**+***n* **days** | *date*] [**timezone**|**tz** *fuseau_horaire*]

où :

hhmm Heure et minute.

+*n* **days**
Occurrence suivante de *hhmm* en *n* nombre de jours.

date Occurrence suivante de *hhmm* à la *date* indiquée, exprimée au format *mm/jj[/aa]*.

timezone|**tz** *fuseau_horaire*
Nom du fuseau horaire du travail. Voir Chapitre 16, «Gestion des fuseaux horaires», à la page 653 pour les noms valides.

heure respecte les règles suivantes:

- Lorsque *hhmm* est antérieure à l'heure courante, l'heure de début est le lendemain ; lorsque *hhmm* est postérieure à l'heure courante, l'heure de début est le jour même.
- Lorsque *hhmm* est supérieure à 2400, elle est divisée par 2400. La partie entière du résultat de la division représente le nombre de *jours en* +, tandis que la partie décimale représente l'heure.

heure_min
Indique la limite inférieure d'une plage horaire exprimée au même format que l'*heure*. Les travaux planifiés pour être lancés après cette heure sont sélectionnés.

heure_max
Indique la limite supérieure d'une plage horaire, exprimée au

même format que l'*heure*. Les travaux planifiés pour être lancés avant cette heure sont sélectionnés.

Si **at** est utilisé seul et qu'il est précédé du caractère +, alors les travaux sélectionnés sont ceux qui contiennent une dépendance **at**.

Si **at** est utilisé seul et qu'il est précédé du caractère ~, alors les travaux sélectionnés sont ceux qui ne contiennent pas de dépendance **at**.

confirmed
Sélectionne ou exclut les travaux qui ont été planifiés à l'aide du mot clé **confirm**.

critical
Permet de sélectionner ou d'exclure les travaux associé au mot clé **critical** dans une définition de flot de travaux.

critnet Permet de sélectionner ou d'exclure les travaux signalés comme **critical** ou qui sont des prédécesseurs de travaux critiques. Ainsi, il s'applique à tous les travaux dont la date de début est critique.

L'heure de début critique d'un travail critique est calculée en soustrayant sa durée estimée de son échéance. L'heure de début critique d'un prédécesseur est calculée en soustrayant sa durée estimée de l'heure de début critique de son successeur. Dans un réseau critique, les heures de début critiques sont calculées en commençant par le travail critique et en fonctionnant de façon rétroactive le long de l aligne des prédécesseurs.

deadline[=*heure* | *heure_min*, | ,*heure_max* | *heure_min*,*heure_max*]
Indique le temps nécessaire pour l'achèvement d'un travail.

hhmm[**+***n* **days** | *date*] [**timezone**|**tz** *fuseau_horaire*]

hhmm Heure et minute.

+*n* **days**
Décalage en jours à partir de l'heure d'échéance planifiée.

date Occurrence suivante de *hhmm* à la *date* indiquée, exprimée au format *mm/jj[/aa]*.

timezone|**tz** *fuseau_horaire*
Indique le fuseau horaire à utiliser lors du calcul de l'heure d'échéance. Voir Chapitre 16, «Gestion des fuseaux horaires», à la page 653 pour les noms de fuseaux horaires. Il s'agit par défaut du fuseau horaire correspondant au poste de travail sur lequel le travail ou le flot de travaux est lancé.

heure_min
Indique la limite inférieure d'une plage horaire exprimée au même format que l'*heure*. Les travaux sélectionnés ont une heure d'échéance planifiée qui n'est pas antérieure à cette heure.

heure_max
Indique la limite supérieure d'une plage horaire, exprimée au même format que l'*heure*. Les travaux sélectionnés ont une heure d'échéance planifiée qui n'est pas postérieure à cette heure.

every[=*fréquence* | *fréquence_min*, | ,*fréquence_max* | *fréquence_min*,*fréquence_max*]
Sélectionne ou exclut des travaux soumis ou non à une fréquence de répétition.

fréquence
Indique la fréquence d'exécution planifiée, exprimée en heures et en minutes (*hhmm*).

fréquence_min
Indique la limite inférieure d'une plage de fréquence, exprimée au même format que la *fréquence*. Les travaux dont la fréquence de répétition est supérieure ou égale à la fréquence indiquée sont sélectionnés.

fréquence_max
Indique la limite supérieure d'une plage de fréquence, exprimée au même format que la *fréquence*. Les travaux dont la fréquence de répétition est inférieure ou égale à la fréquence indiquée sont sélectionnés.

Si **every** est utilisé seul et qu'il est précédé du caractère +, alors les travaux sélectionnés sont ceux qui contiennent une fréquence de répétition.

Si **every** est utilisé seul et qu'il est précédé du caractère ~, alors les travaux sélectionnés sont ceux qui ne contiennent aucune fréquence de répétition.

finished[=*heure* **|** *heure_min***, |** *,heure_max* **|** *heure_min***,***heure_max***]**
Sélectionne ou exclut les travaux selon qu'ils sont terminés ou non.

heure Indique l'heure de fin exacte du travail, exprimée comme suit :

hhmm [*date*] [**timezone|tz** *fuseau_horaire*]

hhmm Heure et minute.

date Occurrence suivante de *hhmm* à la date indiquée, exprimée au format *mm/jj[/aa]*.

timezone|tz *fuseau_horaire*
Nom du fuseau horaire du travail. Voir Chapitre 16, «Gestion des fuseaux horaires», à la page 653 pour les noms valides.

heure_min
Indique la limite inférieure d'une plage horaire exprimée au même format que l'*heure*. Les travaux qui sont terminés à cette heure ou après sont sélectionnés.

heure_max
Indique la limite supérieure d'une plage horaire, exprimée au même format que l'*heure*. Les travaux qui sont terminés à cette heure ou avant sont sélectionnés.

Si **finished** est utilisé seul et qu'il est précédé du caractère +, alors les travaux sélectionnés sont ceux dont l'exécution est terminée.

Si **finished** est utilisé seul et qu'il est précédé du caractère ~, alors les travaux sélectionnés sont ceux dont l'exécution n'est pas terminée.

follows=[*agent_réseau***::][***poste_de_travail***#]{***nom_flot_travaux***(***hhmm* **[***mm/jj***[/***aa***]])[.***travail* **| @] | ***ID_flot_travaux***.***travail***;schedid}| ***travail***[;nocheck][,...]**
Sélectionne ou exclut les travaux selon qu'ils comportent ou non une dépendance de prédécesseur/successeur.

agent_réseau
Indique le nom de l'agent de réseau Tivoli Workload Scheduler qui assure l'interface avec le réseau Tivoli Workload Scheduler distant sur lequel se trouve le travail prérequis. Les caractères génériques

sont acceptés. Pour plus d'informations, voir Chapitre 18, «Gestion des dépendances interréseaux», à la page 671.

poste de travail
Indique le nom du poste de travail sur lequel le travail prérequis est exécuté. Les caractères génériques sont acceptés.

nom_flot_travaux
Indique le nom du flot de travaux dans lequel le travail prérequis est exécuté. Les caractères génériques sont acceptés. Si vous entrez *nom_flot_travaux***.@**, indiquez que le travail cible vient à la suite de tous les travaux du flot de travaux.

nom_travail
Indique le nom du travail prérequis. Lorsque ce paramètre est entré sans *nom_flot de travaux*, il indique que le travail prérequis se trouve dans le même flot de travaux que le travail cible. Pour une dépendance de prédécesseur/successeur, n'utilisez pas de caractères génériques dans le nom du travail.

ID_flot_travaux
Indique l'identificateur unique du flot de travaux. Voir «Arguments», à la page 389 pour plus d'informations sur les identificateurs de flot de travaux.

schedid
Ce mot clé, s'il est présent, s'applique à tous les identificateurs de flots de travaux spécifiés dans la clause et indique que pour tous les flots de travaux spécifiés, vous utilisez les arguments *ID_flot_travaux* et non les arguments *nom_flot_travaux*. Si vous souhaitez sélectionner des flots de travaux à l'aide de l'argument *ID_flot_travaux* et certains flots de travaux à l'aide de l'argument *nom_flot_travaux*, vous devez utiliser deux clauses **follows** différentes : l'une contenant les flots de travaux identifiés par l'argument *nom_flot_travaux* sans les mots clés **schedid** et l'autre contenant les flots de travaux identifiés par l'argument *ID_flot_travaux* avec le mot clé **schedid**.

nocheck
Valide uniquement pour les commandes de soumission et utilisé en association avec le mot clé **schedid**. Le mot clé **nocheck** indique que **conman** n'a pas besoin de vérifier l'existence du travail prérequis *ID_flot_travaux.travail* dans le fichier `Symphony`. Le programme considère que *ID_flot_travaux.travail* existe et dans le cas contraire **batchman** imprime un message d'avertissement dans `stdlist`.

Si **follows** est utilisé seul et qu'il est précédé du caractère +, alors les travaux sont sélectionnés s'ils contiennent des dépendances de prédécesseur/successeur.

Si **follows** est utilisé seul et qu'il est précédé du caractère ~, alors les travaux sont sélectionnés s'ils ne contiennent aucune dépendance de prédécesseur/successeur.

logon=*nom_utilisateur*
Sélectionnez les travaux en fonction des noms d'utilisateur sous lesquels ils sont exécutés. Si le *nom utilisateur* contient des caractères spéciaux, il doit être placé entre guillemets ("). Les caractères génériques sont acceptés.

needs[=[*poste_de_travail***#]***nom_ressource***]**
Sélectionne ou exclut des travaux selon qu'ils comportent ou non une dépendance de ressource.

poste de travail
Indique le nom du poste de travail sur lequel la ressource est définie. Les caractères génériques sont acceptés.

nom_ressource
Indique le nom de la ressource. Les caractères génériques sont acceptés.

Si **needs** est utilisé seul et qu'il est précédé du caractère +, alors les travaux sont sélectionnés s'ils contiennent des dépendances `resource`.

Si **needs** est utilisé seul et qu'il est précédé du caractère ~, alors les travaux sont sélectionnés s'ils ne contiennent aucune dépendance `resource`.

opens[=[*poste_de_travail***#]***nom_ficheir***[(***qualifiant***)]]**
Sélectionne des travaux selon qu'ils comportent ou non une dépendance `file`. Une dépendance `file` se produit lorsque la présence d'un ou de plusieurs fichiers permet l'exécution d'un travail ou d'un flot de travaux.

poste de travail
Indique le nom du poste de travail sur lequel se trouve le fichier. Les caractères génériques sont acceptés.

file_name
Indique le nom du fichier. Ce nom doit être placé entre guillemets (") s'il contient des caractères spéciaux autres que les suivants : caractères alphanumériques, tirets (-), barres obliques (/), barres obliques inversées (\) et traits de soulignement (_). Les caractères génériques sont acceptés.

qualifier
Condition de test valide. Si elle est omise, les travaux sont sélectionnés ou exclus, quel que soit le qualifiant utilisé.

Si **opens** est utilisé seul et qu'il est précédé du caractère +, alors les travaux sont sélectionnés s'ils contiennent des dépendances `file`.

Si **opens** est utilisé seul et qu'il est précédé du caractère ~, alors les travaux sont sélectionnés s'ils ne contiennent aucune dépendance `file`.

priority=*pri* | *pri_min***,** | **,***pri_max* | *pri_min***,***pri_max*
Sélectionne ou exclut des travaux en fonction de leurs priorités.

pri Indique la valeur de la priorité. Vous pouvez entrer une valeur comprise entre **0** et **99**, **hi** ou **go**.

pri_min
Indique la limite inférieure d'une plage de priorité. Les travaux dont la priorité est égale ou supérieure à cette valeur sont sélectionnés.

pri_max
Indique la limite supérieure d'une plage de priorité. Les travaux dont la priorité est inférieure ou égale à cette valeur sont sélectionnés.

prompt[=*nom_invite* | *num_message***]**
Sélectionne ou exclut les travaux selon qu'ils comportent ou non une dépendance `prompt`.

nom_invite
Indique le nom d'une invite globale. Les caractères génériques sont acceptés.

msgnum
Indique le numéro de message d'une invite locale.

Si **prompt** est utilisé seul et qu'il est précédé du caractère +, alors les travaux sont sélectionnés s'ils contiennent des dépendances `prompt`.

Si **prompt** est utilisé seul et qu'il est précédé du caractère ~, alors les travaux sont sélectionnés s'ils ne contiennent aucune dépendance `prompt`.

recovery=*option_reprise*
Sélectionne ou exclut des travaux en fonction de leurs options de reprise.

recv-option
Affecte à l'option de reprise de travaux la valeur **stop**, **continue** ou **rerun**.

scriptname=*file_name*
Sélectionne ou exclut des travaux en fonction du nom de leurs fichiers exécutables.

file_name
Indique le nom d'un fichier exécutable. Ce nom doit être placé entre guillemets (") s'il contient des caractères spéciaux autres que les suivants : caractères alphanumériques, tirets (-), barres obliques (/), barres obliques inversées (\) et traits de soulignement (_). Les caractères génériques sont acceptés.

started[=*heure* | *heure_min*, | ,*heure_max* | *heure_min*,*heure_max*]
Sélectionne ou exclut des travaux selon qu'ils ont été lancés ou non.

heure Indique l'heure de début exacte des flots de travaux, exprimée comme suit :

hhmm [*date*] [**timezone** | **tz** *fuseau_horaire*]

hhmm Heure et minute.

date Occurrence suivante de *hhmm* à la date indiquée, exprimée au format *mm/jj[/aa]*.

timezone | **tz** *fuseau_horaire*
Nom du fuseau horaire du travail. Voir Chapitre 16, «Gestion des fuseaux horaires», à la page 653 pour les noms valides.

heure_min
Indique la limite inférieure d'une plage horaire exprimée au même format que l'*heure*. Uniquement les travaux qui ont été lancés à cette heure ou après sont sélectionnés.

heure_max
Indique la limite supérieure d'une plage horaire, exprimée au même format que l'*heure*. Uniquement les travaux qui ont été lancés à cette heure ou avant sont sélectionnés.

Si **started** est utilisé seul et qu'il est précédé du caractère +, alors uniquement les travaux dont l'exécution a démarré à cette heure sont sélectionnés.

Si **started** est utilisé seul et qu'il est précédé du caractère ~, alors uniquement les travaux dont l'exécution a démarré à cette heure ou ultérieurement et est toujours en cours sont sélectionnés.

state=*état***[,...]**
Sélectionne ou exclut des travaux en fonction de leur état.

état Indique l'état courant du travail. Les états acceptés pour un travail sont les suivants :

ABEND
Le travail s'est terminé avec un code de sortie différent de zéro.

ABENP
Une confirmation **abend** a été reçue, mais le travail n'est pas terminé.

ADD Le travail est en cours de soumission.

CANCL
Ne s'applique qu'aux dépendances interréseau. Le travail distant ou le flot de travaux a été annulé.

DONE
Le travail est terminé avec un état inconnu.

ERROR
Pour les dépendances interréseaux uniquement, une erreur s'est produite lors de la recherche du statut distant.

EXEC Le travail est en cours. L'indicateur + en regard de l'état EXEC signifie que le travail est géré par le processus **batchman** local.

EXTRN
Pour les dépendances interréseaux uniquement, l'état est inconnu. Une erreur s'est produite, une action de réexécution vient d'être effectuée sur le travail du flot de travaux `EXTERNE`, ou le travail ou le flot de travaux distant n'existe pas.

FAIL Impossible de lancer le travail.

FENCE
La valeur de la priorité du travail est inférieure à la priorité minimale.

HOLD
Le travail attend la résolution des dépendances.

INTRO
Le travail est introduit à des fins de lancement par le système. L'indicateur + en regard de l'état INTRO signifie que le travail est géré par le processus **batchman** local.

PEND Le travail est terminé et attend une confirmation.

READY
Le travail est prêt à démarrer et toutes les dépendances sont résolues.

SCHED
L'heure **at** du travail n'est pas arrivée.

SUCC Le travail s'est terminé avec un code de sortie nul.

SUCCP
Une confirmation **succ** a été reçue mais le travail n'est pas terminé.

WAIT Le travail est à l'état WAIT (agent étendu uniquement).

until[=*heure* **|** *heure_min*, **|** ,*heure_max* **|** *heure_min*,*heure_max* **]**
Sélectionne ou exclut des travaux en fonction de l'heure de fin planifiée qui leur est associée.

heure Indique l'heure de fin planifiée exprimée comme suit :

hhmm[**+***n* **days** | *date*] [**timezone**|**tz** *fuseau_horaire*]

hhmm Heure et minute.

+*n* **days**
Occurrence suivante de *hhmm* en *n* nombre de jours.

date Occurrence suivante de *hhmm* à la *date* indiquée, exprimée au format *mm/jj[/aa]*.

timezone|**tz** *fuseau_horaire*
Nom du fuseau horaire du travail. Voir Chapitre 16, «Gestion des fuseaux horaires», à la page 653 pour les noms valides.

heure_min
Indique la limite inférieure d'une plage horaire exprimée au même format que l'*heure*. Les travaux dont l'heure de fin planifiée est égale ou postérieure à l'heure indiquée sont sélectionnés.

heure_max
Indique la limite supérieure d'une plage horaire, exprimée au même format que l'*heure*. Les travaux dont l'heure de fin planifiée est égale ou antérieure à l'heure indiquée sont sélectionnés.

Si **until** est utilisé seul et qu'il est précédé du caractère +, alors les travaux sont sélectionnés s'ils contiennent une heure **until** spécifiée.

Si **until** est utilisé seul et qu'il est précédé du caractère ~, alors les travaux sont sélectionnés s'ils ne contiennent aucune heure **until** spécifiée.

Sélection de flots de travaux dans les commandes

Pour les commandes qui s'exécutent sur les flots de travaux, les flots de travaux cible sont sélectionnés à travers la spécification d'attributs et de qualifiants.

Etant donné que ***date_et_heure_calendrier*** est spécifié en minutes, la combinaison de **nom_flot_travaux** et de l'heure *date_et_heure_calendrier* peut ne pas être unique. Pour cette raison, l'argument **ID_flot_travaux** est disponible pour l'utilisateur, soit à des fins d'affichage, soit pour effectuer des actions sur une instance spécifique d'un flot de travaux.

La syntaxe de sélection des flots de travaux fournie ci-après est décrite dans les pages suivantes. Vous pouvez choisir l'une des deux syntaxes décrites.

Syntaxe

[*poste_de_travail***#**]*nom_flot_travaux*(*hhmm*[*date*]) [{**+** | **~**}*qualifiant_flot_travaux*[...]]

[*poste_de_travail***#**]*ID_flot_travaux* **;schedid**

Arguments

poste de travail
Indique le nom du poste de travail sur lequel le flot de travaux est exécuté. Excepté lorsque `schedid` est également utilisé, les caractères génériques sont autorisés.

nom_flot_travaux
Indique le nom du flot de travaux. Les caractères génériques sont acceptés.

(*hhmm* **[***date***])**
Indique l'heure et la date auxquelles l'instance du flot de travaux est située dans le plan de préproduction. Cette valeur correspond à la valeur affectée au mot clé **schedtime** dans la définition du flot de travaux si aucune restriction temporelle **at** n'a été définie. Une fois que le traitement de l'instance du flot de travaux a démarré, la valeur de ***hhmm*** **[*date*]** est définie sur l'heure à laquelle le flot de travaux a démarré. L'utilisation de caractères génériques n'est pas autorisée dans cette zone. Lorsque vous émettez des commandes **conman** en ligne à partir de l'invite shell, mettez la commande **conman** entre guillemets **(")**. Par exemple, exécutez la commande suivante comme suit :

```
conman "ss my_workstation#my_js(2101 02/23)"
```

qualifiant_flot_travaux
Voir Identificateurs de flot de travaux ci-après.

ID_flot_travaux
Indique l'identificateur de flot de travaux alphanumérique unique généré automatiquement par le planificateur et affecté à ce flot de travaux. Celui-la permet essentiellement aux processus internes d'identifier l'instance du flot de travaux dans le plan de production, mais il peut souvent être utilisé lors de la gestion du flot de travaux à partir du programme de ligne de commande **conman** en spécifiant l'option **;schedid**.

schedid
Indique que l'identificateur du flot de travaux est utilisé pour sélectionner le flot de travaux.

Remarque : Tivoli Workload Scheduler vous aide à identifier l'instance de flot de travaux correct lorsque la sélection de flot de travaux fournit un résultat ambigu si plus d'une instance répond à vos critères de sélection. Par exemple, lorsque plusieurs instances de `WK1#J1` sont incluses dans le plan de production et que, par conséquent, la sélection de flot de travaux fournit un résultat ambigu, l'invite suivante est générée automatiquement pour vous permettre de choisir l'instance appropriée :

```
Process WK1#J1[(0600 03/04/06),(0AAAAAAAAAAAABTB)]
        (tapez "y" pour oui, "n" pour non)?y
Commande réacheminée vers batchman pour WK1#J1[(0600 03/04/06),(0AAAAAAAAAAAABTB)]
Process WK1#J1[(1010 03/04/06),(0AAAAAAAAAAAABTC)] (tapez "y" pour oui, "n" pour non)?n
```

Dans le résultat, seule l'instance du flot de travaux planifiée le `(0600 03/04/06)` et ayant pour identificateur `0AAAAAAAAAAAABTB` est sélectionnée pour exécuter la commande.

Qualifiants de flot de travaux

at[=*heure* | *heure_min*, | ,*heure_max* | *heure_min*,*heure_max* **]**
Sélectionne ou exclut des flots de travaux en fonction de l'heure de début planifiée.

heure Indique l'heure de début planifiée exprimée comme suit :

hhmm[**+***n* **days** | *date*] [**timezone**|**tz** *fuseau_horaire*]

hhmm Heure et minute.

+*n* **days**
Occurrence suivante de *hhmm* en *n* nombre de jours.

date Occurrence suivante de *hhmm* à la *date* indiquée, exprimée au format *mm/jj[/aa]*.

timezone|**tz** *fuseau_horaire*
Nom du fuseau horaire du flot de travaux. Voir Chapitre 16, «Gestion des fuseaux horaires», à la page 653 pour les noms valides.

heure_min
Indique la limite inférieure d'une plage horaire exprimée au même format que l'*heure*. Les flots de travaux dont l'heure de début planifiée est égale ou postérieure à l'heure indiquée sont sélectionnés.

heure_max
Indique la limite supérieure d'une plage horaire, exprimée au même format que l'*heure*. Les flots de travaux dont l'heure de début planifiée est égale ou antérieure à l'heure indiquée sont sélectionnés.

Si **at** est utilisé seul et qu'il est précédé du caractère +, alors les flots de travaux sélectionnés sont ceux qui contiennent une dépendance **at**.

Si **at** est utilisé seul et qu'il est précédé du caractère ~, alors les flots de travaux sélectionnés sont ceux qui ne contiennent pas de dépendance **at**.

carriedforward
Sélectionne les flots de travaux qui ont été reportés s'il est précédé du caractère +, ou les exclut s'il est précédé de ~.

carryforward
Sélectionne les flots de travaux qui ont été planifiés à l'aide du mot clé **carryforward** s'il est précédé du caractère +, ou les exclut s'il est précédé de ~.

finished[=*heure* | *heure_min*, | ,*heure_max* | *heure_min*,*heure_max***]**
Sélectionne ou exclut les flots de travaux selon qu'ils sont terminés ou non.

heure Indique l'heure de fin exacte des flots de travaux, exprimée comme suit :

hhmm [*date*] [**timezone**|**tz** *fuseau_horaire*]

hhmm Heure et minute.

date Occurrence suivante de *hhmm* à la date indiquée, exprimée au format *mm/jj[/aa]*.

timezone | tz *fuseau_horaire*
Nom du fuseau horaire du flot de travaux. Voir Chapitre 16, «Gestion des fuseaux horaires», à la page 653 pour les noms valides.

heure_min
Indique la limite inférieure d'une plage horaire exprimée au même format que l'*heure*. Les flots de travaux qui sont terminés à cette heure ou après sont sélectionnés.

heure_max
Indique la limite supérieure d'une plage horaire, exprimée au même format que l'*heure*. Les flots de travaux qui sont terminés à cette heure ou avant sont sélectionnés.

Si **finished** est utilisé seul et qu'il est précédé du caractère +, alors les flots de travaux sélectionnés sont ceux dont l'exécution est terminée.

Si **finished** est utilisé seul et qu'il est précédé du caractère ~, alors les flots de travaux sélectionnés sont ceux dont l'exécution n'est pas terminée.

follows=[*agent_réseau***::][***poste_de_travail***#]{***nom_flot_travaux***(***hhmm* **[***mm/jj***[/***aa***]])[.***travail* **| @] | *ID_flot_travaux*.*travail*;schedid} | *travail*[;nocheck] [,...]**
Sélectionne ou exclut les flots de travaux selon qu'ils comportent ou non une dépendance de `prédécesseur/successeur`.

agent_réseau
Indique le nom de l'agent de réseau qui assure l'interface avec le réseau Tivoli Workload Scheduler distant contenant le travail ou le flot de travaux prérequis. Les caractères génériques sont acceptés. Pour plus d'informations sur les agents réseau, voir Chapitre 18, «Gestion des dépendances interréseaux», à la page 671.

poste de travail
Indique le nom du poste de travail sur lequel est exécuté le travail ou le flot de travaux prérequis. Les caractères génériques sont acceptés.

nom_flot_travaux
Indique le nom du flot de travaux prérequis. Les caractères génériques sont acceptés.

nom_travail
Indique le nom du travail prérequis. Les caractères génériques sont acceptés.

ID_flot_travaux
Indique l'identificateur unique du flot de travaux. Voir «Arguments», à la page 389 pour plus d'informations sur les identificateurs de flot de travaux.

schedid
Ce mot clé, s'il est présent, s'applique à tous les identificateurs de flots de travaux spécifiés dans la clause et indique que pour tous les flots de travaux spécifiés, vous utilisez les arguments *ID_flot_travaux* et non les arguments *nom_flot_travaux*. Si vous souhaitez sélectionner des flots de travaux à l'aide de l'argument *ID_flot_travaux* et certains flots de travaux à l'aide de l'argument *nom_flot_travaux*, vous devez utiliser deux clauses **follows** différentes : l'une contenant les flots de travaux identifiés par l'argument *nom_flot_travaux* sans les mots clés **schedid** et l'autre

contenant les flots de travaux identifiés par l'argument *ID_flot_travaux* avec le mot clé **schedid**.

nocheck
Valide uniquement pour les commandes de soumission et utilisé en association avec le mot clé **schedid**. Le mot clé **nocheck** indique que **conman** n'a pas besoin de vérifier l'existence du travail prérequis *ID_flot_travaux.travail* dans le fichier `Symphony`. Le programme considère que *ID_flot_travaux.travail* existe et dans le cas contraire **batchman** imprime un message d'avertissement dans `stdlist`.

Si **follows** est utilisé seul et qu'il est précédé du caractère +, alors les flots de travaux sont sélectionnés s'ils contiennent des dépendances de prédécesseur/successeur.

Si **follows** est utilisé seul et qu'il est précédé du caractère ~, alors les flots de travaux sont sélectionnés s'ils ne contiennent aucune dépendance de prédécesseur/successeur.

limit[=*limite* **|** *limite_min***, |** *,limite_max* **|** *limite_min***,***limite_max***]**
Sélectionne ou exclut les flots de travaux selon qu'ils comportent ou non un nombre maximal de travaux.

limit Indique la valeur précise du nombre maximal de travaux.

limite_min
Indique la limite inférieure d'une plage. Les flots de travaux dont le nombre maximal de travaux est égal ou supérieur au nombre maximal indiqué sont sélectionnés.

limite_max
Indique la limite supérieure d'une fourchette. Les flots de travaux dont le nombre maximal de travaux est inférieur ou égal au nombre maximal indiqué sont sélectionnés.

Si **limit** est utilisé seul et qu'il est précédé du caractère +, alors les flots de travaux sont sélectionnés s'ils sont associés à un nombre maximal de travaux.

Si **limit** est utilisé seul et qu'il est précédé du caractère ~, alors les flots de travaux sont sélectionnés s'ils ne sont associés à aucun nombre maximal de travaux.

needs[=[*poste_de_travail***#]***nom_ressource***]**
Sélectionne ou exclut les flots de travaux selon qu'ils comportent ou non une dépendance de ressource.

poste de travail
Indique le nom du poste de travail sur lequel la ressource est définie. Les caractères génériques sont acceptés.

nom_ressource
Indique le nom de la ressource. Les caractères génériques sont acceptés.

Si **needs** est utilisé seul et qu'il est précédé du caractère +, alors les flots de travaux sont sélectionnés s'ils contiennent des dépendances `resource`.

Si **needs** est utilisé seul et qu'il est précédé du caractère ~, alors les flots de travaux sont sélectionnés s'ils ne contiennent aucune dépendance `resource`.

opens[=[*poste_de_travail***#]***nom_ficheir***[(***qualifiant***)]]**
Sélectionne ou exclut les flots de travaux selon qu'ils comportent ou non une dépendance de fichier. Une dépendance de fichier se produit lorsque la présence d'un ou de plusieurs fichiers permet l'exécution d'un travail ou d'un flot de travaux.

poste de travail
Indique le nom du poste de travail sur lequel se trouve le fichier. Les caractères génériques sont acceptés.

file_name
Indique le nom du fichier. Ce nom doit être placé entre guillemets (") s'il contient des caractères spéciaux autres que les suivants : caractères alphanumériques, tirets (-), barres obliques (/), barres obliques inversées (\) et traits de soulignement (_). Les caractères génériques sont acceptés.

qualifier
Condition de test valide. En cas d'omission, les flots de travaux sont sélectionnés ou exclus, quel que soit le qualifiant utilisé.

Si **opens** est utilisé seul et qu'il est précédé du caractère +, alors les flots de travaux sont sélectionnés s'ils contiennent des dépendances `file`.

Si **opens** est utilisé seul et qu'il est précédé du caractère ~, alors les flots de travaux sont sélectionnés s'ils ne contiennent aucune dépendance `file`.

priority=*pri* **|** *pri_min***, |** ***,****pri_max* **|** *pri_min***,***pri_max*
Sélectionne ou exclut les flots de travaux en fonction de leurs priorités.

pri Indique la valeur de la priorité. Vous pouvez entrer une valeur comprise entre **0** et **99**, **hi** ou **go**.

pri_min
Indique la limite inférieure d'une plage de priorité. Les flots de travaux dont la priorité est égale ou supérieure à cette valeur sont sélectionnés.

pri_max
Indique la limite supérieure d'une plage de priorité. Les flots de travaux dont la priorité est inférieure ou égale à cette valeur sont sélectionnés.

prompt[=*nom_invite* **|** *num_message***]**
Sélectionne ou exclut les flots de travaux selon qu'ils comportent ou non une dépendance d'invite.

nom_invite
Indique le nom d'une invite globale. Les caractères génériques sont acceptés.

msgnum
Indique le numéro de message d'une invite locale.

Si **prompt** est utilisé seul et qu'il est précédé du caractère +, alors les flots de travaux sont sélectionnés s'ils contiennent des dépendances `prompt`.

Si **prompt** est utilisé seul et qu'il est précédé du caractère ~, alors les flots de travaux sont sélectionnés s'ils ne contiennent aucune dépendance `prompt`.

started[=*heure* **|** *heure_min***, |** ***,****heure_max* **|** *heure_min***,***heure_max***]**
Sélectionne ou exclut les flots de travaux selon qu'ils ont été lancés ou non.

heure Indique l'heure de début exacte du flot de travaux, exprimée comme suit :

hhmm [*date*] [**timezone** | **tz** *fuseau_horaire*]

hhmm Heure et minute.

date Occurrence suivante de *hhmm* à la date indiquée, exprimée au format *mm/jj[/aa]*.

timezone | **tz** *fuseau_horaire*
Nom du fuseau horaire du flot de travaux. Voir Chapitre 16, «Gestion des fuseaux horaires», à la page 653 pour les noms valides.

heure_min
Indique la limite inférieure d'une plage horaire exprimée au même format que l'*heure*. Les flots de travaux qui sont lancés à cette heure ou après sont sélectionnés.

heure_max
Indique la limite supérieure d'une plage horaire, exprimée au même format que l'*heure*. Les flots de travaux qui sont lancés à cette heure ou avant sont sélectionnés.

Si **started** est utilisé seul et qu'il est précédé du caractère +, alors les flots de travaux sélectionnés sont ceux dont l'exécution a démarré.

Si **started** est utilisé seul et qu'il est précédé du caractère ~, alors les flots de travaux sélectionnés sont ceux dont l'exécution n'a pas démarré.

state=*état***[,...]**
Sélectionne ou exclut les flots de travaux en fonction de leur état.

état Indique l'état courant du flot de travaux. Les états des flots de travaux sont les suivants :

ABEND
Le flot de travaux s'est arrêté de manière anormale.

ADD Le flot de travaux vient d'être soumis.

EXEC Le flot de travaux est en cours d'exécution.

EXTRN
Ne s'applique qu'aux dépendances interréseau. Il s'agit de l'état du flot de travaux EXTERNAL contenant des travaux qui font référence à des travaux ou des flots de travaux dans le réseau distant.

HOLD
Le flot de travaux attend la résolution des dépendances.

READY
Le flot de travaux est prêt à démarrer et toutes les dépendances sont résolues.

STUCK
L'exécution est interrompue. Aucun travail n'est lancé sans l'intervention de l'opérateur.

SUCC Le flot de travaux s'est achevé correctement.

until[=*heure* **|** *heure_min*, **|** ,*heure_max* **|** *heure_min*,*heure_max* **]**
Sélectionne ou exclut les flots de travaux en fonction de l'heure de fin planifiée.

heure Indique l'heure de fin planifiée exprimée comme suit :

hhmm[**+***n* **days** | *date*] [**timezone|tz** *fuseau_horaire*]

hhmm Heure et minute.

+*n* **days**
Occurrence suivante de *hhmm* en *n* nombre de jours.

date Occurrence suivante de *hhmm* à la *date* indiquée, exprimée au format *mm/jj[/aa]*.

timezone|tz *fuseau_horaire*
Nom du fuseau horaire du flot de travaux. Voir Chapitre 16, «Gestion des fuseaux horaires», à la page 653 pour les noms valides.

heure_min
Indique la limite inférieure d'une plage horaire exprimée au même format que l'*heure*. Les flots de travaux dont l'heure de fin planifiée est égale ou postérieure à l'heure indiquée sont sélectionnés.

heure_max
Indique la limite supérieure d'une plage horaire, exprimée au même format que l'*heure*. Les flots de travaux dont l'heure de fin planifiée est égale ou antérieure à l'heure indiquée sont sélectionnés.

Si **until** est utilisé seul et qu'il est précédé du caractère +, les flots de travaux sélectionnés sont ceux qui contiennent une heure de fin planifiée.

Si **until** est utilisé seul et qu'il est précédé du caractère ~, les flots de travaux sélectionnés sont ceux qui ne contiennent aucune heure de fin planifiée.

Gestion des travaux et des flots de travaux des agents de niveau antérieur

Le changement apporté par la version 8.3 de Tivoli Workload Scheduler à la convention de nom des instances de flot de travaux nécessite que vous appliquiez le contournement suivant lors de l'émission de commandes de ligne de commande sur un plan généré sur un gestionnaire de domaine maître Tivoli Workload Scheduler de version 8.3 (ou ultérieure) à partir d'agents Tivoli Workload Scheduler version 8.1, 8.2, or 8.2.1 :

- Vous devez utiliser le symbole @ comme premier caractère de l'identificateur de l'instance du flot de travaux. Par exemple, le flot de travaux qui s'exécute sur le poste de travail `CPU1` avec l'identificateur `0AAAAAAAAAAAAY3` doit être identifié dans la ligne de commande **conman** comme suit :

  ```
  CPU1#@AAAAAAAAAAAAY3
  ```

- Vous ne pouvez pas utiliser le mot clé **follows** lorsque vous ajoutez une dépendance à un travail ou un flot de travaux ou lorsque vous soumettez une commande ou un fichier comme travail.
- Vous ne pouvez pas utiliser le mot clé **into** pour définir le flot de travaux dans lequel le travail doit être ajouté lorsque vous soumettez une commande ou un fichier comme travail.

Par exemple, pour afficher les informations sur le travail `job2` contenu dans l'instance de flot de travaux avec l'identificateur `0AAAAAAAAAAAAT1` s'exécutant sur le poste de travail `CPU1`, exécutez la commande suivante sur les agents de Tivoli Workload Scheduler version 8.1, 8.2 ou 8.2.1 :

```
sj CPU1#@AAAAAAAAAAAAT1.job2
```

Ces modifications seront également visibles dans les rapports et les journaux, ainsi qu'à tous les autres endroits où les noms de flot de travaux sont imprimés ou affichés.

Commandes Conman

Le tableau 65 répertorie les commandes **conman**. Les noms de commande et les mots clés peuvent être entrés en majuscules ou en minuscules et peuvent être abrégés au nombre minimal de caractères suffisant pour les distinguer les uns des autres de façon unique. Certains noms de commandes ont également une forme abrégée spécifique.

Remarque : Les types de poste de travail dans le tableau suivant ont les significations suivantes :

- D - gestionnaires de domaine dynamique, gestionnaires de domaine de sauvegarde
- M - gestionnaires de domaine maître et maîtres de sauvegarde
- F - gestionnaires de domaine et agents tolérants aux pannes
- T - pour l'agents tolérants aux pannes
- S - agents standard (vous pouvez uniquement afficher des fichiers sur un agent standard)

Tableau 65. Liste des commandes Conman

Commande	Forme abrégée	Description	Type	Page
adddep { job | sched }	**adj | ads**	Ajoute des dépendances d'un travail ou d'un flot de travaux.	F	«adddep job», à la page 399 «adddep sched», à la page 401
altpass		Modifie un mot de passe de définition d'objet utilisateur.	F	«altpass», à la page 403
altpri	**ap**	Modifie les priorités d'un travail ou d'un flot de travaux.	F	«altpri», à la page 404
bulk_discovery	**bulk**	Exécute une reconnaissance avec accès simultané. A utiliser avec IBM Tivoli Monitoring 6.1 (Tivoli Enterprise Portal).	F	«bulk_discovery», à la page 405
cancel { job | sched }	**cj | cs**	Annule un travail ou un flot de travaux.	F	«cancel job», à la page 405 «cancel sched», à la page 407
checkhealthstatus	**chs**	Appelle le service **chkhltst** afin de vérifier que la boîte aux lettres peut être lue correctement par **mailman** ou s'il y a des erreurs dans l'en-tête de boîte aux lettres.	MFS	«checkhealthstatus», à la page 408
confirm	**conf**	Confirme l'achèvement d'un travail.	F	«confirm», à la page 409
console	**cons**	Affecte la console Tivoli Workload Scheduler.	FS	«console», à la page 410

Tableau 65. Liste des commandes Conman (suite)

Commande	Forme abrégée	Description	Type	Page
continue	**cont**	Ignore l'erreur suivante.	FS	«continue», à la page 411
deldep { job | sched }	**ddj | dds**	Supprime les dépendances du travail ou du flot de travaux.	F	«deldep job», à la page 412 «deldep sched», à la page 413
deployconf	**deploy**	Obtient la configuration de surveillance la plus récente pour le moteur de surveillance d'événement sur le poste de travail.	FS	«deployconf», à la page 415
display { file | job | sched }	**df | dj | ds**	Affiche les fichiers, les travaux et les flots de travaux.	FS	«display», à la page 415
exit	**e**	Quitte le programme **conman**.	FS	«exit», à la page 418
fence	**d**	Définit la priorité minimale de travail de Tivoli Workload Scheduler.	F	«fence», à la page 418
help(1)	**h**	Affiche les informations relatives à la commande.	FS	«help», à la page 420
kill	**k**	Arrête l'exécution d'un travail.	F	«kill», à la page 421
limit { cpu | sched }	**lc | ls**	Modifie le nombre maximal de travaux d'un poste de travail ou d'un flot de travaux.	F	«limit cpu», à la page 422 «limit sched», à la page 423
link	**lk**	Etablit des liaisons de poste de travail.	FS	«link», à la page 424
listsym	**lis**	Affiche la liste de fichiers journaux `Symphony`.	F	«listsym», à la page 427
recall	**rc**	Affiche les messages d'invite.	F	«recall», à la page 428
redo	**red**	Edite la commande précédente.	FS	«redo», à la page 430
release { job | sched }	**rj | rs**	Libère les dépendances du travail ou du flot de travaux.	F	«release job», à la page 431 «release sched», à la page 432
reply	**rep**	Répond au message d'invite.	F	«reply», à la page 434
rerun	**rr**	Lance une nouvelle exécution d'un travail.	F	«rerun», à la page 435
resetFTA	**N/A**	Récupère un fichier Symphony endommagé sur l'agent tolérant aux pannes spécifié	T	«resetFTA», à la page 438
resource	**res**	Modifie le nombre d'unités de ressource.	F	«resource», à la page 439
setsym	**set**	Sélectionne un fichier journal Symphony.	F	«setsym», à la page 440
showcpus	**sc**	Affiche les informations relatives au poste de travail et aux liaisons.	FS	«showcpus», à la page 440
showdomain	**showd**	Affiche les informations relatives au domaine.	FS	«showdomain», à la page 447
showfiles	**sf**	Affiche les informations relatives aux fichiers.	F	«showfiles», à la page 449
showjobs	**sj**	Affiche les informations relatives aux travaux.	F	«showjobs», à la page 451

Tableau 65. Liste des commandes Conman (suite)

Commande	Forme abrégée	Description	Type	Page
showprompts	**sp**	Affiche les informations relatives aux invites.	F	«showprompts», à la page 468
showresources	**sr**	Affiche les informations relatives aux ressources.	F	«showresources», à la page 470
showschedules	**ss**	Affiche les informations relatives aux flots de travaux.	F	«showschedules», à la page 473
shutdown	**shut**	Arrête les processus de production de Tivoli Workload Scheduler.	FS	«shutdown», à la page 478
start		Lance les processus de production de Tivoli Workload Scheduler.	FS	«start», à la page 479
startappserver		Démarre le processus WebSphere Application Server	DM	«startappserver», à la page 481
startbrokerapp		Démarre l'application Dynamic Workload Broker.	DM	«startbrokerapp», à la page 482
starteventprocessor	**startevtp**	Démarre le serveur de traitement d'événements.	M(2)	«starteventprocessor», à la page 483
startmon	**startm**	Démarre le processus `monman` qui active le moteur de surveillance d'événements sur l'agent.	FS	«startmon», à la page 483
status	**stat**	Affiche l'état de la production de Tivoli Workload Scheduler.	FS	«status», à la page 484
stop		Arrête les processus de production de Tivoli Workload Scheduler.	FS	«stop», à la page 485
stop ;progressive		Stoppe les processus de production de Tivoli Workload Scheduler de façon hiérarchique.		«stop ;progressive», à la page 487
stopappserver	**stopapps**	Arrête le processus WebSphere Application Server	DM	«stopappserver», à la page 488
stopbrokerapp		Arrête l'application Dynamic Workload Broker.	DM	«stopbrokerapp», à la page 490
stopeventprocessor	**stopevtp**	Arrête le serveur de traitement d'événements.	M(2)	«stopeventprocessor», à la page 490
stopmon	**stopm**	Arrête le moteur de surveillance d'événements sur l'agent.	FS	«stopmon», à la page 491
submit { docommand | file | job | sched }	**sbd | sbf | sbj | sbs**	Soumet une commande, un fichier, un travail ou un flot de travaux.	FS(3)	«submit docommand», à la page 492 «submit file», à la page 495 «submit job», à la page 499 «submit sched», à la page 502
switcheventprocessor	**switchevtp**	Fait passer le serveur de traitement d'événements des gestionnaires de domaine maîtres aux maîtres de sauvegarde ou inversement.	M	«switcheventprocessor», à la page 506

Tableau 65. Liste des commandes Conman (suite)

Commande	Forme abrégée	Description	Type	Page
switchmgr	**switchm**	Change le gestionnaire de domaine.	F	«switchmgr», à la page 507
commande système		Envoie une commande au système.	FS	«Commande système», à la page 509
tellop	**to**	Envoie un message à la console.	FS	«tellop», à la page 509
unlink		Ferme les liaisons de poste de travail.	FS	«unlink», à la page 510
version	**v**	Affiche la bannière du programme de ligne de commande **conman**.	FS	«version», à la page 512

1. Non disponible sur le système d'exploitation Windows pris en charge.
2. Inclut les postes de travail installés comme maîtres de sauvegarde mais utilisés comme agents tolérants aux pannes ordinaires.
3. Vous pouvez utiliser les commandes **submit job** (**sbj**) et **submit sched** (**sbs**) sur un agent standard en utilisant les paramètres de connexion ou en spécifiant les paramètres dans le fichier `useropts` lors de l'appel de la ligne de commande **conman**.

Remarque : Dans les commandes, les termes *sched* et *schedule* se rapportent aux *flots de travaux*, et le terme *CPU* aux *postes de travail*.

adddep job

Ajoute des dépendances à un travail.

Vous devez avoir un accès ***adddep*** au travail. Pour inclure des dépendances `needs` et `prompt`, vous devez avoir un accès ***use*** aux ressources et aux invites globales.

Syntaxe

{**adddep job**
| **adj**} *travail_sélectionné*
;*dépendance*[**;**...]
[**;noask**]

Arguments

travail_sélectionné
Voir «Sélection de travaux dans les commandes», à la page 379.

dépendance
Type de dépendance. Indiquez l'un des éléments suivants. Les caractères génériques ne sont pas admis.

at=*hhmm*[**timezone** | **tz** *fuseau_horaire*][**+***n* **days** | *mm/jj[/aa]*] | [**absolute** | **abs**]

confirmed

deadline=*heure* [**timezone**|**tz** *fuseau_horaire*][**+***n* **day**[**s** | *mm/jj[/aa]*]

every=*fréquence*

follows=[*agent_réseau***::**][*poste_de_travail***#**]{*nom_flot_travaux*[*hhmm* [*mm/jj[/aa]*]][*.travail* | **@**] | *ID_flot_travaux.travail***;schedid**} | *travail*[**,**...]

maxdur=[*hhhmm*] [**onmaxdur** *action*]

mindur=[*hhhmm*] [**onmindur** *action*]

needs=[*num*] [*poste_de_travail***#**]*ressource*[**,**...]

opens=[*poste_de_travail***#**]"*file_name*"[**(***qualifiant***)**][**,**...] **priority**[**=***pri* | **hi** | **go**]

prompt="[**:** | **!**]*texte***"** | *nom_invite*[**,**...]

until *heure* [**timezone** | **tz** *fuseau_horaire*][**+***n* **day**[**s**]] | [**absolute** | **abs**] [**;onuntil** *action*]

noask Indique que l'utilisateur ne doit pas être invité à confirmer avant d'effectuer une opération sur chaque travail éligible.

Remarque :

1. Si vous ajoutez deux fois une dépendance sur un flot de travaux à un travail, les deux dépendances sont traitées.
2. Lorsque vous utilisez le mot-clé **deadline**, veillez à ce qu'une valeur supérieure à 0 soit attribuée à l'option **bm check deadline** dans le fichier de configuration `localopts` sur les postes de travail que vous utilisez. Définissez l'option **bm check deadline** sur chaque poste de travail sur lequel vous souhaitez être notifié de l'expiration du délai, ou, si vous souhaitez obtenir des informations à jour sur l'ensemble de l'environnement, définissez l'option dans le gestionnaire de domaine maître. Les échéances des travaux critiques sont évaluées automatiquement, indépendamment de l'option **bm check deadline**. Pour de plus amples informations au sujet de l'option **bm check deadline**, reportez-vous aux détails du fichier Localopts.

Commentaires

Si aucune valeur n'est indiquée pour `priority`, le travail retourne à sa priorité planifiée initialement. Si aucun poste de travail n'est indiqué dans `follows`, `needs` ou `opens`, le poste de travail sur lequel le travail s'exécute est utilisé par défaut.

Vous ne pouvez pas utiliser cette commande pour ajouter une ressource ou une invite sous la forme de dépendances `resource` ou `prompt`, sauf si ces dernières sont déjà référencées par un travail ou un flot de travaux dans le fichier `Symphony`.

Exemples

Pour ajouter une dépendance de ressource au travail `job3` du flot de travaux `sked9(0900 02/19/06)`, exécutez la commande suivante :

```
adj sked9(0900 02/19/06).job3 ; needs=2 tapes
```

Pour ajouter une dépendance de prédécesseur/successeur externe depuis un travail `JOB022` du flot de travaux `MLN#SCHED_02(0600 02/24)` vers `JOBA` du flot de travaux `MLN#NEW_TEST(0900 02/19/06)`, exécutez la commande suivante :

```
adj MLN#NEW_TEST(0900 02/19/06).JOBA ; follows MLN#SCHED_02(0600 02/24/06).JOB022
```

Pour ajouter une dépendance de fichier et une échéance **until** au travail `j6` du flot de travaux `JS2(0900 02/19/06)`, exécutez la commande suivante :

```
adj WK1#JS2(0900 02/19/06).j6 ; opens="/usr/lib/prdata/file5"(-s %p) ; until=2330
```

Pour arrêter un travail lorsqu'il a été exécuté plus de 9 heures et 1 minutes, exécutez la commande suivante :

```
conman addep BROOKS#JOBDEF1 ;maxdur=901 ;onmaxdur kill
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer les tâches suivantes

1. Dans la barre de navigation, cliquez sur **Statut et état de santé du système** > **Surveillance de la charge de travail** > **Surveillance des travaux**.
2. Sélectionnez **Tous les travaux du plan** ou une autre tâche de surveillance des flots de travaux.
3. Choisissez un nom pour le moteur ou spécifiez les propriété de connexion, puis cliquez sur **OK**.
4. Dans le tableau qui contient la liste des travaux, sélectionnez le travail auquel vous souhaitez ajouter une dépendance et cliquez sur **Dépendances...**.
5. Dans le panneau des dépendances, développez une section de dépendance et cliquez sur le bouton correspondant à l'action à effectuer.

adddep sched

Ajoute des dépendances à un flot de travaux.

Vous devez avoir un accès ***adddep*** au flot de travaux. Pour inclure des dépendances `needs` et `prompt`, vous devez avoir un accès ***use*** aux ressources et aux invites globales.

Syntaxe

{**adddep sched** | **ads**} *flot_travaux_sélectionné*
;*dépendance*[**;**...]
[**;noask**]

Arguments

flot_travaux_sélectionné
Voir «Sélection de flots de travaux dans les commandes», à la page 388.

dépendance
Type de dépendance. Indiquez l'un des éléments suivants. Les caractères génériques ne sont pas admis.

at=*hhmm*[**timezone** | **tz** *fuseau_horaire*][**+***n* **days** | *mm/jj[/aa]*] | [**absolute** | **abs**]

carryforward

deadline=*heure* [**timezone**|**tz** *fuseau_horaire*][**+***n* **day**[**s** | *mm/jj[/aa]*]

follows=[*agent_réseau*::][*poste_de_travail*#]{*nom_flot_travaux*[*hhmm* [*mm/jj[/aa]*]][*.travail* | **@**] | *ID_flot_travaux.travail*;**schedid**}| *travail*[**,**...]

limit=*limite*

needs=[*num*] [*poste_de_travail*#]*ressource*[**,**...]

opens=[*poste_de_travail*#]"*file_name*"[**(***qualifiant***)**][**,**...] **priority**[**=***pri* | **hi** | **go**]

prompt="[**:** | **!**]*texte***"** | *nom_invite*[**,**...]

until *heure* [**timezone** | **tz** *fuseau_horaire*][+*n* **day**[**s**] | [**absolute** | **abs**]] [**;onuntil** *action*]

noask Indique que l'utilisateur ne doit pas être invité à confirmer avant d'effectuer une opération sur chaque flot de travaux éligible.

Remarque :

1. Si vous ajoutez deux fois une dépendance à un flot de travaux sur un autre flot de travaux, seule une dépendance est prise en compte.
2. Lorsque vous utilisez le mot-clé **deadline**, veillez à ce qu'une valeur supérieure à 0 soit attribuée à l'option **bm check deadline** dans le fichier de configuration `localopts` sur les postes de travail que vous utilisez. Définissez l'option **bm check deadline** sur chaque poste de travail sur lequel vous souhaitez être notifié de l'expiration du délai, ou, si vous souhaitez obtenir des informations à jour sur l'ensemble de l'environnement, définissez l'option dans le gestionnaire de domaine maître. Les échéances des travaux critiques sont évaluées automatiquement, indépendamment de l'option **bm check deadline**. Pour de plus amples informations au sujet de l'option **bm check deadline**, reportez-vous aux détails du fichier Localopts.

Commentaires

- Si aucune valeur n'est indiquée pour `priority`, le flot de travaux retourne à sa priorité planifiée à l'origine.
- Si vous n'indiquez aucune valeur pour **limit**, 0 est la valeur par défaut.
- Si aucun poste de travail n'est indiqué dans `follows`, `needs` ou `opens`, le poste de travail sur lequel le flot de travaux s'exécute est utilisé par défaut.
- Vous ne pouvez pas utiliser cette commande pour ajouter une ressource ou une invite sous la forme de dépendances `resource` ou `prompt`, sauf si ces dernières existent déjà dans le plan de production. Pour voir quelle ressource et quelles invites existent dans le plan, voir «showresources», à la page 470 and «showprompts», à la page 468.

Exemples

Pour ajouter une dépendance d'invite au flot de travaux `sked9(0900 02/19/06)`, exécutez la commande suivante :

```
ads sked9(0900 02/19/06) ; prompt=msg103
```

Pour ajouter une dépendance de prédécesseur/successeur externe depuis un travail `JOBB` du flot de travaux `CPUA#SCHED_02(0600 02/24/06)` et un nombre maximal de travaux vers le flot de travaux `CPUA#TEST(0900 02/19/06)`, exécutez la commande suivante :

```
ads CPUA#TEST(0900 02/19/06) ; follows CPUA#SCHED_02(0600 02/24/06).JOBB; limit=2
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche comme suit :

1. Dans la barre de navigation, cliquez sur **Statut et état de santé du système** > **Surveillance de la charge de travail** > **Surveillance des flots de travaux**.
2. Sélectionnez **Tous les flots de travaux du plan** ou une autre tâche de surveillance des flots de travaux.
3. Choisissez un nom pour le moteur ou spécifiez les propriété de connexion, puis cliquez sur **OK**

4. Dans le tableau qui contient la liste des flots de travaux, sélectionnez le flot de travaux auquel vous souhaitez ajouter une dépendance et cliquez sur **Dépendances...** tab.
5. Dans le panneau des dépendances, développez une section de dépendance et cliquez sur le bouton correspondant à l'action à effectuer.

altpass

Modifie le mot de passe d'un objet utilisateur dans le plan de production en cours.

Vous devez avoir un accès ***altpass*** à l'objet utilisateur.

Syntaxe

altpass
[*poste_de_travail***#**]
nom_utilisateur
[**;"***mot_de_passe***"**]

Arguments

poste de travail
Indique le poste de travail sur lequel l'utilisateur est défini. Utilisez des majuscules pour cette zone même si vous avez utilisé des majuscules et des minuscules lorsque vous avez spécifié la valeur *poste_de_travail* dans la définition de l'utilisateur. Pour plus d'informations, consultez «Définition d'utilisateur», à la page 211. Ne spécifiez pas cette zone si l'utilisateur appartient à un domaine Windows géré par Active Directory. Par défaut, le poste de travail est celui sur lequel vous exécutez **conman**.

nom_utilisateur
Indique le nom d'un utilisateur. Utilisez le même nom d'utilisateur défini dans la base de données Tivoli Workload Scheduler. Les noms d'utilisateurs sont sensibles à la casse. Pour plus d'informations, voir «Définition d'utilisateur», à la page 211.

password
Indique le nouveau mot de passe. Elle doit figurer entre guillemets. Pour n'indiquer aucun mot de passe pour l'utilisateur, utilisez deux guillemets consécutifs (**""**).

Commentaires

Si vous n'indiquez aucun *mot de passe*, **conman** vous invite à entrer un mot de passe et une confirmation. Le mot de passe ne s'affiche pas lorsqu'il est saisi et ne doit pas être placé entre guillemets. Il est à noter que la modification s'effectue uniquement dans le plan de production en cours. Elle est donc temporaire. Pour apporter un changement permanent, voir «Définition d'utilisateur», à la page 211.

Exemples

Pour remplacer le mot de passe de l'utilisateur `Jim` sur le poste de travail `mis5` par `mynewpw`, exécutez la commande suivante :

```
altpass MIS5#JIM;"mynewpw"
```

Pour remplacer le mot de passe de l'utilisateur `jim` sur le poste de travail `Mis5` par `mynewpw` sans l'afficher, exécutez la commande suivante :

```
altpass MIS5#JIM
password: xxxxxxxx
confirm: xxxxxxxx
```

Pour remplacer le mot de passe de l'utilisateur `Jim`, défini dans un domaine Windows géré par Active Directory nommé `twsDom`, par `mynewpw`, exécutez la commande suivante :

```
altpass TWSDOM\JIM;"mynewpw"
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle décrite dans :

Dynamic Workload Console - Guide d'utilisation, section relative à la modification du mot de passe dans le plan.

altpri

Modifie la priorité d'un travail ou d'un flot de travaux.

Vous devez avoir un accès ***altpri*** au travail ou au flot de travaux.

Syntaxe

{**altpri** | **ap**}
travail_sélectionné | *flot_travaux_sélectionné*
[**;***pri*]
[**;noask**]

Arguments

travail_sélectionné
Voir «Sélection de travaux dans les commandes», à la page 379.

flot_travaux_sélectionné
Voir «Sélection de flots de travaux dans les commandes», à la page 388.

pri Indique le niveau de priorité. Vous pouvez entrer une valeur comprise entre **0** et **99**, **hi** ou **go**.

noask Indique que l'utilisateur ne doit pas être invité à confirmer avant d'effectuer une opération sur chaque travail ou flot de travaux éligible.

Exemples

Pour modifier la priorité du travail `balance` du flot de travaux `glmonth(0900 02/19/06)`, exécutez la commande suivante :

```
ap glmonth(0900 02/19/06).balance;55
```

Pour modifier la priorité du flot de travaux `glmonth(0900 02/19/06)`, exécutez la commande suivante :

```
ap glmonth(0900 02/19/06);10
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle qui suit :

1. Dans la barre de navigation, cliquez sur **Statut et état de santé du système** > **Surveillance de la charge de travail** > **Surveillance des travaux** ou **Surveillance des flots de travaux**.
2. Sélectionnez une tâche de surveillance.
3. Choisissez un nom pour le moteur ou spécifiez les propriété de connexion, puis cliquez sur **OK**
4. De la table contenant la liste d'objets, sélectionnez le travail ou le flot de travaux dont vous souhaitez modifier la priorité et cliquez sur l'onglet **Actions supplémentaires** > **priorité**.
5. Dans le panneau qui s'affiche, définissez la nouvelle priorité et cliquez sur **OK**.

bulk_discovery

Demande une reconnaissance avec accès simultané pour mettre à jour le statut en cours des objets surveillés. Elle est utilisée pour l'intégration à IBM Tivoli Monitoring 6.1 (Tivoli Enterprise Portal).

Vous devez avoir l'accès ***display*** à l'objet fichier.

Syntaxe

{**bulk_discovery** | **bulk**}

Commentaires

Lorsque l'intégration avec IBM Tivoli Monitoring 6.1 est activée, la commande **bulk_discovery** vérifie le statut de tous les travaux et flots de travaux surveillés du plan et écrit les événements correspondants dans le fichier journal des événements.

Par défaut, les événements sont écrits dans le fichier **event.log** .

les messages indiquant le début et la fin de l'activité de reconnaissance avec accès simultané sont journalisés dans le fichier `twsmerge.log`.

cancel job

Annule un travail.

Vous devez avoir un accès ***cancel*** au travail.

Syntaxe

{**cancel job** |
cj} *travail_sélectionné*
[**;pend**]
[**;noask**]

Arguments

travail_sélectionné
Voir «Sélection de travaux dans les commandes», à la page 379.

pend Annule le travail uniquement après la résolution de ses dépendances.

noask Indique que l'utilisateur ne doit pas être invité à confirmer avant d'effectuer une opération sur chaque travail éligible.

Commentaires

Si vous annulez un travail avant de l'avoir lancé, il n'est pas lancé. Si vous annulez un travail après l'avoir lancé, il continue 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.

Si l'option **;pend** n'est pas utilisée, les travaux et les flots de travaux qui dépendent du travail annulé sont libérés immédiatement de la dépendance.

Si l'option **;pend** est indiquée et que le travail n'a pas été lancé, l'annulation est différée jusqu'à ce que toutes les dépendances, y compris l'heure d'exécution définie par l'argument **at**, soient résolues. Une fois que toutes les dépendances sont résolues, le travail est annulé et les travaux ou les flots de travaux qui dépendent du travail annulé sont libérés de la dépendance. Lors de la période pendant laquelle l'annulation est différée, la mention **[Cancel Pend]** (annulation en cours) apparaît dans la colonne Dépendances du travail dans un écran **showjobs**.

Si l'option **;pend** est indiquée et que le travail a déjà été lancé, cette option est ignorée et les travaux ou les flots de travaux qui dépendent du travail annulé sont immédiatement libérés de la dépendance.

Vous pouvez utiliser la commande **rerun** pour relancer les travaux qui ont été annulés ou portant la mention **[Cancel Pend]** (annulation en cours). Vous pouvez aussi ajouter et supprimer des dépendances de des travaux associés à la mention **[Cancel Pend]**.

Pour annuler immédiatement un travail associé à la mention **[Cancel Pend]**, entrez une commande de **libération** du travail ou une commande d'**annulation** sans l'option **;pend**.

En ce qui concerne les travaux parvenus à **échéance**, la mention **[Until]** apparaît dans la colonne Dépendances d'un écran **showjobs** et leurs dépendances ne sont plus évaluées. Si un travail de ce type porte également la mention **[Cancel Pend]**, il n'est annulé que lorsque vous libérez ou supprimez l'échéance **until** ou entrez une autre commande **cancel** sans l'option **;pend**.

Pour arrêter l'évaluation des dépendances, affectez à la priorité d'un travail la valeur zéro, à l'aide de la commande **altpri**. Pour reprendre l'évaluation des dépendances, affectez à la priorité une valeur supérieure à zéro.

Remarque : Dans le cas des dépendances interréseau, l'annulation d'un travail du flot de travaux `EXTERNE` permet de libérer tous les travaux et flots de travaux locaux de la dépendance. Les travaux du flot de travaux `EXTERNES` correspondent à des travaux et à des flots de travaux qui sont signalés comme étant des dépendances interréseaux. Le statut d'une dépendance interréseau n'est pas vérifié à la suite d'une commande **cancel**. Pour plus d'informations, voir «Gestion des dépendances interréseau dans le plan», à la page 676.

Exemples

Pour annuler le travail `report` du flot de travaux `apwkly(0900 02/19/06)` sur le poste de travail `site3`, exécutez la commande suivante :

```
cj site3#apwkly(0900 02/19/06).report
```

Pour annuler le travail setup du flot de travaux mis5(1100 02/10/06) qui n'est pas à l'état ABEND, exécutez la commande suivante :

```
cj mis5(1100 02/10/06).setup~state=abend
```

Pour annuler le travail job3 du flot de travaux sked3(0900 02/19/03) une fois ses dépendances résolues, exécutez la commande suivante :

```
cj sked3(0900 02/19/06).job3;pend
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle qui suit :

1. Dans la barre de navigation, cliquez sur **Statut et état de santé du système** > **Surveillance de la charge de travail** > **Surveillance des travaux**.
2. Sélectionnez **Tous les travaux du plan** ou une autre tâche de surveillance des flots de travaux.
3. Choisissez un nom pour le moteur ou spécifiez les propriété de connexion, puis cliquez sur **OK**.
4. A partir de la table contenant la liste des travaux, sélectionnez un travail et cliquez sur **Plus d'actions** > **Annuler**.

cancel sched

Annule un flot de travaux.

Vous devez avoir un accès ***cancel*** au flot de travaux.

Syntaxe

{**cancel sched** |
cs} *flot_travaux_sélectionné*
[**;pend**]
[**;noask**]

Arguments

flot_travaux_sélectionné
Voir «Sélection de flots de travaux dans les commandes», à la page 388.

pend Annule le flot de travaux uniquement après la résolution de ses dépendances.

noask Indique que l'utilisateur ne doit pas être invité à confirmer avant d'effectuer une opération sur chaque flot de travaux éligible.

Commentaires

Si vous annulez un flot de travaux avant de l'avoir lancé, il n'est pas lancé. Si vous annulez un flot de travaux une fois qu'il est lancé, les travaux en cours sont exécutés jusqu'à la fin mais aucun autre travail n'est lancé.

Si l'option **;pend** n'est pas utilisée, les travaux et les flots de travaux qui dépendent du flot de travaux annulé sont libérés immédiatement de la dépendance.

Si l'option **;pend** est indiquée et que le flot de travaux n'a pas été lancé, l'annulation est différée jusqu'à ce que toutes les dépendances, y compris l'heure d'exécution indiquée par l'argument **at**, soient résolues. Une fois que toutes les

dépendances sont résolues, le flot de travaux est annulé et les travaux ou les flots de travaux dépendants sont libérés de la dépendance. Lorsqu'une **annulation** est différée, la mention **[Cancel Pend]** (Annulation en cours) apparaît dans la colonne Dépendances d'un écran **showschedules**.

Si l'option **;pend** est indiquée et que le flot de travaux a déjà été lancé, les travaux non encore lancés sont annulés et les travaux et flots de travaux dépendants sont libérés de la dépendance.

Pour annuler immédiatement un flot de travaux marqué **[Cancel Pend]** , entrez une commande de **release** pour le flot de travaux ou une autre commande **cancel** sans l'option **;pend**.

Pour arrêter l'évaluation des dépendances, définissez une priorité zéro pour le flot de travaux à l'aide de la commande **altpri**. Pour reprendre l'évaluation des dépendances, affectez à la priorité une valeur supérieure à zéro.

Si le flot de travaux annulé contient des travaux définis avec l'option **every**, seule la dernière instance de ces travaux est signalée comme étant annulée dans un affichage **showjobs**.

Exemples

Pour annuler le flot de travaux `sked1(1200 02/17/06)` sur le poste de travail `site2`, exécutez la commande suivante :

```
cs site2#sked1(1200 02/17)
```

Pour annuler le flot de travaux `mis2(0900 02/19/06)` s'il est à l'état STUCK, exécutez la commande suivante :

```
cs mis2(0900 02/19)+state=stuck
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle qui suit :

1. Dans la barre de navigation, cliquez sur **Statut et état de santé du système** > **Surveillance de la charge de travail** > **Surveillance des flots de travaux**.
2. Sélectionnez **Tous les flots de travaux du plan** ou une autre tâche de surveillance des flots de travaux.
3. Choisissez un nom pour le moteur ou spécifiez les propriété de connexion, puis cliquez sur **OK**.
4. Sélectionnez un flot de travaux et cliquez sur **Plus d'actions** > **Annuler**.

checkhealthstatus

Appelle le service **chkhltst** pour vérifier la connectivité entre le gestionnaire de domaine et les postes de travail. Cette commande vérifie que le fichier Symphony n'est pas endommagé, que les fichiers de boîte aux lettres peuvent être correctement lus par **mailman**, sans erreurs dans l'en-tête de boîte aux lettres, et que la boîte aux lettres n'est pas saturée. Cette commande peut être utile pour diagnostiquer la cause du manque de liaison d'un poste de travail et pour obtenir des suggestions sur la manière de résoudre le problème.

Syntaxe

{**checkhealthstatus** | **chs**} [*poste_de_travail*]

Commentaires

Si le *poste de travail* n'est pas spécifié, le service est lancé localement.

Exemples

Pour vérifier l'état de santé du poste de travail *site1*, lancez la commande suivante :

```
checkhealthstatus site1
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle qui suit :

1. A partir de la barre de navigation, cliquez sur **Statut et état de santé du système** > **Surveillance de l'environnement** > **Surveillance des postes de travail**.
2. Sélectionnez une tâche pour surveiller les postes de travail.
3. Choisissez un nom pour le moteur ou spécifiez les propriété de connexion, puis cliquez sur **OK**.
4. .
5. A partir de la table contenant la liste des postes de travail, sélectionnez le poste de travail pour lequel vous souhaitez vérifier la connectivité et cliquez sur **Plus d'actions** > **Contrôle de l'état de santé...**.

confirm

Confirme la fin d'un travail qui a été planifié à l'aide du mot clé **confirmed.**

Vous devez avoir un accès ***confirm*** au travail.

Syntaxe

{**confirm** | **conf**} *travail_sélectionné*
;{**succ** | **abend**}
[**;noask**]

Arguments

travail_sélectionné
Voir «Sélection de travaux dans les commandes», à la page 379.

succ Confirme que le travail a abouti.

abend
Confirme que le travail a échoué.

noask Indique que l'utilisateur ne doit pas être invité à confirmer avant d'effectuer une opération sur chaque travail éligible.

Commentaires

Il n'est pas nécessaire d'utiliser le mot clé **confirmed** lors de la planification du travail pour que ce dernier passe de l'état ABEND à SUCC. Pour plus

d'informations sur la confirmation d'un travail, voir «confirmed», à la page 244. Pour plus d'informations sur les travaux EXTERNAL, voir «Gestion des dépendances interréseau dans le plan», à la page 676.

tableau 66 ci-après présente l'effet de la commande **confirm** sur les différents états des travaux :

Tableau 66. Changement d'état après exécution de la commande confirm

Etat initial du travail	Etat après confirm : réussite	Etat après confirm : arrêt anormal
READY	aucun effet	aucun effet
HOLD	aucun effet	aucun effet
EXEC	**SUCCP**	**ABENP**
ABENP	**SUCCP**	aucun effet
SUCCP	aucun effet	aucun effet
PEND	**SUCC**	**ABEND**
DONE	**SUCC**	**ABEND**
SUCC	aucun effet	aucun effet
ABEND	**SUCC**	aucun effet
FAIL	aucun effet	aucun effet
SCHED	aucun effet	aucun effet
ERROR (pour les travaux reflet uniquement)	**SUCC**	**ABEND**
tout travail du flot de travaux **EXTERNE**	**SUCC**	**ABEND**

Exemples

Pour générer une confirmation **succ** si le travail job3 du flot de travaux misdly(1200 02/17/06) réussit, exécutez la commande suivante :

```
confirm misdly(1200 02/17/06).job3;succ
```

Pour générer une confirmation **abend** si le travail **234** échoue, exécutez la commande suivante :

```
confirm 234;abend
```

console

Affecte la Tivoli Workload Scheduler Console et définit le niveau de message.

Vous devez avoir un accès ***console*** au poste de travail.

Syntaxe

{**console** | **cons**}
[**sess** | **sys**]
[**;level=***niveau_message*]

Arguments

sess Envoie des messages écran et des invites Tivoli Workload Scheduler à la sortie standard.

sys Arrête l'envoi de messages écran et d'invites Tivoli Workload Scheduler à la sortie standard. Ce processus se déroule automatiquement lorsque vous arrêtez **conman**.

niveau_message
Niveau des messages Tivoli Workload Scheduler envoyés à la console. Indiquez l'un des niveaux suivants :

-1 Il s'agit de la valeur que le produit affecte automatiquement si vous modifiez un quelconque argument pour la console sans réaffecter une valeur à `msglevel`. Avec cette valeur, le produit envoie tous les messages générés par tous les agents et pour toutes les opérations vers la console.

0 Aucun message. Il s'agit de la valeur par défaut sur les agents tolérants aux pannes.

1 Messages d'exception, tels que des invites opérateur et des fins anormales de travail.

2 Niveau 1 avec messages d'aboutissement du flot de travaux.

3 Niveau 2 avec messages d'aboutissement du travail. Il s'agit du niveau par défaut du gestionnaire de domaine maître.

4 Niveau 3 avec messages de lancement de travail.

Commentaires

Si une commande **console** est entrée sans aucune option, l'état en cours de la console s'affiche.

Par défaut, les processus de contrôle Tivoli Workload Scheduler écrivent les messages écran et les invites dans les fichiers de liste standard. Sous UNIX, ils peuvent également être envoyés au daemon **syslog**.

Exemples

Pour lancer l'écriture des invites et des messages de console dans la sortie standard et définir le niveau des messages sur **1**, exécutez la commande suivante :

```
console sess;level=1
```

Pour interrompre l'écriture des invites et des messages de console dans la sortie standard et définir le niveau des messages sur **4**, exécutez la commande suivante :

```
cons sys;l=4
```

Pour afficher l'état courant de la console, exécutez la commande suivante :

```
cons
Console is #J675, level 2, session
```

675 correspond à l'ID processus du shell de l'utilisateur.

continue

Ignore l'erreur de commande suivante.

Syntaxe

{**continue** | **cont**}

Commentaires

Cette commande est utile lorsque des commandes sont entrées de façon non interactive. Elle indique au programme **conman** de continuer à exécuter les commandes même si celle qui a été lancée après **continue** génère une erreur.

Exemples

Pour demander au programme **conman** de continuer à utiliser la commande **rerun** même si la commande **cancel** échoue, exécutez la commande suivante :

```
conman "cont&cancel=176&rerun job=sked5(1200 02/17/06).job3"
```

deldep job

Supprime des dépendances d'un travail.

Vous devez avoir un accès ***deldep*** au travail.

Syntaxe

{**deldep job** | **ddj**} *jobselect*
;*dépendance*[**;**...]
[**;noask**]

Arguments

travail_sélectionné
Voir «Sélection de travaux dans les commandes», à la page 379.

dépendance
Type de dépendance. Indiquez au moins l'un des éléments suivants. Vous pouvez utiliser des caractères génériques dans les arguments *poste_de_travail*, *flot_travaux*, *travail*, *ressource*, *file_name* et *nom_invite*.

at[**=***heure* | *heure_min* | *heure_max* | *heure_min*,*heure_max*]

confirmed

deadline[**=***heure*[**timezone** | **tz** *fuseau_horaire*][**+***n* **days** | *mm/jj[/aa]*]]

every

follows=[*agent_réseau***::**][*poste_de_travail***#**]{*nom_flot_travaux*(*hhmm* [*mm/jj[/aa]*]) [.*travail* | **@**] | *ID_flot_travaux*.*travail***;schedid**}| *travail*[**,**...]

maxdur=[*hhhmm*] [**onmaxdur** *action*]

mindur=[*hhhmm*] [**onmindur** *action*]

needs[**=**[*num*] [*poste_de_travail***#**]*ressource*[**,**...]]

opens[**=**[*poste_de_travail***#**]"*file_name*"[**(***qualifiant***)**][**,**...]]

priority

prompt[**="[:** | **!**]*texte***"** | *nom_invite*[**,**...]]

until[**=***heure* [**timezone**|**tz** *fuseau_horaire*][**+***n* **day**[**s**]] [**;onuntil** *action*]]

noask Indique que l'utilisateur ne doit pas être invité à confirmer avant d'effectuer une opération sur chaque travail éligible.

Commentaires

Si vous supprimez `priority`, le travail retourne à sa priorité planifiée initialement. Lors de la suppression d'une dépendance `opens`, vous pouvez inclure uniquement le nom de fichier de base. **conman** recherche ensuite, sans faire de distinction majuscules/minuscules, les fichiers concordants, sans tenir compte des noms de répertoires. Les dépendances sur tous les fichiers concordants sont supprimées.

Les dépendances supprimées ne restent pas actives lors de l'exécution de la commande **rerun**.

Pour supprimer toutes les dépendances `follows` des travaux d'un flot de travaux spécifique, spécifiez le mot clé `follows` comme suit :

`follows=`*`nom_flot_travaux`*

Dans ce cas, n'utilisez pas de caractère générique (par exemple, `follows=`*`nom_flot_travaux`*`.@`), sinon la commande est rejetée.

Exemples

Pour supprimer une dépendance de ressource du travail `job3` du flot de travaux `sked9(0900 02/19/06)`, exécutez la commande suivante :

```
ddj sked9(0900 02/19/06).job3 ; needs=2 tapes
```

Pour supprimer toutes les dépendances de prédécesseur/successeur externes du flot de travaux `CPUA#TEST(0900 02/19/06)`, exécutez la commande suivante :

```
ddj CPUA#TEST(0900 02/19/06).JOBA ; follows
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer les tâches suivantes :

1. Dans la barre de navigation, cliquez sur **Statut et état de santé du système** > **Surveillance de la charge de travail** > **Surveillance des travaux**.
2. Sélectionnez **Tous les travaux du plan** ou une autre tâche de surveillance des flots de travaux.
3. Choisissez un nom pour le moteur ou spécifiez les propriété de connexion, puis cliquez sur **OK**.
4. Dans le tableau qui contient la liste des travaux, sélectionnez le travail auquel vous souhaitez ajouter une dépendance et cliquez sur **Dépendances...**.
5. Dans le panneau des dépendances, développez une section de dépendance et cliquez sur le bouton correspondant à l'action à effectuer.

deldep sched

Supprime des dépendances d'un flot de travaux.

Vous devez avoir un accès ***deldep*** au flot de travaux.

Syntaxe

{**deldep sched** | **dds**} *jstreamselect*
;*dépendance*[**;**...]
[**;noask**]

Arguments

flot_travaux_sélectionné
Voir «Sélection de travaux dans les commandes», à la page 379.

dépendance
Type de dépendance. Indiquez au moins l'un des éléments suivants. Vous pouvez utiliser des caractères génériques dans les arguments *poste_de_travail*, *nom_flot_travaux*, *nom_travail*, *ressource*, *nom_fichier* et *nom_invite*.

at[=*heure* | *heure_min* | *heure_max* | *heure_min*,*heure_max*]

carryforward

deadline[=*heure*[**timezone** | **tz** *fuseau_horaire*][+*n* **days** | *mm/jj[/aa]*]]

follows=[*agent_réseau*::][*poste_de_travail*#]{*nom_flot_travaux*[*hhmm* [*mm/jj[/aa]*]][.*travail* | **@**] | *ID_flot_travaux.travail*;**schedid**}| *travail*[,...]

limit

needs[=[*num*] [*poste_de_travail*#]*ressource*[,...]]

opens[=[*poste_de_travail*#]"*file_name*"[(*qualifiant*)][,...]]

priority

prompt[="[: | !]*texte*" | *nom_invite*[,...]]

until[=*heure* [**timezone**|**tz** *fuseau_horaire*][+*n* **day**[**s**]] [;**onuntil** *action*]]

noask Indique que l'utilisateur ne doit pas être invité à confirmer avant d'effectuer une opération sur chaque flot de travaux éligible.

Commentaires

Si vous supprimez `priority`, le travail retourne à sa priorité planifiée initialement. Lors de la suppression d'une dépendance `opens`, vous pouvez inclure uniquement le nom de fichier de base. **conman** recherche ensuite, sans faire de distinction majuscules/minuscules, les fichiers concordants, sans tenir compte des noms de répertoires. Les dépendances sur tous les fichiers concordants sont supprimées.

Les dépendances supprimées ne restent pas actives lors de l'exécution de la commande **rerun**.

Exemples

Pour supprimer une dépendance de `ressource` dans le flot de travaux `sked5(0900 02/19/06)`, exécutez la commande suivante :

```
dds sked5(0900 02/19/06);needs=2 tapes
```

Pour supprimer toutes les dépendances de `prédécesseur/successeur` du flot de travaux `sked3(1000 04/19/06)`, exécutez la commande suivante :

```
dds sked3(1000 04/19/06);follows
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche comme suit :

1. Dans la barre de navigation, cliquez sur **Statut et état de santé du système** > **Surveillance de la charge de travail** > **Surveillance des flots de travaux**.

2. Sélectionnez **Tous les flots de travaux du plan** ou une autre tâche de surveillance des flots de travaux.
3. Choisissez un nom pour le moteur ou spécifiez les propriété de connexion, puis cliquez sur **OK**
4. Dans le tableau qui contient la liste des flots de travaux, sélectionnez le flot de travaux auquel vous souhaitez ajouter une dépendance et cliquez sur **Dépendances...** tab.
5. Dans le panneau des dépendances, développez une section de dépendance et cliquez sur le bouton correspondant à l'action à effectuer.

deployconf

Obtient la configuration de surveillance la plus récente pour le moteur de surveillance d'événement sur le poste de travail.

Syntaxe

{**deployconf** | **deploy**} [*domaine***!**]*poste_de_travail*

Arguments

domaine

Spécifie le nom du domaine de destination pour l'opération. Les caractères génériques sont acceptés.

Cet argument est utile lors du déploiement vers plus d'un poste de travail dans un domaine. Par exemple, pour déployer la configuration de surveillance la plus récente vers tous les agents du domaine `AURORABU` utilisez la commande suivante :

```
deploy AURORABU!@
```

Le domaine n'est pas nécessaire si vous n'incluez pas de caractères génériques dans *poste de travail*.

Si le *domaine* n'est pas indiqué et que l'argument *poste_de_travail* comporte des caractères génériques, le domaine par défaut est celui où **conman** est exécuté.

poste de travail

Indique le nom du poste de travail où s'exécute le moteur de surveillance. Les caractères génériques ne sont pas admis.

Commentaires

Si la configuration existante est déjà mise à jour, la commande n'a pas d'effet.

Le droit de `démarrer` des actions sur des objets `cpu` doit être activé dans le fichier de sécurité pour exécuter cette commande.

display

Affiche un fichier de travail ou une définition de flot de travaux.

Si vous indiquez un fichier par son nom, vous devez avoir un accès en lecture au fichier. En ce qui concerne les fichiers de travail et les définitions de flot de travaux, vous devez avoir un accès ***display*** au travail ou au flot de travaux.

Syntaxe

{**display file** | **df**} *file_name* **[;offline]**

{**display job** | **dj**} *jobselect* **[;offline]**

{**display sched** | **ds**} *jstreamselect*
[valid {**at** *date* | **in** *date date*}
[;offline]

Arguments

file_name
Indique le nom du fichier, en général, un fichier script de travail. Ce nom doit être placé entre guillemets (") s'il contient des caractères autres que les suivants : caractères alphanumériques, tirets (-), barres obliques (/), barres obliques inverses (\) et traits de soulignement (_). Les caractères génériques sont acceptés. Le fichier doit être accessible à partir de votre poste de travail de connexion. Utilisez cette option si vous ne souhaitez montrer que le contenu du fichier du script de travail.

travail_sélectionné
Travail dont le fichier de travail est affiché. Voir «Sélection de travaux dans les commandes», à la page 379. Le fichier de travail doit être accessible à partir de votre poste de travail de connexion. Ce mot clé s'applique uniquement au chemin d'accès et au nom de fichier du fichier de script des travaux définis avec l'option **scriptname**.

flot_travaux_sélectionné
Flot de travaux dont la définition est affichée. Voir «Sélection de flots de travaux dans les commandes», à la page 388.

valid Spécifie le jour ou la période au cours de laquelle les instances de flot de travaux à afficher doivent être actives. Cela signifie que l'intervalle de validité de ces instances de flot de travaux doivent contenir la période spécifiée dans l'argument **valid**. Le format utilisé pour *date* dépend de la valeur affectée à la variable ***format_date*** spécifiée dans le fichier `localopts`. Sauf indication contraire, l'instance sélectionnée est l'instance valide aujourd'hui.

offline
Envoie la sortie de la commande au périphérique de sortie de **conman**. Pour plus d'informations sur ce périphérique, voir «Sortie offline», à la page 374.

Exemples

Pour afficher le fichier `c:\maestro\jclfiles\arjob3`, exécutez la commande suivante :

```
df c:\apps\maestro\jclfiles\arjob3
```

Pour afficher hors ligne le fichier script du travail `createpostreports` du flot de travaux `FINALPOSTREPORTS`, exécutez la commande suivante :

```
dj FINALPOSTREPORTS(2359 02/14/13).CREATEPOSTREPORTS
```

Voici un exemple de sortie de cette commande :

```
M235062_99#FINALPOSTREPORTS(2359 02/14/13).CREATEPOSTREPORTS /opt/TWA/TWS/
                                                              CreatePostReports
   #!/bin/sh
  ################################################################
  # Eléments sous licence - Propriété d'IBM
  # Restricted Materials of IBM
  # 5698-WSH
  # (C) Copyright IBM Corp. 1998, 2011 All Rights Reserved.
  # US Government Users Restricted Rights - Use, duplication or
  # disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
  ################################################################
  #@(#) $Id: CreatePostReports.sh,v 1.0

  ##
  ## CreatePostReports message catalog definitions.
  ##

  ##
  ## message set id
  ##
  MAE_CREATEPOSTREPORTS_SET=226
  MAE_COPYRIGHT_SET=234

  ##
  ....
  ...
  ....
  #
  # End
  #
```

Pour afficher la définition du flot de travaux mod, exécutez la commande suivante :

```
ds mod
```

Voici un exemple de sortie de cette commande :

```
Job Stream Name   Workstation       Valid From  Updated On  Locked By
 ---------------  ----------------  ----------  ----------  ----------------
MOD        M235062_99    06/30/2007 03/04/2006 -

SCHEDULE M235062_99#MOD VALIDFROM 06/30/2007
ON RUNCYCLE SCHED1_PREDSIMPLE VALIDFROM 07/18/2007 "FREQ=DAILY;INTERVAL=1"
( AT 1111 )
CARRYFORWARD
FOLLOWS M235062_99#SCHED_FIRST1.@
FOLLOWS M235062_99#SCHED_FIRST.JOB_FTA
PRIORITY 66
:
M235062_99#JOBMDM
 SCRIPTNAME "/usr/acct/scripts/gl1" STREAMLOGON root
 DESCRIPTION "general ledger job1"
 TASKTYPE UNIX
 RECOVERY STOP
 PRIORITY 30
NEEDS 16 M235062_99#JOBSLOTS
PROMPT PRMT3

B236153_00#JOB_FTA
FOLLOWS M235062_99#SCHED_FIRST1.@
FOLLOWS M235062_99#SCHED_FIRST.JOB_FTA
PRIORITY 66
:
M235062_99#JOBMDM
 SCRIPTNAME "/usr/acct/scripts/gl1" STREAMLOGON root
 DESCRIPTION "general ledger job1"
 TASKTYPE UNIX
 RECOVERY STOP
```

```
 PRIORITY 30
NEEDS 16 M235062_99#JOBSLOTS
PROMPT PRMT3

B236153_00#JOB_FTA
 DOCOMMAND "echo pippo" STREAMLOGON root
 DESCRIPTION "general ledger job1"
 TASKTYPE UNIX
 RECOVERY STOP
 FOLLOWS JOBMDM
END
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle décrite dans :

le manuel Dynamic Workload Console - Guide d'utilisation, section Liste des définitions d'objet dans la base de données.

Pour plus d'informations sur la création et l'édition d'objets de planification, voir :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Conception de votre charge de travail".

exit

Quitte le programme de ligne de commande **conman**.

Syntaxe

{**exit** | **e**}

Commentaires

Si vous exécutez le programme de ligne de commande Conman en mode Aide sous UNIX, la commande active le programme **conman** en mode Saisie de commande.

Exemples

Pour quitter le programme de ligne de commande **conman**, exécutez la commande suivante :

```
exit
```

ou

```
e
```

fence

Modifie la priorité minimale de travail sur un poste de travail. Les travaux ne sont pas lancés si leur niveau de priorité est inférieur ou égal à la priorité minimale de travail.

Vous devez avoir un accès ***fence*** au poste de travail.

Syntaxe

{**fence** | **f**} *poste_de_travail*
;pri
[**;noask**]

Arguments

poste de travail
Indique le nom du poste de travail. Par défaut, il s'agit de votre poste de travail de connexion.

pri Indique le niveau de priorité. Vous pouvez entrer une valeur comprise entre **0** et **99**, **hi**, **go** ou **system**. Si vous entrez **system**, la priorité minimale de travail a la valeur zéro.

noask Indique que l'utilisateur ne doit pas être invité à confirmer avant d'effectuer une opération sur chaque poste de travail éligible.

Commentaires

La priorité minimale de travail empêche le lancement des travaux dont le niveau de priorité est faible, quel que soit le niveau de priorité affecté aux flots de travaux dont ils dépendent. Il est donc possible de différer l'exécution des travaux de priorité faible dans des flots de travaux de priorité élevée tout en permettant le lancement des travaux de priorité élevée dans des flots de travaux de priorité faible.

Lors du premier démarrage de Tivoli Workload Scheduler après installation, la priorité minimale de travail a pour valeur zéro. Lorsque vous modifiez la priorité minimale de travail, celle-ci est différée pendant le traitement préproduction au plan de production du jour suivant.

Pour afficher la valeur en cours affectée à la priorité minimale de travail, utilisez la commande **status**.

Exemples

Pour modifier la priorité minimale de travail du poste de travail `site4`, exécutez la commande suivante :

```
fence site4;20
```

Pour modifier la priorité minimale de travail du poste de travail sur lequel vous exécutez le programme **conman**, exécutez la commande suivante :

```
f ;40
```

Pour empêcher Tivoli Workload Scheduler de lancer tous les travaux sur le poste de travail `tx3`, exécutez la commande suivante :

```
f tx3;go
```

Pour modifier la priorité minimale de travail en la définissant sur zéro sur le poste de travail sur lequel vous exécutez le programme **conman**, exécutez la commande suivante :

```
f ;system
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle qui suit :

1. Dans la barre de navigation, cliquez sur **Statut et état de santé du système** > **Surveillance de la charge de travail** > **Surveillance des travaux** ou **Surveillance des flots de travaux**.
2. Sélectionnez une tâche de surveillance.
3. Choisissez un nom pour le moteur ou spécifiez les propriété de connexion, puis cliquez sur **OK**
4. A partir de la table contenant la liste des objets, sélectionnez le travail ou le flot de travaux pour lequel vous souhaitez modifier la priorité et cliquez sur l'onglet **Plus d'actions** > **Priorité minimale...**.
5. Dans le panneau qui s'affiche, définissez la nouvelle priorité et cliquez sur **OK**.

help

Affiche l'aide relative aux commandes. Non disponible sous Windows.

Syntaxe

{**help** | **h**} {*commande*|*mot_clé*}

Arguments

commande

Indique le nom d'une commande système ou **conman**. Pour les commandes **conman**, entrez le nom complet de la commande ; les abréviations et les formats abrégés ne sont pas pris en charge. Pour les commandes comportant deux mots, il suffit d'entrer le premier mot pour afficher l'aide relative à toutes les versions de la commande. Par exemple, si vous entrez **help display**, toutes les informations relatives aux commandes **display file**, **display job**, et **display sched** s'affichent.

mot_clé

Vous pouvez également entrer les mots clés suivants :

COMMANDS
Affiche la liste de toutes les commandes Conman.

SETUPCONMAN
Décrit comment configurer la commande conman.

RUNCONMAN
Comment exécuter conman.

SPECIALCHAR
Décrit les caractères génériques, les délimiteurs et les autres caractères spéciaux que vous pouvez utiliser.

JOBSELECT
Répertorie des informations concernant la sélection des travaux relatifs aux commandes.

JOBSTATES
Répertorie des informations relatives aux états des travaux.

JSSELECT
Répertorie des informations concernant la sélection des flots de travaux relatifs aux commandes.

JSSTATES
Répertorie des informations relatives aux états des flots de travaux.

MANAGEBACKLEVEL
Gestion des travaux et des flots de travaux des agents de niveau antérieur.

Exemples

Pour afficher une liste de toutes les commandes **conman**, exécutez la commande suivante :

```
help commands
```

Pour afficher les informations relatives à la commande **fence**, exécutez la commande suivante :

```
help fence
```

Pour afficher les informations relatives aux commandes **altpri job** et **altpri sched**, exécutez la commande suivante :

```
h altpri
```

Pour afficher des informations relatives aux caractères spéciaux que vous pouvez utiliser, exécutez la commande suivante :

```
h specialchar
```

kill

Arrête un travail qui est en cours d'exécution. Sous UNIX, vous devez utiliser la commande UNIX **kill**. Vous devez avoir un accès ***kill*** au travail.

Syntaxe

{**kill** | **k**} *jobselect*
[;noask]

Arguments

travail_sélectionné
Voir «Sélection de travaux dans les commandes», à la page 379.

noask Indique que l'utilisateur ne doit pas être invité à confirmer avant d'effectuer une opération sur chaque travail éligible.

Commentaires

Le temps de réponse est légèrement plus long car l'opération **kill** n'est pas réalisée par **conman** mais par un processus de production Tivoli Workload Scheduler.

A la suite de la commande kill, les travaux sont à l'état ABEND (arrêt anormal). Les travaux ou les flots de travaux qui dépendent de ces travaux ne sont pas libérés. Ces travaux ne peuvent pas être exécutés à nouveau.

Exemples

Pour supprimer le travail `report` du flot de travaux `apwkly(0600 03/05/06)` sur le poste de travail `site3`, exécutez la commande suivante :

```
kill site3#apwkly(0600 03/05/06).report
```

Pour arrêter le travail numéro `124` qui s'exécute sur le poste de travail `geneva`, exécutez la commande suivante :

```
kill geneva#124
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer les tâches suivantes :

1. Dans la barre de navigation, cliquez sur **Statut et état de santé du système** > **Surveillance de la charge de travail** > **Surveillance des travaux**.
2. Sélectionnez **Tous les travaux du plan** ou une autre tâche de surveillance des flots de travaux.
3. Choisissez un nom pour le moteur ou spécifiez les propriété de connexion, puis cliquez sur **OK**.
4. Dans le tableau qui contient la liste des travaux, sélectionnez le travail auquel vous souhaitez ajouter une dépendance et cliquez sur **Dépendances...**.
5. Dans le panneau des dépendances, développez une section de dépendance et cliquez sur le bouton correspondant à l'action à effectuer.

limit cpu

Modifie le nombre maximal de travaux pouvant être exécutés simultanément sur un poste de travail. Vous devez avoir un accès ***limit*** au poste de travail.

Syntaxe

{**limit cpu** | **lc**} *poste de travail*
;*limite*
[**;noask**]

Arguments

poste de travail
Indique le nom du poste de travail. Les caractères génériques sont acceptés. Par défaut, il s'agit de votre poste de travail de connexion.

limit Indique le nombre de travaux qui peuvent s'exécuter simultanément sur le poste de travail. Les valeurs prises en charge vont de **0** à **1024** et **system**.

Si vous définissez limit cpu à **0** :
- Pour un flot de travaux à l'état **READY**, seuls les travaux avec des valeurs de priorité **hi** et **go** peuvent être lancés sur le poste de travail.
- Pour un flot de travaux avec une valeur de priorité **hi** ou **go**, tous les travaux avec une valeur de priorité différente de 0 peuvent être lancés sur le poste de travail.

Si vous définissez limit cpu à **system**, il n'y a pas de limite au nombre de travaux simultanés sur le poste de travail.

noask Indique que l'utilisateur ne doit pas être invité à confirmer avant d'effectuer une opération sur chaque poste de travail éligible.

Commentaires

Pour afficher le nombre maximal de travaux autorisés sur votre poste de travail de connexion, utilisez la commande **status**.

Lors du premier démarrage de Tivoli Workload Scheduler après installation, le nombre maximal de travaux de poste de travail a pour valeur zéro ; vous devez augmenter ce nombre avant de lancer des travaux. Lorsque vous modifiez la limite, le travail est différé lors du processus de préproduction au plan de processus du jour suivant.

Tivoli Workload Scheduler tente de lancer autant de travaux que possible, dans les limites autorisées. Le nombre de processus qu'il est possible de lancer sur un poste de travail est limité de fait. Si cette limite est atteinte, le système envoie un message indiquant que les ressources système ne sont pas disponibles. Lorsqu'un travail ne peut pas être lancé pour cette raison, il passe à l'état **fail** (échec). Une réduction du nombre maximal de travaux permet d'éviter ces échecs.

Exemples

Pour modifier le nombre maximal de travaux du poste de travail sur lequel vous exécutez le programme **conman**, exécutez la commande suivante :

```
lc ;12
```

Pour modifier le nombre maximal de travaux du poste de travail rx12, exécutez la commande suivante :

```
lc rx12;6
```

Pour définir à 10 le nombre maximal de travaux sur tous les postes de travail appartenant au domaine et aux domaines enfant, exécutez la commande suivante :

```
lc @!@;10
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle qui suit :

1. A partir de la barre de navigation, cliquez sur **Statut et état de santé du système** > **Surveillance de l'environnement** > **Surveillance des postes de travail**.
2. Sélectionnez une tâche pour surveiller les postes de travail.
3. Choisissez un nom pour le moteur ou spécifiez les propriété de connexion, puis cliquez sur **OK**.
4. A partir de la table contenant la liste des postes de travail, sélectionnez un poste de travail et cliquez sur **Plus d'actions** > **Limite...**.

limit sched

Modifie la valeur du mot clé **limit** déterminée dans la définition d'un flot de travaux. Pour des informations supplémentaires sur la définition d'une limite dans la définition de flot de travaux, voir «limit», à la page 262. Vous devez avoir un accès ***limit*** au flot de travaux.

Syntaxe

{**limit sched** |
ls } *flot_travaux_sélectionné*
;*limite*
[**;noask**]

Arguments

flot_travaux_sélectionné
Voir «Sélection de flots de travaux dans les commandes», à la page 388.

limit Indique le nombre maximal de travaux. Vous pouvez entrer une valeur comprise entre **0** et **1024**.

noask Indique que l'utilisateur ne doit pas être invité à confirmer avant d'effectuer une opération sur chaque flot de travaux éligible.

Exemples

Pour modifier le nombre maximal de travaux dans tous les flots de travaux dont le nom comporte la mention `sales`, exécutez la commande suivante :

```
ls sales@;4
```

Pour modifier le nombre maximal de travaux du flot de travaux `CPUA#Job1`, exécutez la commande suivante :

```
ls CPUA#apwkly;6
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle qui suit :

1. Dans la barre de navigation, cliquez sur **Statut et état de santé du système** > **Surveillance de la charge de travail** > **Surveillance des flots de travaux**.
2. Sélectionnez **Tous les flots de travaux du plan** ou une autre tâche de surveillance des flots de travaux.
3. Choisissez un nom pour le moteur ou spécifiez les propriété de connexion, puis cliquez sur **OK**.
4. Sélectionnez un flot de travaux et cliquez sur **Plus d'actions** > **Limite...**.

link

Etablit des liaisons de données entre postes de travail. Dans un réseau Tivoli Workload Scheduler, les agents standard et tolérants aux pannes sont reliés à leurs gestionnaires de domaine et ces derniers sont reliés à leurs gestionnaires de domaine parent. Les agents étendus ne sont pas reliés ; ils communiquent via un hôte.

Vous devez avoir un accès ***link*** au poste de travail cible.

Syntaxe

{**link** | **lk**}
[*domaine***!**]*poste_de_travail*
[**;noask**]

Arguments

domaine

Indique le nom du domaine dans lequel des liaisons sont établies. Les caractères génériques sont acceptés.

Cet argument est utile lors de la connexion de plusieurs postes de travail d'un domaine. Par exemple, pour connecter tous les agents du domaine `stlouis`, utilisez la commande suivante :

```
lk stlouis!@
```

Le domaine n'est pas nécessaire si vous n'incluez pas de caractères génériques dans *poste de travail*.

Si le *domaine* n'est pas indiqué et que l'argument *poste_de_travail* comporte des caractères génériques, le domaine par défaut est celui où **conman** est exécuté.

poste de travail

Indique le nom du poste de travail à connecter. Les caractères génériques sont acceptés.

Cette commande n'est pas prise en charge sur les postes de travail de moteur distant.

noask Indique que l'utilisateur ne doit pas être invité à confirmer avant d'effectuer une opération sur chaque poste de travail éligible.

Commentaires

Si l'option **autolink** a pour valeur **on** dans une définition de poste de travail, ce dernier est connecté automatiquement à chaque démarrage de Tivoli Workload Scheduler. Si l'option **autolink** a pour valeur **off**, vous devez utiliser les commandes **link** et **unlink** pour contrôler le processus de connexion. Pour plus d'informations sur **autolink** voir «Définition de poste de travail», à la page 149.

Lorsque l'utilisateur possède un accès **link** aux postes de travail en cours de connexion, les règles ci-dessous s'appliquent :

- Un utilisateur exécutant **conman** sur le gestionnaire de domaine maître peut relier n'importe quel poste de travail du réseau.
- Un utilisateur exécutant **conman** sur un autre gestionnaire que le gestionnaire de domaine maître peut relier n'importe quel poste de travail de son domaine et des domaines subordonnés. L'utilisateur ne peut pas relier des postes de travail dans les domaines homologues.
- Un utilisateur exécutant **conman** sur un agent peut relier n'importe quel poste de travail dans son domaine local à condition que le poste de travail soit un gestionnaire de domaine ou un hôte. Un agent homologue dans le domaine local ne peut pas être relié.
- Pour relier un domaine subordonné lors de l'exécution de **conman** dans un domaine de niveau supérieur, il n'est pas nécessaire que les liaisons intermédiaires soient ouvertes.

Exemples

La figure 23, à la page 426 et le tableau 67, à la page 426 présentent les liaisons établies par les commandes **link** exécutées par les utilisateurs en différents points du réseau.

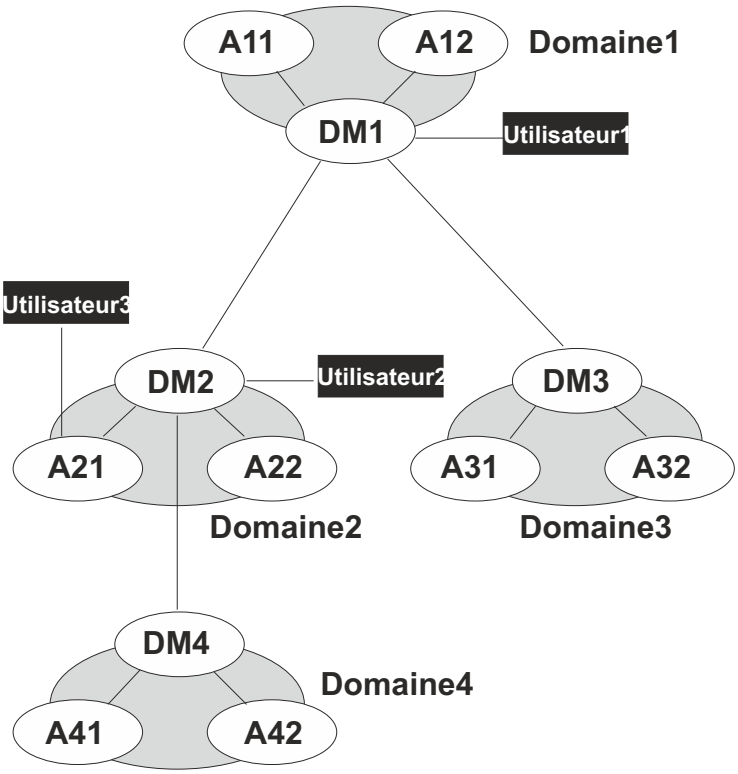


Figure 23. Liaisons réseau

DM*n* sont des gestionnaires de domaine et **A***nn* sont des agents.

Tableau 67. Liaisons ouvertes

Commande	Liaisons établies par Utilisateur1	Liaisons établies par Utilisateur2	Liaisons établies par Utilisateur3
link @!@	Toutes les liaisons sont ouvertes.	DM1-DM2 DM2-A21 DM2-A22 DM2-DM4 DM4-A41 DM4-A42	DM2-A21
link @	DM1-A11 DM1-A12 DM1-DM2 DM1-DM3	DM1-DM2 DM2-A21 DM2-A22 DM2-DM4	DM2-A21
link DOMAIN3!@	DM3-A31 DM3-A32	Non autorisé	Non autorisé
link DOMAIN4!@	DM4-A41 DM4-A42	DM4-A41 DM4-A42	Non autorisé
link DM2	DM1-DM2	Sans objet	DM2-A21
link A42	DM4-A42	DM4-A42	Non autorisé
link A31	DM3-A31	Non autorisé	Non autorisé

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle qui suit :

1. A partir de la barre de navigation, cliquez sur **Statut et état de santé du système** > **Surveillance de l'environnement** > **Surveillance des postes de travail**.
2. Sélectionnez une tâche pour surveiller les postes de travail.
3. Choisissez un nom pour le moteur ou spécifiez les propriété de connexion, puis cliquez sur **OK**.
4. A partir de la table contenant la liste des postes de travail, sélectionnez un poste de travail et cliquez sur **Plus d'actions** > **Lier**.

listsym

Affiche le plan de production (fichiers Symphony) déjà traité.

Syntaxe

{**listsym** | **lis**} [**trial** | **forecast**]
[**;offline**]

Arguments

essai Répertorie les plans d'essai.

Prévision
Répertorie les plans prévisionnels.

offline
Envoie la sortie de la commande au périphérique de sortie de **conman**. Pour plus d'informations sur ce périphérique, voir «Sortie offline», à la page 374.

Résultats

Date du calendrier
Date utilisée pour la sélection des flots de travaux à exécuter.

Date réelle
Date à laquelle **batchman** a lancé l'exécution du fichier `Symphony`.

Heure de début
Heure à laquelle **batchman** a lancé l'exécution du fichier `Symphony`.

Date de journalisation
Date de consignation du plan (fichier `Symphony`) par la commande **stageman**.

Num. d'exécution
Numéro d'exécution attribué au plan (fichier `Symphony`). Il est utilisé en interne pour la synchronisation du réseau Tivoli Workload Scheduler.

Size Nombre d'enregistrements contenus dans le fichier `Symphony`.

Num. de journalisation
Numéro de consignation indiquant l'ordre chronologique des fichiers journaux. Ce numéro peut être utilisé dans une commande **setsym** pour accéder à un fichier journal spécifique.

Nom du fichier
Nom du fichier journal attribué par la commande **stageman**.

Exemples

Pour afficher les fichiers du plan de production, exécutez la commande suivante :

```
listsym
```

Voici un exemple de sortie de cette commande :

```
Job Stream  Actual Start   Log   Run      Log
Date     Date  Time    Date   Num  Size  Num  Filename
03/05/06  03/05/06 21:06  03/05/06   42  534   1 M200603052111  Exp
03/04/06  03/04/06 15:59  03/05/06   41  463   2 M200603052106  Exp
03/04/06  03/04/06 15:51  03/04/06   40  362   3 M200603041559  Exp
03/04/06  03/04/06 14:31  03/04/06   39  460   4 M200603041551  Exp
03/04/06  03/04/06 14:26  03/04/06   38  436   5 M200603041431  Exp
03/04/06  03/04/06 14:24  03/04/06   37  436   6 M200603041426  Exp
03/04/06  03/04/06 14:19  03/04/06   36  436   7 M200603041424  Exp
03/04/06  03/04/06 14:17  03/04/06   35  436   8 M200603041419  Exp
03/04/06  03/04/06 14:17  03/04/06   34  364   9 M200603041417  Exp
```

Pour afficher les fichiers contenant des plans d'essai, exécutez la commande suivante :

```
listsym trial
```

Voici un exemple de sortie de cette commande :

```
Job Stream  Actual Start   Log   Run      Log
Date     Date  Time    Date   Num  Size  Num  Filename
03/03/06          03/03/06   0  126   1 Tpippo      Exp
03/03/06          03/03/06   0  1850   2 Tangelo2     Exp
03/03/06          03/03/06   0  1838   3 Tangelo1     Exp
```

Pour afficher les fichiers contenant les plans prévisionnels, exécutez la commande suivante :

```
listsym forecast
```

Voici un exemple de sortie de cette commande :

```
Job Stream  Actual Start   Log   Run      Log
Date     Date  Time    Date   Num  Size  Num  Filename
03/03/06          03/03/06   0   62   1 Fpluto      Exp
```

Voir aussi

Dans Dynamic Workload Console :

1. A partir de la barre de navigation, cliquez sur **Planification** > **Prévision de la charge de travail** > **Gestion des plans disponibles**.
2. Choisissez un nom pour le moteur ou spécifiez les propriété de connexion, puis cliquez sur **OK**.
3. Cliquez sur un type de plan ou saisissez le nom de fichier du plan souhaité
4. Cliquez sur **Afficher liste des plans**.

recall

Affiche des invites qui attendent une réponse.

Vous devez avoir un accès ***display*** aux invites.

Syntaxe

{**recall** | **rc**}
[*poste de travail*]
[**;offline**]

Arguments

poste de travail
Indique le nom du poste de travail sur lequel l'invite a été émise. Si aucun poste de travail n'est indiqué, seules les invites du poste de travail de connexion et les invites globales s'affichent.

offline
Envoie la sortie de la commande au périphérique de sortie de **conman**. Pour plus d'informations sur ce périphérique, voir «Sortie offline», à la page 374.

Résultats

Etat Etat de l'invite. Les invites en attente sont toujours à l'état ASKED (demandée).

Message ou invite
Pour les invites nommées, numéro de message, nom de l'invite et texte du message. Pour les invites non nommées, numéro de message, nom du travail ou du flot de travaux et texte du message.

Exemples

Pour afficher les invites en attente générées sur le poste de travail où vous exécutez le programme **conman**, exécutez la commande suivante :

```
recall
```

ou :

```
rc
```

Pour afficher les invites en attente du poste de travail `site3`, exécutez la commande suivante :

```
rc site3
```

Pour afficher les invites en attente sur tous les postes de travail et envoyer les résultats au périphérique hors ligne de **conman**, exécutez la commande suivante :

```
rc @;offline
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle qui suit :

1. A partir de la barre de navigation, cliquez sur **Statut et état de santé du système** > **Surveillance de la charge de travail** > **Surveillance des invites**.
2. Sélectionnez **Toutes les invites du plan**, qui répertorie toutes les invites quel que soit leur statut, ou créer et sélectionner une autre tâche prédéfinie, qui répertorie uniquement les invites dont le statut est ASKED.
3. Choisissez un nom pour le moteur ou spécifiez les propriété de connexion, puis cliquez sur **OK**.

redo

Modifie et exécute à nouveau la commande précédente.

Syntaxe

{**redo** | **red**}

Contexte

Si vous exécutez la commande **redo**, le programme **conman** affiche la dernière commande, que vous pouvez modifier et réexécuter à votre guise. Utilisez la barre d'espacement pour placer le curseur sous le caractère à modifier, puis entrez les instructions ci-dessous.

Instructions

d[*rép***]** Supprime le caractère situé au-dessus de la lettre **d**. Cette commande peut être suivie par d'autres instructions.

i*texte* Insère du texte avant le caractère situé au-dessus de la lettre **i**.

r*texte* Remplace un ou plusieurs caractères par du texte en commençant par le caractère au-dessus du **r**. Le remplacement est implicite si aucune autre instruction n'est entrée.

>*texte* Ajoute du texte à la fin de la ligne.

>d[*rép* | *texte* **]**
Supprime les caractères en fin de ligne. Cette commande peut être suivie d'une autre instruction ou d'un autre texte.

>r*texte* Remplace des caractères à la fin de la ligne par du texte.

Exemples d'instruction

ddd Supprime les trois caractères au-dessus des **d**.

iabc Insère **abc** avant le caractère situé au-dessus de la lettre **i**.

rabc Remplace les trois caractères, en commençant par celui qui est situé au-dessus du **r**, par **abc**.

abc Remplace les trois caractères situés au-dessus de **abc**, par **abc**.

d diabc
Supprime le caractère au-dessus du premier **d**, ignore un caractère, supprime le caractère au-dessus du second **d**, puis le remplace par **abc**.

>abc Ajoute **abc** à la fin de la ligne.

>ddabc
Supprime les deux derniers caractères de la ligne, puis les remplace par **abc**.

>rabc Remplace les trois derniers caractères de la ligne par **abc**.

Exemples

Pour insérer un caractère, exécutez la commande suivante :

```
redo
setsm 4
    iy
setsym 4
```

Pour remplacer un caractère, exécutez la commande suivante :

```
redo
setsym 4
       5
setsym 5
```

release job

Libère les travaux des dépendances.

Vous devez avoir un accès ***release*** au travail.

Syntaxe

{**release job** |
rj} *travail_sélectionné*
[**;***dépendance*[**;**...]]
[**;noask**]

Arguments

travail_sélectionné
Indique le ou les travaux à libérer. Voir «Sélection de travaux dans les commandes», à la page 379.

dépendance
Type de dépendance. Vous pouvez indiquer l'un des éléments suivants. Vous pouvez utiliser des caractères génériques dans les arguments *poste_de_travail, nom_flot_travaux, nom_travail, ressource, nom_fichier* et *nom_invite*.

at[**=***heure* | *heure_min* | *heure_max* | *heure_min*,*heure_max*]

confirmed

deadline[**=***heure*[**timezone** | **tz** *fuseau_horaire*][**+***n* **days** | *mm/jj[/aa]*]]

every

follows=[*agent_réseau***::**][*poste_de_travail***#**]{*nom_flot_travaux*[*hhmm* [*mm/jj[/aa]*]][*.travail* | **@**] | *ID_flot_travaux.travail***;schedid**}| *travail*[**,**...]

needs[**=**[*num*] [*poste_de_travail***#**]*ressource*[**,**...]]

opens[**=**[*poste_de_travail***#**]"*file_name*"[**(***qualifiant***)**][**,**...]]

priority

prompt[**="[:** | **!**]*texte***"** | *nom_invite*[**,**...]]

until[**=***heure* [**timezone**|**tz** *fuseau_horaire*][**+***n* **day**[**s**]] [**;onuntil** *action*]]

noask Indique que l'utilisateur ne doit pas être invité à confirmer avant d'effectuer une opération sur chaque travail éligible.

Commentaires

Lors de la libération d'une dépendance `opens`, vous pouvez inclure uniquement le nom de fichier de base. **conman** recherche ensuite, sans faire de distinction majuscules/minuscules, les fichiers concordants, sans tenir compte des noms de répertoires. Les dépendances sur tous les fichiers concordants sont libérées.

En ce qui concerne les dépendances `needs`, le travail libéré reçoit le nombre requis d'unités de la ressource, même si celles-ci risquent de ne pas être disponibles. Cela peut entraîner l'affichage par les unités disponibles d'un nombre négatif dans **showresources**.

Lorsque vous libérez un travail d'une dépendance de type `priority`, celui-ci reprend sa priorité d'origine planifiée.

Les dépendances libérées restent en vigueur lors de l'exécution de la commande **rerun**.

Exemples

Pour libérer le travail `job3` du flot de travaux `ap(1000 03/05/06)` de toutes ses dépendances, exécutez la commande suivante :

```
rj ap(1000 03/05/06).job3
```

Pour libérer tous les travaux du poste de travail `site4` de ses dépendances sur une invite baptisée `glprmt`, exécutez la commande suivante :

```
rj site4#@.@;prompt=glprmt
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle qui suit :

1. Dans la barre de navigation, cliquez sur **Statut et état de santé du système** > **Surveillance de la charge de travail** > **Surveillance des travaux**.
2. Sélectionnez **Tous les travaux du plan** ou une autre tâche de surveillance des flots de travaux.
3. Choisissez un nom pour le moteur ou spécifiez les propriété de connexion, puis cliquez sur **OK**.
4. A partir de la table contenant la liste des travaux, sélectionnez un travail et cliquez sur **Plus d'actions** > **Libérer**.

release sched

Libère les flots de travaux des dépendances.

Vous devez avoir un accès ***release*** au flot de travaux.

Syntaxe

{**release sched**
| **rs**} *flot_travaux_sélectionnés*
[**;***dépendance*[**;**...]]
[**;noask**]

Arguments

flot_travaux_sélectionné
Voir «Sélection de flots de travaux dans les commandes», à la page 388.

dépendance
Type de dépendance. Indiquez l'un des éléments suivants. Vous pouvez utiliser des caractères génériques dans les arguments *poste_de_travail*, *flot_travaux*, *travail*, *ressource*, *nom_fichier* et *nom_invite*.

at[=*heure* | *heure_min* | *heure_max* | *heure_min*,*heure_max*]

carryforward

deadline[=*heure*[**timezone** | **tz** *fuseau_horaire*][+*n* **days** | *mm/jj[/aa]*]]

follows=[*agent_réseau*::][*poste_de_travail*#]{*nom_flot_travaux*[*hhmm* [*mm/jj[/aa]*]][.*travail* | **@**] | *ID_flot_travaux.travail*;**schedid**} | *travail*[,...]

limit

needs[=[*num*] [*poste_de_travail*#]*ressource*[,...]]

opens[=[*poste_de_travail*#]"*file_name*"[(*qualifiant*)][,...]]

priority

prompt[="[: | !]*texte*" | *nom_invite*[,...]]

until[=*heure* [**timezone** | **tz** *fuseau_horaire*][+*n* **day**[**s**]] [;**onuntil** *action*]]

noask Indique que l'utilisateur ne doit pas être invité à confirmer avant d'effectuer une opération sur chaque flot de travaux éligible.

Commentaires

Lors de la suppression d'une dépendance `opens`, vous pouvez inclure uniquement le nom de fichier de base. **conman** recherche ensuite, sans faire de distinction majuscules/minuscules, les fichiers concordants, sans tenir compte des noms de répertoires. Les dépendances sur tous les fichiers concordants sont libérées.

En ce qui concerne les dépendances `needs`, le flot de travaux libéré reçoit le nombre requis d'unités de la ressource, même si celles-ci risquent de ne pas être disponibles. Cela peut entraîner l'affichage par les unités disponibles d'un nombre négatif dans **showresources**.

Lorsque vous libérez un flot de travaux d'une dépendance de type `priority`, le flot de travaux reprend sa priorité d'origine.

Dans certaines circonstances, la commande **deldep**, une fois soumise, est susceptible de s'exécuter avec succès, même si elle est à nouveau envoyée à **batchman**. Pour plus d'informations, voir «Traitement des commandes Conman», à la page 378.

Exemples

Pour libérer l'instance de flot de travaux ayant l'ID *ID_flot_travaux* `0AAAAAAAAAAAABSE` de toutes ses dépendances, exécutez la commande suivante :

```
rs 0AAAAAAAAAAAABSE; schedid
```

Pour libérer le flot de travaux `sked5(1105 03/07/06)` de toutes ses dépendances `opens`, exécutez la commande suivante :

```
rs sked5(1105 03/07/06);opens
```

Pour libérer tous les flots de travaux du poste de travail `site3` de ses dépendances dans le flot de travaux `main#sked23`, exécutez la commande suivante :

```
rs site3#@;follows=main#sked23
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche comme suit :

1. Dans la barre de navigation, cliquez sur **Statut et état de santé du système** > **Surveillance de la charge de travail** > **Surveillance des flots de travaux**.
2. Sélectionnez **Tous les flots de travaux du plan** ou une autre tâche de surveillance des flots de travaux.
3. Choisissez un nom pour le moteur ou spécifiez les propriété de connexion, puis cliquez sur **OK**.
4. Sélectionnez un flot de travaux et cliquez sur **Plus d'actions** > **Libérer**.

reply

Répond à une invite de travail ou de flot de travaux.

Vous devez avoir un accès ***reply*** à l'invite nommée ou globale. Pour répondre à une invite non nommée, vous devez détenir un accès ***reply*** aux invites et ***reply*** au travail ou flot de travaux associés.

Syntaxe

{**reply** | **rep**}
{ *nom_invite* | [*poste_de_travail***#**]*num_message*}
;*reply*
[**;noask**]

Arguments

nom_invite
Indique le nom d'une invite globale. Les caractères génériques sont acceptés.

poste de travail
Indique le nom du poste de travail sur lequel une invite non nommée a été émise.

msgnum
Indique le numéro de message d'une invite non nommée. Vous pouvez afficher des numéros de message à l'aide des commandes **recall** et **showprompts**.

reply Indique la réponse : **O** pour oui ou **N** pour non.

noask Indique que l'utilisateur ne doit pas être invité à confirmer avant d'effectuer une opération sur chaque invite éligible.

Commentaires

Si la réponse est **O**, les dépendances sur l'invite sont respectées. Si la réponse est **N**, les dépendances ne sont pas respectées et l'invite n'est pas émise de nouveau.

Vous pouvez répondre aux invites avant qu'elles soient émises. La commande **showprompts** permet d'afficher toutes les invites, qu'elles aient été émises ou non.

Exemples

Pour répondre **O** à l'invite globale `arpmt`, exécutez la commande suivante :

```
reply arprmt;y
```

Pour répondre **N** au message **24** sur le poste de travail `site4`, exécutez la commande suivante :

```
rep site4#24;n
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle qui suit :

1. A partir de la barre de navigation, cliquez sur **Statut et état de santé du système** > **Surveillance de la charge de travail** > **Surveillance des invites**.
2. Sélectionnez **Toutes les invites du plan** ou une autre tâche pour surveiller les invites.
3. Choisissez un nom pour le moteur ou spécifiez les propriété de connexion, puis cliquez sur **OK**.
4. A partir de la table des résultats, sélectionnez une invite et cliquez sur **Répondre oui** ou **Répondre non**.

rerun

Lance une nouvelle exécution d'un travail.

Vous devez avoir un accès ***rerun*** au travail.

Syntaxe

{**rerun** | **rr**}
travail_sélectionné
[**;from=**[*poste_de_travail***#**]*travail*
[**;at=***heure*]
[**;pri=***pri*]]
[**;step=***étape*]
[**;noask**]

Arguments

travail_sélectionné
Indique le nom d'un ou de plusieurs travaux. Les caractères génériques sont acceptés.

from=[*poste_de_travail***#**]*travail*
Indique le nom d'un travail défini dans la base de données dont le fichier ou la commande va être exécuté à la place du travail indiqué par l'argument *travail_sélectionné*.

*poste_de_travail***#**
Indique le nom du poste de travail sur lequel le travail **from** s'exécute. Par défaut, le poste de travail sur lequel **conman** est en cours d'exécution est utilisé.

travail Indique le nom de la définition de travail donnée par l'argument **from**. Les types de noms de travail suivants ne sont pas admis :

- Les noms des travaux soumis à l'aide des commandes **submit file** et **submit docommand**.
- Les noms d'alias des travaux soumis à l'aide de la commande **submit job**.

Pour utiliser l'argument **from**, vous devez avoir accès à la base de données à partir de l'ordinateur sur lequel **conman** est en cours d'exécution.

at=*heure*
Spécifie l'heure de démarrage de réexécution du travail, exprimée comme suit :

*hhmm [**timezone** | **tz** fuseau_horaire]* [**+***n* **days** | *date*]

où :

hhmm Heure et minute.

+*n* **days**
Occurrence suivante de *hhmm* en *n* nombre de jours.

date Occurrence suivante de *hhmm* à la *date* indiquée, exprimée au format *mm/jj[/aa]*.

timezone | **tz** *fuseau_horaire*
Nom du fuseau horaire du travail. Voir Chapitre 16, «Gestion des fuseaux horaires», à la page 653 pour connaître les noms valides.

pri=*pri*
Indique la priorité à affecter au travail réexécuté. Si aucune priorité n'est indiquée, le travail reçoit la même priorité que le travail d'origine.

step=*étape*
Indique que le travail est réexécuté à l'aide de ce nom, à la place du nom de travail d'origine. Pour plus d'informations, voir “Usage Notes”.

noask Indique que l'utilisateur ne doit pas être invité à confirmer avant d'effectuer une opération sur chaque travail éligible.

Commentaires

Vous pouvez réexécuter des travaux qui sont à l'état SUCC, FAIL ou ABEND. Un travail réexécuté est placé dans le même flot de travaux que le travail d'origine, et hérite des dépendances de ce dernier. Si vous réexécutez un travail répétitif (**every**), le travail sera réexécuté selon la même fréquence que le travail d'origine.

Remarque : Vous pouvez soumettre la commande **rerun** pour les travaux du flot de travaux `EXTERNES` qui sont à l'état ERROR. Les travaux du flot de travaux `EXTERNES` correspondent à des travaux et à des flots de travaux qui sont signalés comme étant des dépendances interréseaux. Après toute exécution de la commande **rerun**, le travail passe immédiatement à l'état **extrn** et **conman** commence à vérifier l'état.

Lorsque l'option **;from** est utilisée, le nom du travail réexécuté est fonction de la valeur de l'option globale **enRetainNameOnRerunFrom**. Si l'option a pour valeur **O**, les travaux réexécutés conservent les noms de travaux d'origine. Si l'option a pour valeur **N**, les noms de travaux **from** sont attribués aux travaux réexécutés. Pour plus d'informations, voir le manuel Tivoli Workload Scheduler - *Guide d'administration*.

Dans les écrans **conman**, les travaux réexécutés qui s'affichent portent la mention **>>rerun as**. Pour faire référence à un travail réexécuté dans une autre commande telle que **altpri**, vous devez utiliser le nom de travail d'origine.

Lors de la réexécution d'un travail à l'aide de l'option **;step**, le travail s'exécute sous le nom *step* à la place de son nom d'origine. Dans un script de travail, vous pouvez utiliser la commande **jobinfo** pour revenir au nom du travail, puis exécuter le script différemment pour chaque itération. Dans le script UNIX suivant, la commande **jobinfo** permet d'affecter à une variable **STEP** le nom qui a été utilisé pour exécuter le travail. Le programme utilise ensuite la variable **STEP** pour déterminer le mode d'exécution du script.

```
...
MPATH=`maestro`
STEP=`$MPATH/bin/jobinfo nom_travail`
if [$STEP = JOB3]
  then
  ...
  STEP=JSTEP1
  fi
if [$STEP = JSTEP1]
  then
  ...
  STEP=JSTEP2
  fi
if [$STEP = JSTEP2]
  then
  ...
  fi
...
```

Dans les écrans **conman**, les travaux réexécutés avec l'option **;step** s'affichent avec la mention **>>rerun step**.

Pour plus d'informations sur **jobinfo**, voir «jobinfo», à la page 559.

Exemples

Pour réexécuter le travail `job4` du flot de travaux `sked1` sur le poste de travail `main`, exécutez la commande suivante :

```
rr main#sked1.job4
```

Pour réexécuter le travail `job5` du flot de travaux `sked2` en utilisant la définition du travail `jobx` dont l'heure d'exécution **at** et la priorité sont respectivement définies sur 18:30 et **25**, exécutez la commande suivante :

```
rr sked2.job5;from=jobx;at=1830;pri=25
```

Pour réexécuter le travail `job3` du flot de travaux `sked4` en utilisant le nom de travail `jstep2`, exécutez la commande suivante :

```
rr sked4.job3;step=jstep2
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle qui suit :

1. Dans la barre de navigation, cliquez sur **Statut et état de santé du système** > **Surveillance de la charge de travail** > **Surveillance des travaux**.
2. Sélectionnez **Tous les travaux du plan** ou une autre tâche de surveillance des flots de travaux.

3. Choisissez un nom pour le moteur ou spécifiez les propriété de connexion, puis cliquez sur **OK**.
4. A partir de la table contenant la liste des travaux, sélectionnez un travail et cliquez sur **Plus d'actions** > **Réexécuter...**.

resetFTA

Génère un fichier `Sinfonia` mis à jour et envoie celui-ci à l'agent tolérant aux pannes sur lequel le fichier `Symphony` a été endommagé.

Remarque : La suppression et le remplacement du fichier `Symphony` entraînent la perte de certaines données, par exemple les événements relatifs aux statuts de travail ou le contenu du message Mailbox.msg et des files d'attente de messages tomaster.msg. Si des informations d'état concernant un travail figuraient dans ces files d'attente, ce travail est réexécuté. Il est recommandé d'appliquer cette commande avec précaution.
Lors du processus, les fichiers suivants sont déplacés vers le répertoire *rép_base_TWA*/TWS/tmp :

- Appserverbox.msg
- clbox.msg
- Courier.msg
- Intercom.msg
- Mailbox.msg
- Monbox.msg
- Moncmd.msg
- Symphony
- Sinfonia

Avant l'exécution de la commande, un message d'information s'affiche pour demander une confirmation et s'assurer que la commande n'est pas émise par erreur. Si l'un des fichiers cible ne peut pas être déplacé parce qu'il est utilisé par un autre processus (par exemple, si le processus mailman est toujours en cours d'exécution), l'opération n'est pas effectuée et un message d'erreur s'affiche.

Autorisation

Vous devez avoir l'accès **RESETFTA** à l'agent tolérant aux pannes que vous souhaitez réinitialiser.

Syntaxe

UC **resetFTA**

Arguments

cpu Représente l'agent tolérant aux pannes à réinitialiser.

Cette commande n'est pas disponible dans Dynamic Workload Console.

Exemples

Pour réinitialiser l'agent tolérant aux pannes avec le nom omaha, exécutez la commande suivante :

```
resetFTA omaha
```

Voir aussi

Pour plus d'informations concernant la procédure de reconstitution d'un agent tolérant aux pannes, consultez la section relative à la procédure de reconstitution d'un agent tolérant aux pannes dans *Tivoli Workload Scheduler - Guide d'identification et de résolution des problèmes*..

resource

Modifie le nombre total d'unités d'une ressource.

Vous devez avoir un accès ***resource*** à la ressource.

Syntaxe

{**resource** | **reso**} [*poste_de_travail***#**]
*ressource***;***num*
[**;noask**]

Arguments

poste de travail
Indique le nom du poste de travail sur lequel la ressource est définie. Par défaut, le poste de travail sur lequel **conman** est en cours d'exécution est utilisé.

ressource
Indique le nom de la ressource.

nombre Indique le nombre total d'unités de ressources. Les valeurs correctes sont comprises entre **0** et **1024**.

noask Indique que l'utilisateur ne doit pas être invité à confirmer avant d'effectuer une opération sur chaque ressource éligible.

Exemples

Pour remplacer le nombre d'unités de la ressource `tapes` par **5**, exécutez la commande suivante :

```
resource tapes;5
```

Pour remplacer le nombre d'unités de la ressource `jobslots` du poste de travail `site2` par **23**, exécutez la commande suivante :

```
reso site2#jobslots;23
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche comme suit :

1. A partir de la barre de navigation, cliquez sur **Statut et état de santé du système** > **Surveillance de la charge de travail** > **Surveillance des ressources**.
2. Sélectionnez **Toutes les ressources du plan** ou une autre tâche pour surveiller les ressources.
3. Choisissez un nom pour le moteur ou spécifiez les propriété de connexion, puis cliquez sur **OK**.
4. A partir de la table des résultats, sélectionnez une ressource et cliquez sur **Modifier des unités...**.

setsym

Sélectionne un fichier d'archivage de plan de production. Les écrans de commande ultérieurs affichent le contenu du plan de production archivé. Les informations contenues dans un fichier d'archivage de plan de production ne peuvent pas être modifiées.

Syntaxe

{**setsym** | **set**} [**essai** | **prévision**] [*num_fichier*]

Arguments

Essai Répertorie les plans d'essai.

Prévision
Répertorie les plans prévisionnels.

num_fichier
Indique le numéro du fichier d'archivage de plan de production. Si aucun numéro de fichier journal n'est indiqué, le pointeur retourne en position zéro (plan de production en cours (`Symphony`)). Utilisez la commande **listsym** pour afficher la liste des numéros de fichiers d'archivage.

Exemples

Pour sélectionner le fichier archive du plan de production **5**, exécutez la commande suivante :

```
setsym 5
```

Pour sélectionner le plan de production courant (fichier `Symphony`), exécutez la commande suivante :

```
set
```

Voir aussi

Dans Dynamic Workload Console :

1. A partir de la barre de navigation, cliquez sur **Planification** > **Prévision de la charge de travail** > **Gestion des plans disponibles**.
2. Choisissez un nom pour le moteur ou spécifiez les propriété de connexion, puis cliquez sur **OK**.
3. Cliquez sur **Plans archivés** ou entrez un nom de fichier pour le plan.
4. Cliquez sur **Afficher liste des plans**.

showcpus

Affiche des informations relatives aux postes de travail et aux liaisons.

L'information affichée est mise à jour uniquement tant que Tivoli Workload Scheduler (batchman) s'exécute sur les postes de travail. Le fait que **`batchman`** soit actif ou non est vérifié sur l'écran par le message `Batchman LIVES` ou `Batchman down` lorsque vous émettez la commande `conman start`.

Vous devez avoir un accès ***list*** à l'objet qui est affiché si l'option ***enListSecChk*** a été définie sur **yes** sur le gestionnaire de domaine maître lorsque le plan de production a été créé ou étendu.

Syntaxe

{**showcpus** | **sc**} [[*domaine***!**]*poste de travail*]
[;info|;link]
[;offline]

{**showcpus** | **sc**} [[*domaine***!**]*poste de travail*] **[;getmon]**

Arguments

domaine
Indique le nom d'un domaine. Par défaut, le programme utilise le domaine dans lequel la commande est en cours d'exécution.

poste de travail
Indique le nom d'un poste de travail. Par défaut, le programme utilise le poste de travail sur lequel la commande est en cours d'exécution. Si vous omettez le domaine et le poste de travail, le programme génère les résultats suivants :

- La commande suivante affiche tous les postes de travail figurant dans le domaine du poste de travail où la commande a été exécutée, ainsi que les gestionnaire de domaine connectés si le poste de travail est un gestionnaire de domaine.

  ```
  conman "sc"
  ```

- La commande suivante affiche tous les postes de travail figurant dans le domaine du poste de travail où la commande a été exécutée, sans les gestionnaire de domaine connectés.

  ```
  conman "sc @"
  ```

info Affiche les informations au format **info**.

link Affiche les informations au format **link**.

offline
Envoie la sortie de la commande au périphérique de sortie de **conman**. Pour plus d'informations sur ce périphérique, voir «Sortie offline», à la page 374.

getmon
Renvoie la liste des règles d'événement définies pour le programme de surveillance s'exécutant sur le poste de travail spécifié selon le format suivant :

```
<nom_règle>::<eventProvider>#<eventType>:<scope>
```

La portée de la règle est automatiquement générée : informations sur les attributs de règle, tels que les postes de travail où elle est utilisée, un travail ou un nom de fichier, etc.

L'en-tête de la sortie contient également l'horodatage de la dernière génération du module de configuration de règle.

Remarque : Cette option n'est pas valide sur les postes de travail Tivoli Dynamic Workload Broker (ou les gestionnaires de domaine dynamique). Dans ce cas, vous pouvez récupérer les informations concernant les règles actives définies dans ces postes de travail dans le fichier *TWA_home*\TWS\monconf\TWSObjectsMonitor.cfg sur le gestionnaire de domaine maître.

Résultats

Lorsque le paramètre `getmon` n'est pas utilisé, la sortie de la commande est produite selon trois formats **standard**, **info** et **lien**. La valeur par défaut est **standard**. La signification des caractères affichés dépend du type de format que vous sélectionnez.

Lorsque vous effectuez l'exécution sur un poste de travail dont la version de Tivoli Workload Scheduler est antérieure à la version 8.6, la commande **sc** présente comme agents tolérants aux pannes les types de postes de travail introduits avec le pool, le pool dynamique, l'agent et le moteur distant Tivoli Workload Scheduler version 8.6.

Lorsque le paramètre `getmon` est utilisé, la liste des règles est fournie dans une sortie séparée.

Exemples

1. Pour afficher les informations relatives au poste de travail sur lequel vous exécutez le programme **conman** au format **info**, exécutez la commande suivante :

   ```
   showcpus ;info
   ```

   Voici un exemple de sortie de cette commande :

   ```
   CPUID         VERSION   TIME ZONE               INFO
   MASTER     8.6.0.0 US/Pacific       Linux 2.6.5-7.191-s390 #1 SM
   FTA1      8.6.0.0             Linux 2.4.9-e.24 #1 Tue May
   FTA2      8.6.0.0             HP-UX B.11.11 U 9000/785
   ```

2. Pour afficher les informations de liaison relatives à tous les postes de travail, exécutez la commande suivante :

   ```
   sc @!@;link
   ```

   Voici un exemple de sortie :

   ```
   CPUID       HOST      FLAGS  ADDR  NODE
   MASTER   MASTER   AF T 51099 9.132.239.65
   FTA1    FTA1    AF T 51000 CPU235019
   FTA2    FTA2    AF T 51000 9.132.235.42
   BROKER1  MASTER   A T 51111 9.132.237.17
   ```

3. Pour afficher les informations relatives au poste de travail, exécutez la commande suivante :

   ```
   showcpus
   ```

   Si vous exécutez la commande dans un environnement où la liaison principale du poste de travail avec son domaine ou gestionnaire supérieur n'est pas active, le programme génère la sortie suivante :

   ```
   CPUID   RUN    NODE     LIMIT FENCE  DATE     TIME    STATE    METHOD DOMAIN
   MASTER 360 *WNT MASTER  10  0 03/05/2010 1348  I J  E     MASTERDM
   FTA1  360 WNT FTA   10  0 03/05/2010 1348 FTI JW M      MASTERDM
   FTA2  360 WNT FTA   10  0 03/05/2010 1348 FTI JW M      MASTERDM
   FTA3  360 WNT MANAGER 10  0 03/05/2010 1348 LTI JW M      DOMAIN1
   FTA4  360 WNT FTA   10  0 03/05/2010 1348 F I J M      DOMAIN1
   FTA5  360 WNT FTA   10  0 03/05/2010 1348  I J M      DOMAIN1
   SA1   360 WNT S-AGENT 10  0 03/05/2010 1348 F I J M      DOMAIN1
   XA_FTA4 360 OTHR X-AGENT 10  0 03/05/2010 1348 L I J M      DOMAIN1
   FTA6  360 WNT MANAGER 10  0 03/05/2010 1348 F I J M      DOMAIN2
   FTA7    360  WNT  FTA      10    0 03/05/2010 1349 F I J  M           DOMAIN2
   FTA7    360  WNT  FTA      10    0 03/05/2010 1349 F I J  M           DOMAIN2
   BROKER 360 OTHR BROKER  10  0 03/05/2010 1349 LTI JW       MASTERDM
   ```

Si vous exécutez la commande dans un environnement où la liaison principale du poste de travail avec son domaine ou gestionnaire supérieur n'est pas active où au moins une liaison secondaire est active, le programme génère la sortie suivante :

```
CPUID   RUN     NODE     LIMIT FENCE   DATE    TIME    STATE     METHOD DOMAIN
MASTER 360 *WNT MASTER  10  0 03/05/2010 1348  I J  E     MASTERDM
FTA1  360 WNT FTA   10  0 03/05/2010 1348 FTI JW M      MASTERDM
FTA2  360 WNT FTA   10  0 03/05/2010 1348 FTI JW M      MASTERDM
FTA3  360 WNT MANAGER 10  0 03/05/2010 1348 FTI JW M      DOMAIN1
FTA4  360 WNT FTA   10  0 03/05/2010 1348 F I J M      DOMAIN1
FTA5  360 WNT FTA   10  0 03/05/2010 1348 L I  M      DOMAIN1
SA1   360 WNT S-AGENT 10  0 03/05/2010 1348 F I J M      DOMAIN1
XA_FTA4 360 OTHR X-AGENT 10  0 03/05/2010 1348 L I J M      DOMAIN1
FTA6  360 WNT MANAGER 10  0 03/05/2010 1348 F I J M      DOMAIN2
FTA7  360 WNT FTA   10  0 03/05/2010 1349 F I J M      DOMAIN2
```

Si vous exécutez la commande dans un environnement où la liaison principale du poste de travail avec son domaine ou gestionnaire supérieur et toutes les liaisons secondaires sont actives, le programme génère la sortie suivante :

```
CPUID   RUN     NODE     LIMIT FENCE   DATE    TIME     STATE    METHOD DOMAIN
MASTER 360 *WNT MASTER  10  0 03/05/2010 1348  I J  E     MASTERDM
FTA1  360 WNT FTA   10  0 03/05/2010 1348 FTI JW M      MASTERDM
FTA2  360 WNT FTA   10  0 03/05/2010 1348 FTI JW M      MASTERDM
FTA3  360 WNT MANAGER 10  0 03/05/2010 1348 FTI JW M      DOMAIN1
FTA4  360 WNT FTA   10  0 03/05/2010 1348 F I J M      DOMAIN1
FTA5  360 WNT FTA   10  0 03/05/2010 1348 F I  M      DOMAIN1
SA1   360 WNT S-AGENT 10  0 03/05/2010 1348 F I J M      DOMAIN1
XA_FTA4 360 OTHR X-AGENT 10  0 03/05/2010 1348 L I J M      DOMAIN1
FTA6  360 WNT MANAGER 10  0 03/05/2010 1348 F I J M      DOMAIN2
FTA7  360 WNT FTA   10  0 03/05/2010 1349 F I J M      DOMAIN2
```

4. Pour obtenir une liste des moniteurs de règles actifs sur le poste de travail CPU1, exécutez cette commande :

   ```
   sc CPU1 getmon
   ```

   Vous obtenez la sortie suivante :

   ```
   Monitoring configuration for CPU1:
   ******************************************
   *** Package Date : 04/22/2009 12:00 GMT ***
   ******************************************
   Rule1::FileMonitor#FileCreated:Workstation=CPU1,CPU2;File="\tmp\filename"
   Rule2::FileMonitor#ModificationCompleted:Workstation=CPU1,CPU3;File="\staging\orders"
   Rule3::TWSObjectsMonitor#JobSubmit:JobKey=CPU1#JS1.Job1
   Rule5::TWSObjectsMonitor#JobLate:JobKey=CPU1#JS1.Job1
   ```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche comme suit :

1. A partir de la barre de navigation, cliquez sur **Statut et état de santé du système** > **Surveillance de l'environnement** > **Surveillance des postes de travail**.
2. Sélectionnez une tâche pour surveiller les postes de travail.
3. Choisissez un nom pour le moteur ou spécifiez les propriété de connexion, puis cliquez sur **OK**.

Format standard

ID poste de travail
Nom du poste de travail auquel ces informations s'appliquent.

EXECUTION
Numéro d'exécution du fichier Symphony.

NOEUD
Type de noeud et type de poste de travail. Les types de noeud sont les suivants :
- UNIX
- WNT
- OTHER
- ZOS
- IBM i

Les types de poste de travail sont les suivants :
- MASTER
- GESTIONNAIRE
- FTA
- S-AGENT
- X-AGENT
- AGENT
- POOL
- D-POOL
- REM-ENG

LIMIT
Nombre maximal de travaux Tivoli Workload Scheduler.

FENCE
Priorité minimale de travail Tivoli Workload Scheduler.

DATE HEURE
Date et heure de début de l'exécution du plan de production courant (fichier Symphony) par Tivoli Workload Scheduler.

ETAT Affiche les informations suivantes :

- L'état des liens et processus du poste de travail. Jusqu'à cinq caractères sont affichés comme suit. L'explication des caractères est divisée en fonction de la portée des caractères :

  ```
  [L|F] [T|H|X|B] [I] [J] [W|H|X] [M] [E|e] [D] [A|R]
  ```

  où :

  L La liaison principale est ouverte ("linked") sur le gestionnaire de domaine/supérieur.

  Si le poste de travail est de type agent ou moteur distant, cet indicateur précise que le poste de travail est connecté au serveur Workload Broker.

  Si le poste de travail est de type pool ou pool dynamique, cet indicateur précise que le poste de travail de courtier de charge de travail auprès duquel le pool ou pool dynamique est enregistré est lié à son gestionnaire de domaine/supérieur.

  F Le poste de travail est complètement connecté via les liaisons principale et secondaires. Cet indicateur ne s'affiche que si l'option globale ***enSwfaultTol*** a pour valeur *YES* à l'aide de la ligne de commande **optman** sur le gestionnaire de domaine maître. Il indique que le poste de travail est directement lié à son gestionnaire de domaine, ainsi qu'à tous ses gestionnaires de domaine de sauvegarde intégrale. Pour plus d'informations sur l'utilisation de la ligne de commande **optman**, voir *IBM Tivoli Workload Scheduler - Guide d'administration*.

T Cet indicateur s'affiche si l'agent tolérant aux pannes est directement connecté au gestionnaire de domaine à partir duquel vous exécutez la commande.

H Le poste de travail est connecté via son hôte.

X Le poste de travail est connecté comme agent étendu (x-agent).

B Le poste de travail communique au moyen du serveur de courtier de charge de travail.

I Si le poste de travail est de type agent MASTER, MANAGER, FTA, S-AGENT, X-AGENT, cet indicateur précise que le programme **jobman** a terminé l'initialisation du démarrage.

Si le poste de travail est de type agent, pool ou pool dynamique, cet indicateur précise que l'agent est correctement initialisé.

Si le poste de travail est de type moteur distant, cet indicateur précise que la communication entre le poste de travail de moteur distant et le moteur distant s'est initialisée correctement.

J Si le poste de travail est de type agent MASTER, MANAGER, FTA, S-AGENT, X-AGENT, cet indicateur précise que le programme **jobman** est en cours d'exécution.

Si le poste de travail est de type agent, cet indicateur précise que JobManager est en cours d'exécution. Etant donné qu'aucune surveillance n'est effectuée sur les postes de travail du pool dynamique, le caractère J apparaît toujours pour ce type de poste de travail.

Si le poste de travail est de type pool, cet indicateur précise que le processus JobManager est en cours d'exécution sur au moins un agent enregistré auprès du pool.

Si le poste de travail est de type moteur distant, cet indicateur précise que la commande ping lancée sur le moteur distant a réussi.

W Le poste de travail est lié via le protocole TCP/IP à l'aide du processus **writer**.

Si le poste de travail exécutant **conman** est directement lié au poste de travail à distance, l'indicateur W apparaît car le processus mailman local est lié au processus writer.

```
LTI JW
```

Si le poste de travail exécutant **conman** est directement lié au poste de travail à distance, l'indicateur W n'apparaît pas car le processus mailman local n'est pas lié au processus writer.

```
L I J
```

Pour plus de détails sur le processus **writer**, voir la rubrique relative aux processus réseau dans le manuel *Tivoli Workload Scheduler - Guide d'administration*.

Remarque : Si le poste de travail exécutant **conman** correspond à l'hôte de l'agent étendu, l'agent étendu est à l'état

```
LXI JX
```

Si le poste de travail exécutant **conman** n'est pas l'hôte de l'agent étendu, l'agent étendu est à l'état

```
LHI JH
```

- L'état de l'agent de surveillance. Jusqu'à trois caractères sont affichés comme suit :

  ```
  [M] [E|e] [D]
  ```

  où :

  M Le processus `monman` s'exécute. Cet indicateur est affiché pour tous les postes de travail du réseau lorsque la fonction d'automatisation de la charge de travail gérée par des événements est activée (l'option globale de `enEventDrivenWorkloadAutomation` est définie sur `yes`), à l'exception des postes de travail sur lesquels `monman` a été arrêté manuellement (au moyen de **conman** ou de Dynamic Workload Console).

  E Le serveur de traitement d'événements est installé et s'exécute sur le poste de travail.

  e Le serveur de traitement d'événements est installé sur le poste de travail mais ne s'exécute pas.

  D Le poste de travail utilise une configuration de surveillance de module mise à jour. Cet indicateur est affiché pour les postes de travail sur lesquels le module le plus récent de règles d'événement a été déployé (soit manuellement au moyen de la commande `planman deploy` ou automatiquement à la fréquence spécifiée par l'option globale `deploymentFrequency`).

- L'état du serveur WebSphere Application Server. Un indicateur à un caractère est affiché si le serveur d'application est installé.

  ```
  [A|R]
  ```

  où :

  A Le serveur WebSphere Application Server a été démarré.

  R Le serveur WebSphere Application Server redémarre.

  L'indicateur est vide si le serveur d'applications est en panne ou s'il n'a pas été installé.

METHODE
Nom de la méthode d'accès indiquée dans la définition du poste de travail. Réservé uniquement aux agents étendus.

DOMAINE
Nom du domaine auquel appartient le poste de travail.

Format info

ID poste de travail
Nom du poste de travail auquel ces informations s'appliquent.

VERSION
Version de l'agent Tivoli Workload Scheduler installée sur le poste de travail.

FUSEAU HORAIRE
Fuseau horaire du poste de travail. Il s'agit de la même valeur que celle de la variable d'environnement TZ. Pour un agent étendu, il s'agit du fuseau horaire de son hôte. Pour un poste de travail de moteur distant, il s'agit du fuseau horaire du moteur distant.

INFO Zone d'informations. Elle contient la version du système d'exploitation et le modèle de matériel de tous les types de poste de travail (à l'exception des

postes de travail de l'agent étendu et du courtier). Pour les agents étendus et les postes de travail de moteur distant, aucune information n'est répertoriée. Pour les postes de travail de moteur distant, cette zone affiche le moteur distant.

Format link

ID poste de travail
Nom du poste de travail auquel ces informations s'appliquent.

HOTE Nom du poste de travail servant d'hôte à un agent standard ou à un agent étendu. Pour les gestionnaires de domaine et les agents tolérants aux pannes, ce nom est identique à l'ID du poste de travail. Pour les postes de travail standard d'un agent et d'un courtier, il s'agit du nom du gestionnaire de domaine. Pour les agents étendus, ce nom correspond à celui du poste de travail hôte.

INDICATEURS
Etat des propriétés du poste de travail. Il peut compter jusqu'à cinq caractères et s'affiche comme suit :

`[A] [B] [F] [s] [T]`

A La liaison automatique est activée dans la définition du poste de travail.
B Cet indicateur est uniquement utilisé dans un environnement complet et précise si l'indicateur **deactivate job launching** est désactivé.
F Le mode Statut intégral est activé dans la définition du poste de travail.
s ID du serveur **mailman** du poste de travail.
T La liaison est définie comme TCP/IP.

ADR Numéro de port TCP/IP du poste de travail.

NOEUD
Nom de noeud du poste de travail.

showdomain

Affiche les informations relatives au domaine.

L'information affichée est mise à jour uniquement tant que Tivoli Workload Scheduler (batchman) s'exécute. Le fait que batchman soit actif ou non est vérifié sur l'écran par le message `Batchman LIVES` ou `Batchman down` lorsque vous émettez la commande `conman start`.

Vous devez avoir un accès ***list*** à l'objet qui est affiché si l'option ***enListSecChk*** a été définie sur **yes** sur le gestionnaire de domaine maître lorsque le plan de production a été créé ou étendu.

Syntaxe

{**showdomain** |
showd} [*domaine*]
[**;info**]
[**;offline**]

Arguments

domaine
Indique le nom du domaine. Par défaut, le domaine dans lequel **conman** est en cours d'exécution est utilisé. Les caractères génériques sont acceptés.

info Affiche les informations au format **info**.

offline
Envoie la sortie de la commande au périphérique de sortie de **conman**. Pour plus d'informations sur ce périphérique, voir «Sortie offline», à la page 374.

Résultats

La sortie de la commande est générée dans deux formats : **standard** et **info**.

Exemples

Pour afficher les informations relatives au domaine `masterdm`, exécutez la commande suivante :

```
showdomain masterdm
```

Voici un exemple de sortie :

```
DOMAIN            MANAGER          PARENT
*MASTERDM    *MASTER
```

Pour afficher les postes de travail appartenant à tous les domaines au format **info**, exécutez la commande suivante :

```
showdomain @;info
```

Voici un exemple de sortie :

```
DOMAIN           MEMBER-CPUs      CPU-Type
 MASTERDM    *MASTER      MASTER
 DOM1        FTA1        MANAGER
 DOM2        FTA2        MANAGER
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche comme suit :

1. A partir de la barre de navigation, cliquez sur **Statut et état de santé du système** > **Surveillance de l'environnement** > **Surveillance des domaines**.
2. Sélectionnez une tâche pour surveiller les domaines.
3. Choisissez un nom pour le moteur ou spécifiez les propriété de connexion, puis cliquez sur **OK**.

Format standard

DOMAINE
Nom du domaine auquel ces informations s'appliquent.

GESTIONNAIRE
Nom du gestionnaire de domaine.

PARENT
Nom du domaine parent.

Format info

DOMAINE
Nom du domaine auquel ces informations s'appliquent.

POSTES DE TRAVAIL MEMBRES
Noms des postes de travail du domaine.

TYPE POSTE DE TRAVAIL
Type de chaque poste de travail : MASTER, MANAGER, FTA, S-AGENT, X-AGENT ou BROKER.

showfiles

Affiche des informations relatives aux dépendances sur les fichiers. Une dépendance de fichier se produit lorsque la présence d'un ou de plusieurs fichiers permet l'exécution d'un travail ou d'un flot de travaux.

L'information affichée est mise à jour uniquement tant que Tivoli Workload Scheduler (batchman) s'exécute. Le fait que batchman soit actif ou non est vérifié sur l'écran par le message `Batchman LIVES` ou `Batchman down` lorsque vous émettez la commande `conman start`.

Syntaxe

{**showfiles** |
sf} [[*num_poste_de_travail*]*fichier*]
[**;***état*[**;**...]]
[**;keys**]
[**;offline**]

{**showfiles** |
sf} [[*poste de travail***#**]*fichier*]
[**;***état*[**;**...]]
[**;deps**[**;keys** | **info** | **logon**]]
[**;offline**]

Arguments

poste de travail
Indique le nom du poste de travail sur lequel se trouve le fichier. Par défaut, le poste de travail sur lequel **conman** est en cours d'exécution est utilisé. Les caractères génériques sont acceptés.

fichier Indique le nom du fichier. Ce nom doit être placé entre guillemets (") s'il contient des caractères autres que les suivants : caractères alphanumériques, tirets (-), barres obliques (/), barres obliques inversées (\) et traits de soulignement (_). Par défaut, toutes les dépendances sur les fichiers s'affichent. Les caractères génériques sont acceptés.

état Indique l'état des dépendances de fichiers à afficher. Par défaut, les dépendances sur les fichiers s'affichent avec tous les états. Les états sont les suivants :

Yes Le fichier existe et est disponible.

Non Le fichier n'est pas disponible ou n'existe pas.

? La disponibilité est en cours de vérification.

<blank>
Le fichier n'a pas encore été vérifié ou était disponible et a été utilisé pour respecter une dépendance de travail ou flot de travaux.

keys Affiche la liste composée d'une seule colonne des objets sélectionnés par la commande.

deps Affiche les informations au format **deps**. Utilisez **keys**, **info** ou **logon** pour modifier l'affichage.

offline
Envoie la sortie de la commande au périphérique de sortie de **conman**. Pour plus d'informations sur ce périphérique, voir «Sortie offline», à la page 374.

Résultats

La sortie de la commande est générée dans trois formats : **standard**, **keys** et **deps**. Les arguments **keys**, **info** et **logon** modifient l'affichage **deps**.

Exemples

Pour afficher l'état d'une dépendance de fichier sur le fichierd:\apps\mis\lib\data4, exécutez la commande suivante :

```
showfiles d:\apps\mis\lib\data4
```

Pour afficher **hors ligne** l'état de toutes les dépendances de fichier sur tous les postes de travail au format deps, exécutez la commande suivante :

```
sf @#@;deps;offline
```

Pour afficher l'état de toutes les dépendances de fichier sur tous les postes de travail au format deps, exécutez la commande suivante :

```
sf @#@;deps
```

Voici un exemple de sortie :

```
                                              (Est) (Est)
Workstation Job Stream SchedTime  Job      State Pr Start Elapse ReturnCode Dependencies

MASTER#/test/^LFILEJOB^ Les dépendances sont les suivantes :
 MASTER     #LFILEJOB  0600 11/26 ******** READY 10
                                LFILEJOB HOLD  10 (11/26)                ^LFILEJOB^

MASTER#/usr/home/me10_99/`/usr/home/me10_99/bin/parms FILE_JS1` Les dépendances sont les suivantes :
 MASTER     #FILE_JS1  0600 11/26 ******** HOLD  10 (11/26)                parms FILE_JS1`
                                FILE_JS1 HOLD  10 (11/26)

MASTER#/usr/home/me10_99/`/usr/home/me10_99/bin/parms FILE_JOB1` Les dépendances sont les suivantes :
 MASTER     #FILE_JOB1 0600 11/26 ******** READY 10
                                FILE_JB1 HOLD  10 (11/26)                parms FILE_JB1`
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle qui suit :

1. A partir de la barre de navigation, cliquez sur **Statut et état de santé du système** > **Surveillance de la charge de travail** > **Surveillance des fichiers**.
2. Sélectionnez une tâche pour surveiller les fichiers.

3. Choisissez un nom pour le moteur ou spécifiez les propriété de connexion, puis cliquez sur **OK**.

Format standard

Existe Etat de la dépendance de fichier

Nom de fichier
Nom du fichier.

Format Keys

Les fichiers sont répertoriés dans une liste, chacun d'eux sur une ligne différente. Les noms de répertoires ne sont pas inclus. Chaque fichier est répertorié au format suivant :

```
workstation#file
```

Format Deps

Les fichiers sont répertoriés, suivis des travaux et flots de travaux dépendants. Les travaux sont répertoriés au format **showjobs** standard. Les flots de travaux sont répertoriés au format **showschedules** standard.

Format Deps;keys

Les travaux et les flots de travaux qui comportent des dépendances `file` sont répertoriés, un par ligne, au format suivant :

```
poste_de_travail#flot_travaux[.job]
```

Format Deps;info

Les fichiers sont répertoriés, suivis des travaux et flots de travaux dépendants. Les travaux sont répertoriés au format **showjobs;info**. Les flots de travaux sont répertoriés au format **showschedules** standard.

Format Deps;logon

Les fichiers sont répertoriés, suivis des travaux et flots de travaux dépendants. Les travaux sont répertoriés au format **showjobs;logon**. Les flots de travaux sont répertoriés au format **showschedules** standard.

showjobs

Affiche les informations relatives aux travaux.

L'information affichée est mise à jour uniquement tant que Tivoli Workload Scheduler (batchman) s'exécute. Le fait que batchman soit actif ou non est vérifié sur l'écran par le message `Batchman LIVES` ou `Batchman down` lorsque vous émettez la commande `conman start`.

Vous devez avoir un accès ***list*** à l'objet qui est affiché si l'option ***enListSecChk*** a été définie sur **yes** sur le gestionnaire de domaine maître lorsque le plan de production a été créé ou étendu.

Syntaxe

{**showjobs** |
sj} [*travail_sélectionné*]
[**;keys** | **info** | **step** | **logon** | **crit** | **keys retcod**]
[**;short** | **single**]
[**;offline**]
[**;showid**]

{**showjobs** |
sj} [*travail_sélectionné*]
[**;deps**[**;keys** | **info** | **logon**]]
[**;short** | **single**]
[**;offline**]
[**;showid**]
[**;props**]

{**showjobs**
| **sj**} [*travail_sélectionné* |
[**poste_de_travail#**]*jobnumber.hhmm*]
[**;stdlist**[**;keys**]]
[**;short** | **single**]
[**;offline**]
[**;showid**]
[**;props**]

Arguments

crit Affiche des informations au format **crit**.

deps Affiche des informations au format deps. En d'autres termes, les travaux utilisés dans les dépendances de *prédécesseur/successeur* sont répertoriés, suivis des travaux et flots de travaux dépendants. Les travaux sont répertoriés au format `showjobs` de base. Les flots de travaux sont répertoriés au format `showschedules` de base. Utilisez "keys", "info" ou "logon" pour modifier l'affichage "deps".

hhmm Heure de début du travail. Utilisez cet argument avec les arguments **stdlist** et **single** pour afficher une instance spécifique du travail.

info Affiche les informations au format **info**.

numéro_travail
Numéro du travail.

travail_sélectionné
Voir «Sélection de travaux dans les commandes», à la page 379.

keys Affiche la liste composée d'une seule colonne des objets sélectionnés par la commande.

logon Affiche les informations au format **logon**.

offline
Envoie la sortie de la commande au périphérique de sortie de **conman**. Pour plus d'informations sur ce périphérique, voir «Sortie offline», à la page 374.

props Affiche les informations suivantes sur l'instance de travail spécifiée, vous devez avoir un accès Display aux propriétés de l'instance de travail spécifiée qui est affichée :

Informations générales

- Job
- Poste de travail
- Tâche
- Type de tâche
- Flot de travaux
- Poste de travail du flot de travaux

- Heure planifiée
- Priority
- Connexion
- Monitored
- Confirmation requise
- Interactif
- Critique

Informations d'exécution

- Status
- Statut interne
- Dépendances non satisfaites
- Numéro du travail
- Options de réexécution
- Information
- Promu
- Code retour
- Expression du mappage de code retour

Informations sur l'heure

- Démarrage réel
- Premier démarrage
- Dernier démarrage
- Dernière action de démarrage
- Durée maximale
- Action durée maximale
- Durée minimale
- Action durée minimale
- Dernier démarrage critique
- Deadline
- Intervalle de répétition
- Durée réelle
- Durée estimée

Informations de reprise

- Action
- Message
- Définition de travail
- Poste de travail

Informations supplémentaires

Cette section présente des propriétés supplémentaires spécifiques pour les travaux reflet et les travaux définis par le JSDL. Pour les travaux reflet, elle contient les informations suivantes :

Pour les travaux reflet distribués :

- Heure planifiée du travail distant
- Travail distant
- Flot de travaux distant

- Poste de travail du flot de travaux distant

Pour les travaux reflet z/OS :

- Heure planifiée du travail distant
- Travail distant
- Poste de travail du travail distant
- Code d'erreur du travail distant

Pour plus d'informations, voir «Modification de l'état du travail reflet après que la liaison est établie», à la page 693.

Remarque : Les informations sur les travaux archivés ne peuvent pas être extraites à l'aide de l'option **props**.

retcod Affiche le code retour du travail. Cet argument doit être utilisé avec l'argument **keys**, par exemple :

```
%sj @; keys retcod
```

short Réduit l'affichage des travaux **réexécutés** et **répétitifs** afin d'inclure uniquement ce qui suit :

- Première répétition
- Travaux avec des états différents
- Travaux concordant exactement

Remarque : Cette zone affiche les propriétés spécifiques si le travail est un travail reflet ou un travail défini par JSDL.

showid
Affiche l'identificateur de chaque flot de travaux.

single Sélectionne uniquement le travail parent d'une chaîne pouvant inclure des réexécutions, des répétitions et des travaux de reprise. Le travail doit être identifié par son numéro dans *travail_sélectionné*. Cette fonction est utile avec l'option **stdlist**.

stdlist Affiche les informations au format **stdlist**. Utilisez l'argument **keys** pour modifier l'affichage.

Remarque : Les informations sur les travaux archivés ne peuvent pas être extraites à l'aide de l'option **stdlist**.

step Affiche les informations au format **step**.

poste de travail
Nom du poste de travail sur lequel le travail est exécuté. Les caractères génériques sont acceptés.

Résultats

La sortie de la commande **showjobs** est générée dans huit formats : **standard**, **keys**, **info**, **step**, **logon**, **deps**, **crit** et **stdlist**. Les arguments **keys**, **info**, **crit** et **logon** modifient l'affichage.

Exemples

Pour afficher le statut de tous les travaux du flot de travaux `acctg` sur le poste de travail `site3`, vous pouvez exécuter la commande `showjobs` selon l'un de ces deux formats :

```
showjobs site3#acctg.@
```

ou :

```
showjobs site3#acctg
```

Pour afficher le statut du travail JBA appartenant au flot de travaux TEST1 sur le poste de travail CPUA, sur lequel vous exécutez **conman**, et demander l'affichage de l'identificateur du flot de travaux, exécutez la commande suivante :

```
sj CPUA#TEST1(0900 02/19/06).JBA
```

Voici un exemple de sortie de cette commande :

```
Workstation Job Stream SchedTime Job State Pr Start Elapse ReturnCode Dependencies

CPUA    #TEST1  0900 02/19 *** HOLD  0(02/19)         {02/20/06}; -TEST-
                               JBA HOLD 66(14:30)         J2(0600 02/24/06).JB1
```

La dépendance **at** est définie sur (14:30) dans la colonne Démarrage et la dépendance de prédécesseur/successeur entre le travail J2(0600 02/24/06).JB1 et le travail JOBA est indiquée dans la colonne Dépendances.

Dans la colonne Dépendances la date placée entre accolades, {02/20/06}, indique que l'instance du flot de travaux a été reportée et la date indique le jour où elle a été ajoutée au plan de production pour la première fois.

Pour afficher le statut des travaux appartenant au flot de travaux JSDOC sur le poste de travail site3, sur lequel vous exécutez **conman**, et demander l'affichage de l'identificateur du flot de travaux, exécutez la commande suivante :

```
%sj JSDOC.@;showid
```

Voici un exemple de sortie de cette commande :

```
Workstation Job Stream SchedTime Job State Pr Start Elapse ReturnCode Dependencies

site3    #JSDOCOM 0600 11/26 *** SUCC 10 11/26 00:01      {0AAAAAAAAAAAACRZ}
                             JDOC SUCC 10 11/26 00:01    0  #J25565
```

L'identificateur 0AAAAAAAAAAAACRZ du flot de travaux JDOCOM est indiqué dans la colonne Dépendances.

Remarque : La date ou l'heure affichée dans la colonne **Démarrage** est convertie en fonction du fuseau horaire défini sur le poste de travail sur lequel le flot de travaux doit s'exécuter.

Pour afficher le statut des travaux appartenant au flot de travaux JSDOCOM sur le poste de travail site3, et demander l'affichage des informations sur l'ID utilisateur sous lequel le travail s'exécute, exécutez la commande suivante :

```
sj site3#JSDOCOM.@;logon
```

Voici un exemple de sortie de cette commande :

```
Workstation Job Stream SchedTime  Job    State Job#    Logon   ReturnCode
site3    #JSDOCOM 0600 11/26
                                       JDOCOM SUCC #J25565 me10_99   0
```

Pour afficher le statut de tous les travaux à l'état HOLD sur tous les postes de travail au format **deps**, exécutez la commande suivante :

```
sj @#@.@+state=hold;deps
```

Voici un exemple de sortie :

```
Workstation Job Stream SchedTime Job State Pr Start Elapse RetCode Dependencies

CPUA#JS2.JOBB Les dépendances sont les suivantes :

CPUA        #JS21     0900 02/19 ***** HOLD  0(02/19)              {02/20/06}; -TEST- JOBA HOLD 66(14:30)
                                                                                 JS22(0600 02/24/06).JOBB

CPUA#JS25.JOBC Les dépendances sont les suivantes :

CPUA       #JS25      0600 02/24 ***** HOLD 10(02/24)              {02/20/06}
                                 jobaa HOLD 10(02/24)(00:01)       TEST1; JOBC          TEST2; JOB1
                                                                                 JS18(0600 02/24/06).@

CPUA#JS25.JOB1 Les dépendances sont les suivantes :

CPUA       #JS25      0600 02/24 ***** HOLD 10(02/24)               {02/20/06}
                                 JOBC  HOLD 10(02/24)(00:01)       JOB1
                                 jobaa HOLD 10(02/24)(00:01)       TEST1; JOBC         TEST2; JOB1
JS18(0600 02/24/06).@
```

Pour afficher le journal à partir des fichiers de liste standard du travail J25 du flot de travaux JS25(0600 02/19/06) sur le poste de travail CPUA tournant dans un environnement UNIX, exécutez la commande suivante :

```
sj CPUA#JS25(0600 02/19/06).J25;stdlist
```

Le programme génère la sortie suivante :

```
================================================================
= TRAVAIL    : CPUA#JS25[(0600 02/19/06),(0AAAAAAAAAAAABQM)].J25
= UTILISATEUR  : tme10_99
= FICHIER JCL  : ls
= Numéro du travail : 28630
= Lun 02/20/06 07:57:37 PST
================================================================
Tivoli Workload Scheduler (UNIX)/JOBMANRC
AWSBIS307I Starting /usr/home/tme10_99/jobmanrc ls

Tivoli Workload Scheduler (UNIX)/JOBINFO 8.4 (9.5)
Installation réalisée pour l'utilisateur "tme10_99".
Variable locale LANG définie sur : "fr"
...
... <commandes stdout, stderr et echo>
...
AWSBIS308I Fin du travail
================================================================
= Statut de sortie : 0
= Heure système (Secondes) : 0 Durée écoulée (Minutes) : 0
= Temps utilisateur (Secondes) : 0
= Lun 02/20/06 07:57:38 PST
================================================================
```

où :

Statut de sortie
Statut du travail à l'issue de son exécution

Temps écoulé
Durée écoulée du travail

Heure système
Temps passé par le noyau système pour exécuter le travail

Temps utilisateur
Temps passé par l'utilisateur système pour exécuter le travail

Remarque : Les zones **Heure système** et **Temps utilisateur** sont utilisées sur UNIX uniquement. Leur valeur sous Windows est toujours **0**, ceci parce que, sous Windows, le processus **joblnch.exe** s'exécute en un temps très court qui peut être considéré comme nul.

Pour afficher les propriétés du travail reflet de numéro de travail 546863237, exécutez la commande suivante :

```
sj 546863237;props
```

Voici un exemple de sortie de cette commande :

```
Informations générales
  Job = D_SHADOW_JOB
  Workstation = REMENG1
  Task =
    <jsdl:jobDefinition
          xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
          xmlns:dshadow="http://www.ibm.com/xmlns/prod/scheduling/1.0/dshadow"
          xsi:schemaLocation="http://www.ibm.com/xmlns/prod/scheduling/1.0/dshadow">
       <jsdl:application name="distributedShadowJob">
          <dshadow:DistributedShadowJob>
             <dshadow:JobStream>JS1</dshadow:JobStream>
             <dshadow:Workstation>MYWKST</dshadow:Workstation>
             <dshadow:Job>JOBDEF1</dshadow:Job>
             <dshadow:matching>
                <dshadow:previous/>
             </dshadow:matching>
          </dshadow:DistributedShadowJob>
       </jsdl:application>
    </jsdl:jobDefinition>
  Task Type = distributedShadowJob
  Job Stream = JSDIST
  Job Stream Workstation = MYWKST
  Scheduled Time = 2010/08/11 06:00 TZ CEST
  Priority = 10
  Login =
  Monitored = No
  Requires Confirmation = No
  Interactive = No
  Critical = No
Runtime Information
  Status = Undecided
  Internal Status = DONE
  Not Satisfied Dependencies = 0
  Job Number = 546863237
  Rerun Options =
  Information =
  Promoted = No
  Return Code =
  Return Code Mapping Expression =
Time Information
  Actual Start = 2010/08/11 12:00 TZ CEST
  Earliest Start =
  Latest Start =
  Latest Start Action =
  Maximum Duration =
  Maximum Duration Action =
  Minimum Duration = 000:01 (hhh:mm)
  Minimum Duration Action = Continue
  Critical Latest Start =
  Deadline =
  Repeat Range =
```

```
  Actual Duration =
  Estimated Duration = 00:02 (hh:mm)

Recovery Information
  Action =
  Message =
  Job Definition =
  Workstation =
Extra Information
  Remote Job Scheduled Time = 08/11/2010 06:00 TZ CEST
  Remote Job = JOBDEF1
  Remote Job Stream = JS1
  Remote Job Stream Workstation = MYWKST
```

Dans l'exemple suivant, le programme affiche le statut du travail dbseload en générant le code retour **7** et l'état TERMINE :

```
$ conman sj workstation#DAILY_DB_LOAD
Tivoli Workload Scheduler (UNIX)/CONMAN 8.4 (1.36.2.22) Eléments sous licence -
Propriété d'IBM(R)
5698-WSH
(C) Copyright IBM Corp 1998, 2007 All rights reserved.
US Government User Restricted Rights
Use, duplication or disclosure restricted by
GSA ADP Schedule Contract with IBM Corp.
IBM est une marque d'International Business Machines
Corporation aux Etats-Unis et/ou dans certains autres pays.
Installation réalisée pour l'utilisateur "tme10_99".
Variable locale LANG définie sur : "fr"
Date de planification : (Exp) 02/20/06 (#35) sur CPUA.
Batchman ACTIF. Limit:50,Fence:0,Audit Level:0
sj workstation#DAILY_DB_LOAD
(Est) (Est)
UC Calendrier Travail Etat Pr Démarrage
Temps écoulé Dépendances Code retour
WORKSTATION #DAILY_DB_LOAD ****************************** SUCC 10 22:11
00:04
DATASPLT SUCC 10 22:11
00:01 #J17922 0
DATAMRGE ABEND 10 22:12
00:01 #J17924 1
CHCKMRGE SUCC 10 22:12
00:01 #J17926 0
DATACLNS SUCC 10 22:12
00:01 #J17932 0
DATARMRG SUCC 10 22:13
00:01 #J18704 0
DBSELOAD SUCC 10 22:13
00:01 #J18706 7
DATAREPT SUCC 10 22:13
00:01 #J18712 0
DATARTRN SUCC 10 22:14
00:01 #J18714 0
$
```

L'exemple suivant affiche le code retour d'un travail spécifique intitulé workstation#daily_db_load.dbseload :

```
$ conman sj workstation#daily_db_load.dbseload\;keys\;retcod

Tivoli Workload Scheduler (UNIX)/CONMAN 8.4 (1.36.2.22) Eléments sous licence -
Propriété d'IBM(R)
5698-WSH
(C) Copyright IBM Corp 1998, 2007 All rights reserved.
US Government User Restricted Rights
Use, duplication or disclosure restricted by
GSA ADP Schedule Contract with IBM Corp.
```

```
IBM est une marque d'International Business Machines
Corporation aux Etats-Unis et/ou dans certains autres pays.
Installation réalisée pour l'utilisateur "tme10_99".
Variable locale LANG définie sur : "fr"
Date de planification : (Exp) 02/20/06 (#35) sur CPUA.
Batchman ACTIF. Limit:50,Fence:0,Audit Level:0
sj workstation#daily_db_load.dbseload;keys;retcod 8
$
```

Une fois intégrée à un script, la fonction *retcod* peut se révéler assez puissante.

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle qui suit :

1. Dans la barre de navigation, cliquez sur **Statut et état de santé du système** > **Surveillance de la charge de travail** > **Surveillance des travaux**.
2. Sélectionnez **Tous les travaux du plan** ou une autre tâche de surveillance des flots de travaux.
3. Choisissez un nom pour le moteur ou spécifiez les propriété de connexion, puis cliquez sur **OK**.

Format standard

CPU Poste de travail sur lequel le travail s'exécute.

Schedule
Nom du flot de travaux.

SchedTime
Date et heure prévues de l'exécution du travail dans le plan.

Job Nom du travail. Il se peut que la mention suivante précède un nom de travail :

>> rerun as
Travail réexécuté à l'aide de la commande **rerun** ou par suite d'une reprise automatique.

>> rerun step
Travail réexécuté à l'aide de la commande **rerun ;step**.

>> every run
Deuxième exécution et exécutions ultérieures d'un travail répétitif.

>>recovery
Exécution d'un travail de reprise.

State Etat du travail ou du flot de travaux. Les états des travaux sont les suivants :

ABEND
Le travail s'est terminé avec un code de sortie différent de zéro.

ABENP
Une confirmation **abend** a été reçue mais le travail n'est pas terminé.

ADD Le travail est en cours de soumission.

CANCL
Ne s'applique qu'aux dépendances interréseau. Le travail distant ou le flot de travaux a été annulé.

DONE
Le travail est terminé avec un état inconnu.

ERROR
Pour les dépendances interréseaux uniquement, une erreur s'est produite lors de la recherche du statut distant.

EXEC Le travail est en cours.

EXTRN
Pour les dépendances interréseaux uniquement, l'état est inconnu. Une erreur s'est produite, une action de réexécution vient d'être effectuée sur le travail du flot de travaux `EXTERNE`, ou le travail ou le flot de travaux distant n'existe pas.

FAIL Impossible de lancer le travail.

FENCE
La valeur de priorité du travail est inférieure à la priorité minimale.

HOLD
Le travail attend la résolution des dépendances.

INTRO
Le travail est introduit à des fins de lancement par le système.

PEND Le travail est terminé et attend une confirmation.

READY
Le travail est prêt à démarrer et toutes les dépendances sont résolues.

SCHED
L'heure **at** définie pour le travail n'a pas été atteinte.

SUCC Le travail s'est terminé avec un code de sortie nul.

SUCCP
Une confirmation SUCC a été reçue mais le travail n'est pas terminé.

WAIT Le travail est à l'état WAIT (agent étendu).

Les états de flot de travaux sont les suivants :

ABEND
Le flot de travaux s'est terminé avec un code de sortie différent de zéro.

ADD Le flot de travaux a été ajouté par intervention de l'opérateur.

CANCELP
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.

ERROR
Pour les dépendances interréseaux uniquement, une erreur s'est produite lors de la recherche du statut distant.

EXEC Le flot de travaux est en cours d'exécution.

EXTRN
Ne s'applique qu'aux dépendances interréseau. Il s'agit de l'état du

flot de travaux EXTERNAL contenant des travaux qui font référence à des travaux ou des flots de travaux dans le réseau distant.

HOLD
Le flot de travaux attend la résolution des dépendances.

READY
Le flot de travaux est prêt à démarrer et toutes les dépendances sont résolues.

STUCK
L'exécution du flot de travaux a été interrompue. Aucun travail n'est lancé sans l'intervention de l'opérateur.

SUCC Le flot de travaux s'est achevé correctement.

Pr Priorité du travail ou du flot de travaux. Un signe plus (+) précédant la priorité indique que le travail a été lancé.

(Est)Start
Heure de début du travail ou du flot de travaux. Les parenthèses indiquent une estimation de l'heure de début. Si la commande est effectuée le jour où le travail est planifié pour s'exécuter, le paramètre **Start** affiche l'heure en tant que (Est)Start. Si la commande est effectuée un autre jour que le jour où le travail est planifié pour s'exécuter, le paramètre **Start** affiche la date en tant que (Est)Start. Par exemple, si vous avez le travail suivant dont l'heure de début a lieu le jour où le travail est planifié pour s'exécuter :

```
SCHEDULE MASTERB1#JS_B
ON RUNCYCLE RULE1 "FREQ=DAILY;"
AT 1700
:
MASTERB1#JOB1
 AT 1800
END
```

Vous obtenez la sortie suivante :

```
%sj @#@
                                              (Est) (Est)
CPU      Schedule SchedTime  Job   State Pr Start  Elapse RetCode Deps
MASTERB1#JS_B   1700 08/18 ***** HOLD 10(17:00)
                             JOB1 HOLD 10(18:00)
```

Par exemple, si vous avez le travail suivant dont l'heure de début a lieu un autre jour que celui auquel le travail est planifié pour s'exécuter :

```
SCHEDULE MASTERB1#JS_A
ON RUNCYCLE RULE1 "FREQ=DAILY;"
AT 0400
:
MASTERB1#JOB_A
 AT 0500
END
```

Vous obtenez la sortie suivante :

```
%sj @#@
                                              (Est) (Est)
CPU      Schedule SchedTime  Job   State Pr Start  Elapse RetCode Deps
MASTERB1#JS_A   0400 08/19 ***** HOLD 10(08/19)
                             JOB_A HOLD 10(08/19)
```

(Est)Elapse
Heure d'exécution du travail ou du flot de travaux. Les parenthèses indiquent une estimation basée sur les statistiques consignées.

dépendances
Liste des commentaires et des dépendances sur les travaux. Une combinaison des éléments suivants peut être répertoriée :

- Pour une dépendance de `prédécesseur/successeur`, un nom de travail ou de flot de travaux s'affiche.

  Si le travail ou le flot de travaux est un prédécesseur suspendu, son nom est suivi d'un `[P]`.

  En cas de dépendance orpheline, un `[O]` est affiché.

  Pour plus d'informations sur les prédécesseurs ou les dépendances orphelines en attente, voir «Gestion des dépendances externes de prédécesseur/successeur des travaux et des flots de travaux», à la page 65.
- Pour une dépendance `opens`, le nom de fichier s'affiche. Si le fichier réside sur un agent étendu et que son nom comporte plus de 25 caractères, seuls les 25 derniers caractères s'affichent.
- Pour une dépendance `needs`, un nom de ressource placé entre tirets (-) s'affiche. Si le nombre d'unités demandées est supérieur à un, le nombre s'affiche avant le premier tiret.
- Pour une heure d'échéance **deadline**, l'heure précédée du signe inférieur (<) s'affiche.
- Pour une fréquence **every**, la fréquence de répétition précédée par une perluète (&) s'affiche.
- Pour une échéance **until**, l'heure précédée du signe inférieur (<) s'affiche.
- Pour une durée **maximum duration** dépassée, **[MaxDurationExceeded]** s'affiche en plus du paramètre maxdur=*hhh:mm*.
- Pour une durée **maximum duration** dépassée, et pour laquelle **onmaxdur** *action* est défini sur `Kill`, **[KillSubmitted]** s'affiche.
- Pour une durée **maximum duration** dépassée, et pour laquelle **onmaxdur** *action* est défini sur `Continue`, **[Continue]** s'affiche.
- Pour une durée **minimum duration** qui n'est pas atteinte et pour laquelle un travail se termine avec succès, [MinDurationNotReached] s'affiche en plus du paramètre mindur=*hhh:mm*.
- Pour une durée **minimum duration** qui n'est pas atteinte, et pour laquelle **onmindur** *action* est défini sur `Continue`, **[Continue]** s'affiche.
- Pour une durée **minimum duration** qui n'est pas atteinte, et pour laquelle **onmindur** *action* est défini sur `Abend`, **[Abended]** s'affiche.
- Pour une durée **minimum duration** qui n'est pas atteinte, et pour laquelle **onmindur** *action* est défini sur `Confirm`, **[ConfirmSubmitted]** s'affiche.
- Pour une dépendance `prompt`, le numéro d'invite s'affiche au format #*num*. Pour les invites globales, le nom d'invite vient à la suite, entre parenthèses.
- Pour les travaux en cours d'exécution, le numéro d'identification de processus (PID) s'affiche au format **#J***nnnnn*.
- Les travaux soumis sur UNIX via les commandes Tivoli Workload Scheduler **at** et **batch** portent la mention **[Userjcl]**.

- Lors du signalement des dépendances horaires, la commande **showjobs** indique dans la colonne **Start** :
  - Uniquement l'heure *hh:mm* si le jour où les dépendances horaires sont définies correspond au jour où la commande **showjobs** est exécutée.
  - Uniquement la date *MM/JJ* si le jour où les dépendances horaires sont définies ne correspond pas au jour où la commande **showjobs** est exécutée.
- Les travaux annulés portent la mention **[Cancelled]**.
- Les travaux annulés à l'aide de l'option **;pend** portent la mention **[Cancel Pend]** (Annulation en cours).
- Les travaux pour lesquels une heure d'**expiration** est définie, y compris les travaux annulés avec l'option **;pend** , portent la mention **[Until]**.
- **[Recovery]** signifie que l'intervention de l'opération est requise.
- **[Confirmed]** signifie qu'une confirmation est requise car le travail a été planifié à l'aide du mot clé **confirm**.
- **[Script]** concerne les réseaux complets uniquement, c'est-à-dire que le travail comporte un script centralisé et que IBM Tivoli Workload Scheduler for z/OS ne l'a pas encore chargé sur l'agent.

Format Keys

Les noms des travaux sont répertoriés, un par ligne, au format suivant :

```
poste_de_travail#flot_travaux hhmm mm/jj.travail
```

Par exemple :

```
CPU  Schedule SchedTime Job   State Pr Start Elapse RetCode Deps

MYCPU+#SCHED_F+ 0600 03/04 ******* HOLD 55(03/04)        [03/04/06]; #33
                (M235062+#)JOBMDM HOLD 30(03/04)        #1(PRMT3);-16 JOBSLOTS-

MYCPU+#SCHED_F+ 1010 03/04 ******* HOLD 55(03/04)        [03/04/06]; #34
                (M235062+#)JOBMDM HOLD 30(03/04)        #1(PRMT3);-16 JOBSLOTS-
```

Format info

CPU Poste de travail sur lequel le travail s'exécute.

Schedule
Nom du flot de travaux.

SchedTime
Date et heure prévues de l'exécution du travail dans le plan.

Job Nom du travail. Il se peut que la mention suivante précède un nom de travail :

>> rerun as
Travail réexécuté à l'aide de la commande **rerun** ou par suite d'une reprise automatique.

>> rerun step
Travail réexécuté à l'aide de la commande **rerun ;step**.

>> every run
Deuxième exécution et exécutions ultérieures d'un travail répétitif.

>>recovery
Exécution d'un travail de reprise.

Job File
Nom du script ou du fichier exécutable du travail. Les noms de fichiers longs risquent de faire l'objet d'un renvoi à la ligne, ce qui donne lieu à un saut de page incorrect. Pour éviter cela, ajoutez le signe | avant **more**.

Opt Option de reprise de travaux, s'il y a lieu. Les options de reprise sont **RE** pour réexécution, **CO** pour poursuite et **ST** pour arrêt.

Job Nom du travail de reprise, s'il y a lieu.

Prompt
Numéro de l'invite de reprise, s'il y a lieu.

Par exemple :

```
conman "sj;info | more
```

génère un exemple de sortie similaire à ce qui suit :

```
 --------Restart---------
CPU   Schedule SchedTime Job   JobFile        Opt Job Prompt
M235062+#SCHED_22 1010 03/06
                    JOBMDM  /usr/acct/scripts/gl1
                  (B236153+#)JOB_FTA echo job12
M235062+#SCHED_22 0600 03/07
                    JOBMDM  /usr/acct/scripts/gl1
                  (B236153+#)JOB_FTA echo job12
M235062+#FINAL    2359 02/13
                    STARAPPSERVER  /opt/IBM/TWA/TWS/../wastools/startWas.sh
                                                                 Colorado
                    MAKEPLAN  /opt/IBM/TWA/TWS/MakePlan TWSRCMAP:(RC=0) OR (RC=4)
                    SWITCHPLAN  /opt/IBM/TWA/TWS/SwitchPlan
M235062+#FINALPOSTREPORTS    2359 02/13
                    CHECKSYNC /opt/IBM/TWA/TWS/CheckSync
                    CREATEPOSTREPORTS  /opt/IBM/TWA/TWS/CreatePostReports
                                                                 Colorado
                    UPDATESTATS  /opt/IBM/TWA/TWS/UpdateStats
                                                                 Colorado
M235062+#SCHED12 1010 03/06
                    JOBMDM  /usr/acct/scripts/gl1
                  (B236153+#)JOB_FTA echo job12
```

Format Step

Le format n'est pas pris en charge sous Windows.

UC Poste de travail sur lequel le travail s'exécute.

Schedule
Nom du flot de travaux.

SchedTime
Date et heure prévues de l'exécution du travail dans le plan.

Job Nom du travail. Il se peut que la mention suivante précède un nom de travail :

>> rerun as
Travail réexécuté à l'aide de la commande **rerun** ou par suite d'une reprise automatique.

>> repeated as
Deuxième exécution et exécutions ultérieures d'un travail **répétitif**.

State Etat du travail ou du flot de travaux. Voir "Format standard" pour obtenir des informations relatives à l'état.

Return code
Le code retour du travail.

Job# Numéro d'identification de processus affiché au format **#J***nnnnn*.

Step Liste des processus descendants qui sont associés au travail. Pour les travaux d'agent étendu, seuls les processus hôte sont répertoriés.

Format Logon

UC Poste de travail sur lequel le travail s'exécute.

Calendrier
Nom du flot de travaux.

SchedTime
Date et heure prévues de l'exécution du travail dans le plan.

Job Nom du travail. Il se peut que la mention suivante précède un nom de travail :

>> rerun as
Travail réexécuté à l'aide de la commande **rerun** ou par suite d'une reprise automatique.

>> repeated as
Deuxième exécution et exécutions ultérieures d'un travail **répétitif**.

Etat Etat du travail ou du flot de travaux. Voir "Format standard" pour obtenir des informations relatives à l'état.

Return code
Le code retour du travail.

Job# Numéro d'identification de processus affiché au format **#J***nnnnn*.

Connexion
Nom d'utilisateur sous lequel le travail s'exécute.

Sur les systèmes d'exploitation Windows, les formats suivants sont disponibles :

Nom d'utilisateur
Où *nom_utilisateur* représente le nom de l'utilisateur Windows.

domaine\nom_utilisateur
Où *domaine* correspond au domaine Windows de l'utilisateur et *nom_utilisateur* correspond au nom de l'utilisateur Windows.

nom_utilisateur@domaine_internet
Où *nom_utilisateur@domaine_internet* représente le nom d'un utilisateur système dans un format d'adresse électronique. Le nom d'utilisateur est suivi du symbole @, lui-même suivi du nom du domaine Internet auquel l'utilisateur est associé.

Remarque : Insérez le caractère d'échappement '\' avant le caractère '@' dans la valeur `nom_utilisateur@domaine_internet` de la zone de connexion. Par exemple, si vous utilisez l'utilisateur `administrator@bvt.com` dans la zone de connexion, utilisez la syntaxe suivante :

```
..... ; logon=administrator\@bvt.com
```

Format Stdlist

Un fichier de liste standard est créé automatiquement par le processus **jobmon** sous Windows ou par le processus **jobman** sous UNIX chaque fois qu'un travail est lancé par **jobmon** ou **jobman**. Vous pouvez afficher le contenu des fichiers liste standard en utilisant le programme **conman**. Les fichiers de liste standard comprennent les informations suivantes :

- Bannières de début et fin de page
- Commandes echo
- Sortie stdout du travail
- Sortie stderr du travail

Pour préciser un format de date particulier à utiliser dans les fichiers liste standard, modifiez le format de date IBM Tivoli Workload Scheduler avant de créer les fichiers. Il vous suffit de modifier le format de date local.

Selon l'environnement, modifiez le format de date local comme suit :

- Sous UNIX, définissez la variable LANG dans l'environnement au démarrage du processus **netman**. Si la variable LANG n'est pas définie, l'environnement local du système d'exploitation utilise "C" par défaut.
- Sous Windows, procédez comme suit :
  1. Cliquez sur Panneau de configuration → Options régionales et linguistiques, puis modifiez l'environnement local (emplacement).
  2. Cliquez avec le bouton droit de la souris sur **Poste de travail**, sélectionnez Propriétés, cliquez sur **Avancé**, et sur Variables d'environnement, puis définissez la variable LANG dans les variables système.
  3. Arrêtez et redémarrez le système.

Les fichiers de liste standard des travaux sélectionnés s'affichent.

Format Stdlist;keys

Les noms des fichiers de liste standard des travaux sélectionnés sont répertoriés, un par ligne.

Format crit

CPU Poste de travail sur lequel le travail s'exécute.

Schedule
Nom du flot de travaux.

SchedTime
Date et heure prévues de l'exécution du travail dans le plan.

Job Nom du travail. Il se peut que la mention suivante précède un nom de travail :

>> rerun as
Travail réexécuté à l'aide de la commande **rerun** ou par suite d'une reprise automatique.

>> repeated as
Deuxième exécution et exécutions ultérieures d'un travail **répétitif**.

State Etat du travail ou du flot de travaux. Voir "Format standard" pour obtenir des informations relatives à l'état.

Pr Priorité du travail ou du flot de travaux. Un signe plus (+) précédant la priorité indique que le travail a été lancé.

(Est)Start
Heure de début du travail ou du flot de travaux. Les parenthèses indiquent une estimation de l'heure de début. Si l'heure de début dépasse 24 heures dans le passé ou dans l'avenir, la date est répertoriée à la place de l'heure.

(Est)Elapse
Heure d'exécution du travail ou du flot de travaux. Les parenthèses indiquent une estimation basée sur les statistiques consignées.

CP Indique si le travail est marqué comme critique (**C**) et/ou promu **P**).

CritStart
Heure la plus tardive à laquelle peut démarrer un travail sans gêner les échéances des successeurs essentiels à la mission.

Par exemple, le résultat de la commande générique suivante :

```
%sj @#@;crit
```

est :

```
                                              (Est) (Est)    Crit
CPU   Schedule SchedTime Job   State Pr Start Elapse CP  Start

MYCPU_F+#JSA   1600 03/05 ******** HOLD 10
                          JOBA1  HOLD 10        CP  1759 03/05
                          JOBA2  HOLD 10           1758 03/05
                          JOBA3  HOLD 10           1757 03/05
                          JOBA4  HOLD 10        C  1659 03/05
```

Il est à noter que :

- L'indicateur **C** s'applique uniquement aux travaux définis comme critiques dans leur définition de flot de travaux. Il est défini au niveau ou à l'heure **submit**.
- L'indicateur **P** s'applique aux travaux critiques et à leurs prédécesseurs (qui sont des travaux n'étant pas définis comme critiques, mais risquant néanmoins d'avoir un impact sur la réalisation opportune du travail critique d'un successeur). Il est défini au moment de l'exécution si le travail a été promu.
- Les travaux et prédécesseurs critiques font l'objet d'une heure de début critique.

  Le planificateur calcule l'heure de début critique d'un travail critique en soustrayant sa durée estimée de l'échéance. Il calcule l'heure de début critique d'un prédécesseur critique en soustrayant sa durée estimée de l'heure de début critique de son successeur direct. Au sein d'un réseau critique, le planificateur calcule l'heure de début critique du premier travail en premier lieu, puis fonctionne à rebours dans la chaîne de ses prédécesseurs. Ces calculs sont réitérés autant de fois que nécessaire, tant que le travail critique n'a pas été exécuté.

Format Deps

Les travaux utilisés dans les dépendances de prédécesseur/successeur sont répertoriés, suivis des travaux et flots de travaux dépendants. Les travaux sont répertoriés au format **showjobs** standard. Les flots de travaux sont répertoriés au format **showschedules** standard.

Format Deps;keys

Les travaux et les flots de travaux qui comportent des dépendances de prédécesseur/successeur sont répertoriés, un par ligne.

Format Deps;info

Les travaux utilisés dans les dépendances de prédécesseur/successeur sont répertoriés, suivis des travaux et des flots de travaux dépendants. Les travaux sont répertoriés au format **showjobs;info**. Les flots de travaux sont répertoriés au format **showschedules** standard.

Format Deps;logon

Les travaux utilisés dans les dépendances de prédécesseur/successeur sont répertoriés, suivis des travaux et des flots de travaux dépendants. Les travaux sont répertoriés au format **showjobs;logon**. Les flots de travaux sont répertoriés au format **showschedules** standard.

showprompts

Affiche les informations relatives aux invites.

L'information affichée est mise à jour uniquement tant que Tivoli Workload Scheduler (batchman) s'exécute. Le fait que batchman soit actif ou non est vérifié sur l'écran par le message Batchman LIVES ou Batchman down lorsque vous émettez la commande conman start.

Vous devez avoir un accès ***list*** à l'objet qui est affiché si l'option ***enListSecChk*** a été définie sur **yes** sur le gestionnaire de domaine maître lorsque le plan de production a été créé ou étendu.

Syntaxe

{**showprompts** | **sp**} [*invite_sélectionnée*]
[**;keys**]
[**;offline**]

{**showprompts** | **sp**} [*invite_sélectionnée*]
[**;deps**[**;keys** | **info** | **logon**]][**;offline**]

Arguments

invite_sélectionnée
[*nom_invite* | [*poste_de_travail*#]*num_message*][**;***état*[**;**...]]

nom_invite
Indique le nom d'une invite globale. Les caractères génériques sont acceptés.

poste de travail
Indique le nom du poste de travail sur lequel une invite non nommée est émise. Par défaut, le poste de travail sur lequel **conman** est en cours d'exécution est utilisé.

msgnum
Indique le numéro de message d'une invite non nommée.

état Indique l'état des invites à afficher. Les états sont les suivants :

OUI L'invite a généré l'envoi de la réponse **y**.

NON L'invite a généré l'envoi de la réponse **n**.

DEMANDEE
L'invite a été émise mais aucune réponse n'a été générée.

INACTIVE
L'invite n'a pas été émise.

keys Affiche la liste composée d'une seule colonne des objets sélectionnés par la commande.

deps Affiche les informations au format **deps**. Utilisez **keys**, **info** ou **logon** pour modifier l'affichage.

info Affiche les informations au format **info**.

logon Affiche les informations au format **logon**.

offline
Envoie la sortie de la commande au périphérique de sortie de **conman**. Pour plus d'informations sur ce périphérique, voir «Sortie offline», à la page 374.

Remarque : Les numéros d'invite affectés aux invites globales et locales changent lorsque le plan de production est étendu.

Résultats

La sortie de la commande est générée dans trois formats : **standard**, **keys** et **deps**. Les arguments **keys**, **info** et **logon** modifient l'affichage **deps**.

Exemples

Pour afficher le statut de toutes les `invites` générées sur le poste de travail où vous exécutez le programme **conman**, exécutez la commande suivante :

```
showprompts
```

Voici un exemple de sortie :

```
Message d'état ou invite
ASKED 1(PRMT3) !Voulez-vous continuer ?
INACT 3(CPUA#SCHED_12[(0600 03/12/06),(0AAAAAAAAAAAABST)]) Etes-vous prêt à traiter job1?
INACT 5(CPUA#SCHED_12[(1010 03/12/06),(0AAAAAAAAAAAABSU)]) Etes-vous prêt à traiter job2?
INACT 7(CPUA#SCHED_22[(0600 03/12/06),(0AAAAAAAAAAAABTR)]) Etes-vous prêt à traiter job3?
```

Pour afficher le statut de toutes les `invites mis` générées au format **deps**, exécutez la commande suivante :

```
sp mis@;asked;deps
```

Pour afficher le statut de l'`invite 7` sur le poste de travail `CPUA`, exécutez la commande suivante :

```
sp CPUA#7
```

La commande génère la sortie suivante :

```
INACT 7(CPUA#SCHED_22[(0600 03/12/06),(0AAAAAAAAAAAABTR)]) Etes-vous prêt à traiter job3?
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche comme suit :

1. A partir de la barre de navigation, cliquez sur **Statut et état de santé du système** > **Surveillance de la charge de travail** > **Surveillance des invites**.

2. Sélectionnez **Toutes les invites du plan** ou une autre tâche pour surveiller les invites.
3. Choisissez un nom pour le moteur ou spécifiez les propriété de connexion, puis cliquez sur **OK**.

Format standard

Etat Etat de l'invite.

Message ou invite
Pour les invites nommées, numéro de message, nom et texte de l'invite. Pour les invites non nommées, numéro de message, nom du travail ou du flot de travaux et texte de l'invite.

Format Keys

Les invites sont répertoriées, une par ligne. Les invites nommées sont répertoriées avec les noms et numéros de message qui leur sont associés. Les invites non nommées sont répertoriées avec les numéros de messages qui leur sont associés et avec les noms des travaux ou flots de travaux dans lesquels elles apparaissent sous forme de dépendances.

Format Deps

Les invites utilisées sous forme de dépendances sont répertoriées, suivies des travaux et flots de travaux dépendants. Les travaux sont répertoriés au format **showjobs** standard. Les flots de travaux sont répertoriés au format **showschedules** standard.

Format Deps;keys

Les travaux et les flots de travaux qui comportent des dépendances `prompt` sont répertoriés à raison d'un par ligne.

Format Deps;info

Les invites utilisées sous forme de dépendances sont répertoriées, suivies des travaux et flots de travaux dépendants. Les travaux sont répertoriés au format **showjobs;info**. Les flots de travaux sont répertoriés au format **showschedules** standard.

Format Deps;logon

Les invites utilisées sous forme de dépendances sont répertoriées, suivies des travaux et flots de travaux dépendants. Les travaux sont répertoriés au format **showjobs;logon**. Les flots de travaux sont répertoriés au format **showschedules** standard.

showresources

Affiche les informations relatives aux ressources.

L'information affichée est mise à jour uniquement tant que Tivoli Workload Scheduler (batchman) s'exécute. Le fait que batchman soit actif ou non est vérifié sur l'écran par le message `Batchman LIVES` ou `Batchman down` lorsque vous émettez la commande `conman start`.

Vous devez avoir un accès ***list*** à l'objet qui est affiché si l'option ***enListSecChk*** a été définie sur **yes** sur le gestionnaire de domaine maître lorsque le plan de production a été créé ou étendu.

Syntaxe

{**showresources** | **sr**} [[*poste de travail*#]*nom_ressource*]
[**;keys**]
[**;offline**]

{**showresources** | **sr**} [[*poste de travail*#]*nom_ressource*]
[**;deps**[**;keys** | **info** | **logon**]]
[**;offline**]

Arguments

poste de travail
Indique le nom du poste de travail sur lequel la ressource est définie. Par défaut, le poste de travail sur lequel **conman** est en cours d'exécution est utilisé.

nom_ressource
Indique le nom de la ressource. Les caractères génériques sont acceptés.

keys Affiche la liste composée d'une seule colonne des objets sélectionnés par la commande.

deps Affiche les informations au format **deps**. Utilisez **keys**, **info** ou **logon** pour modifier l'affichage.

info Affiche les informations au format **info**.

logon Affiche les informations au format **logon**.

offline
Envoie la sortie de la commande au périphérique de sortie de **conman**. Pour plus d'informations sur ce périphérique, voir «Sortie offline», à la page 374.

Résultats

La sortie de la commande est générée dans trois formats : **standard**, **keys** et **deps**. Les arguments **keys**, **info** et **logon** modifient l'affichage **deps**.

Exemples

Pour afficher les informations relatives à toutes les ressources sur le poste de travail où vous exécutez le programme **conman**, exécutez la commande suivante :

```
showresources
```

Voici un exemple de sortie :

```
Num UC Ressource    Total disponible  Qté Util par
CPUA #JOBSLOTS  16    16    Pas de réservation pour cette ressource
```

Pour afficher les informations relatives à la ressource `jobslots` sur le poste de travail `CPUA` au format **deps**, exécutez la commande suivante :

```
sr CPUA#JOBSLOTS;deps
```

Voici un exemple de sortie :

```
                                              (Est) (Est)
Workstation Job Stream SchedTime Job State Pr Start Elapse RetCode Dependencies

CPUA    #JOBSLOTS Les dépendances sont les suivantes :


  FTAA   #SCHED_F+ 0600 03/04 ****** HOLD 55(03/04)       [03/04/06];#33
                        (CPUA#)JOBMDM HOLD 30(03/04)       #1(PRMT3);-16 JOBSLOTS-

  FTAA   #SCHED_F+ 1010 03/04 ****** HOLD 55(03/04)       [03/04/06];#34
                        (CPUA#)JOBMDM HOLD 30(03/04)       #1(PRMT3);-16 JOBSLOTS-

  FTAA   #SCHED_F+ 0600 03/05 ****** HOLD 55(03/05)       [03/04/06];#35
                        (CPUA#)JOBMDM HOLD 30(03/05)       #1(PRMT3);-16 JOBSLOTS-
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle qui suit :

1. A partir de la barre de navigation, cliquez sur **Statut et état de santé du système** > **Surveillance de la charge de travail** > **Surveillance des invites**.
2. Sélectionnez **Toutes les ressources du plan** ou une autre tâche pour surveiller les ressources.
3. Choisissez un nom pour le moteur ou spécifiez les propriété de connexion, puis cliquez sur **OK**.

Format standard

CPU Poste de travail sur lequel la ressource est définie.

Resource
Nom de la ressource.

Total Nombre total d'unités de ressources définies.

Available
Nombre d'unités de ressources non allouées.

Qty Nombre d'unités de ressources allouées à un travail ou à un flot de travaux.

Used By
Nom du travail ou du flot de travaux.

Format Keys

Les ressources sont répertoriées à raison d'une par ligne.

Format Deps

Les ressources utilisées sous forme de dépendances sont répertoriées, suivies des travaux et flots de travaux dépendants. Les travaux sont répertoriés au format **showjobs** standard. Les flots de travaux sont répertoriés au format **showschedules** standard.

Format Deps;keys

Les travaux et les flots de travaux qui comportent des dépendances `resource` sont répertoriés à raison d'un par ligne.

Format Deps;info

Les ressources utilisées sous forme de dépendances sont répertoriées, suivies des travaux et flots de travaux dépendants. Les travaux sont répertoriés au format **showjobs;info**. Les flots de travaux sont répertoriés au format **showschedules** standard.

Format Deps;logon

Les ressources utilisées sous forme de dépendances sont répertoriées, suivies des travaux et flots de travaux dépendants. Les travaux sont répertoriés au format **showjobs;logon**. Les flots de travaux sont répertoriés au format **showschedules** standard.

showschedules

Affiche les informations relatives aux flots de travaux.

L'information affichée est mise à jour uniquement tant que Tivoli Workload Scheduler (batchman) s'exécute. Le fait que batchman soit actif ou non est vérifié sur l'écran par le message `Batchman LIVES` ou `Batchman down` lorsque vous émettez la commande `conman start`.

Vous devez avoir un accès ***list*** à l'objet qui est affiché si l'option ***enListSecChk*** a été définie sur **yes** sur le gestionnaire de domaine maître lorsque le plan de production a été créé ou étendu.

Syntaxe

{**showscheds** | **ss**} [*flot_travaux_sélectionné*]
[**;keys**]
[**;offline**]
[**;showid**]

{**showscheds** | **ss**} [*flot_travaux_sélectionné*]
[**;deps**[**;keys** | **info** | **logon**]]
[**;offline**]
[**;showid**]

Arguments

flot_travaux_sélectionné
Voir «Sélection de flots de travaux dans les commandes», à la page 388.

keys Affiche la liste composée d'une seule colonne des objets sélectionnés par la commande.

deps Affiche des informations au format deps. En d'autres termes, les flots de travaux utilisés dans les dépendances de *prédécesseur/successeur* sont répertoriés, suivis des travaux et flots de travaux dépendants. Les travaux sont répertoriés au format `showjobs` de base. Les flots de travaux sont répertoriés au format `showschedules` de base. Utilisez "keys", "info" ou "logon" pour modifier l'affichage "deps".

info Affiche les informations au format **info**.

logon Affiche les informations au format **logon**.

offline
Envoie la sortie de la commande au périphérique de sortie de **conman**. Pour plus d'informations sur ce périphérique, voir «Sortie offline», à la page 374.

showid
Affiche l'identificateur de chaque flot de travaux.

Résultats

La sortie de la commande est générée dans trois formats : standard, **keys** et **deps**. Les arguments **keys**, **info** et **logon** modifient l'affichage **deps**. La liste affichée dans la sortie de la commande n'inclut pas les travaux qui ont été réexécutés lors des processus de planification précédents mais le total final les inclut.

Exemples

Pour afficher le statut du flot de travaux CLEM_DOCOM sur le poste de travail site3 et demander l'affichage de l'identificateur du flot de travaux, exécutez la commande suivante :

```
%ss @#JS_DOCOM ;showid
```

Voici un exemple de résultat de cette commande :

```
                                        (Est) (Est) Jobs Sch
Workstation Job Stream SchedTime State Pr Start Elapse # OK Lim
site3   #JS_DOCOM 0600 11/26 SUCC 10 11/26 00:01 1  1   {0AAAAAAAAAAAACRZ}
```

Pour afficher le statut de tous les flots de travaux à l'état HOLD sur le poste de travail où vous exécutez le programme **conman**, exécutez la commande suivante :

```
showschedules @+state=hold
```

Voici un exemple de sortie de cette commande :

```
                                          (Est) (Est) Jobs Sch
Workstation Job Stream SchedTime State Pr Start Elapse # OK Lim
site3    #FILE_JS1 0600 11/26 HOLD 10 (11/26)    1  0    parms FILE_JS1`
```

Pour afficher le statut de tous les flots de travaux dont le nom commence par sched sur le poste de travail CPUA au format **deps;info**, exécutez la commande suivante :

```
ss CPUA#sched@;deps;info
```

Voici un exemple de sortie :

```
    --------Restart---------
   CPU  Schedule SchedTime Job  JobFile       Opt  Job    Prompt

CPUA #JS_FIRST1[(0600 03/10/06),(0AAAAAAAAAAAABVY)] Les dépendances sont les suivantes :
   CPUA#MOD 0212 03/10
                          JOBMDM /usr/scripts/gl1(B236153+#)JOB_FTA1 echo Start gl1?
   CPUA#MOD 0251 03/10
                          JOBMDM /usr/scripts/gl2(B236153+#)JOB_FTA2 echo Start gl2?
```

Pour afficher **hors ligne** le statut de tous les flots de travaux à l'état ABEND sur tous les postes de travail, exécutez la commande suivante :

```
ss @#@+state=abend;off
```

Pour afficher le statut de tous les flots de travaux sur tous les postes de travail, exécutez la commande suivante :

```
%ss @#@
```

Voici un exemple de sortie de cette commande :

```
                                        (Est) (Est) Jobs Sch
Workstation Job Stream SchedTime State Pr Start Elapse # OK Lim
site3    #JS_DOCOM  0600 11/26 SUCC 10 11/26 00:01  1  1
site3    #JS_SCRIPT 0600 11/26 SUCC 10 11/26 00:03  1  1
site2    #JS_PRED1  1000 11/26 SUCC 10 11/26 00:01  1  1
site3    #JS_SCRIPT1 0600 11/26 ABEND 10 11/26 00:01  1  0
site3    #LFILEJOB  0600 11/26 READY 10         1  0
site1    #RES_100  0600 11/26 SUCC 10 11/26 00:09  1  1
site3    #FILE_JS1  0600 11/26 HOLD 10 (11/26)     1  0    parms FILE_JS1`
site3    #FILE_JOB  0600 11/26 SUCC 10 11/26 00:01  1  1
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle qui suit :

1. Dans la barre de navigation, cliquez sur **Statut et état de santé du système** > **Surveillance de la charge de travail** > **Surveillance des flots de travaux**.
2. Sélectionnez **Tous les flots de travaux du plan** ou une autre tâche pour surveiller les flots de travaux.
3. Choisissez un nom pour le moteur ou spécifiez les propriété de connexion, puis cliquez sur **OK**.

Format standard

UC Poste de travail sur lequel le flot de travaux s'exécute.

Schedule
Nom du flot de travaux.

SchedTime
Date et heure prévues de l'exécution du flot de travaux dans le plan.

State Etat du flot de travaux. Les états sont les suivants :

ADD Le flot de travaux a été ajouté par intervention de l'opérateur.

ABEND
Le flot de travaux s'est terminé avec un code de sortie différent de zéro.

CANCELP
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.

ERROR
Pour les dépendances interréseaux uniquement, une erreur s'est produite lors de la recherche du statut distant.

EXEC Le flot de travaux est en cours d'exécution.

EXTRN
Ne s'applique qu'aux dépendances interréseau. Il s'agit de l'état du flot de travaux EXTERNAL contenant des travaux qui font référence à des travaux ou des flots de travaux dans le réseau distant.

HOLD
Le flot de travaux attend la résolution des dépendances.

READY
Le flot de travaux est prêt à démarrer et toutes les dépendances sont résolues.

STUCK
L'exécution du flot de travaux a été interrompue. Aucun travail n'est lancé sans l'intervention de l'opérateur.

SUCC Le flot de travaux s'est achevé correctement.

Pr Priorité du flot de travaux.

(Est)Start
Heure de début du travail ou du flot de travaux. Les parenthèses indiquent une estimation de l'heure de début. Si la commande est effectuée le même jour que celui où le flot de travaux est planifié pour s'exécuter, le paramètre **Start** affiche une heure (Est)Start. Si la commande est effectuée un autre jour que celui où l'exécution du flot de travaux est planifiée, le paramètre Start affiche une date (Est)Start. Par exemple, si vous possédez le flot de travaux suivant dont l'heure de début a lieu le même jour que celui où le flot de travaux est planifié pour s'exécuter :

```
SCHEDULE MASTERB1#JS_B
ON RUNCYCLE RULE1 "FREQ=DAILY;"
AT 1800
:
MASTERB1#JOB1
END
```

Vous obtenez la sortie suivante :

```
%ss @#@
                                        (Est)  (Est)    Jobs   Sch
CPU      Schedule SchedTime  State Pr Start  Elapse   #  OK  Lim
MASTERB1#JS_B     1800 08/18 HOLD  10(18:00)          1   0
```

Par exemple, si vous possédez le flot de travaux suivant dont l'heure de début a lieu un autre jour que celui où le flot de travaux est planifié pour s'exécuter :

```
SCHEDULE MASTERB1#JS_A
ON RUNCYCLE RULE1 "FREQ=DAILY;"
AT 0500
:
MASTERB1#JOB1
END
```

Vous obtenez la sortie suivante :

```
%ss @#@
                                        (Est)  (Est)  Jobs  Sch
CPU      Schedule SchedTime  State Pr Start  Elapse #  OK Lim
MASTERB1#JS_A     0500 08/19 HOLD  10(08/19)        1   0
```

(Est)Elapse
Heure d'exécution du flot de travaux. Les parenthèses indiquent une estimation basée sur les statistiques consignées.

Jobs # Nombre de travaux du flot de travaux.

Jobs OK
Nombre de travaux dont l'exécution a abouti.

Sch Lim
Nombre maximal de travaux dans un flot de travaux. Si aucune valeur n'est indiquée, aucune limite ne s'applique.

dépendances

Liste des commentaires et des dépendances sur les flots de travaux. Une combinaison des éléments suivants peut être répertoriée :

- Pour une dépendance `de prédécesseur/successeur`, un nom de travail ou de flot de travaux s'affiche. Si le travail ou le flot de travaux est un prédécesseur suspendu, son nom est suivi d'un `[P]`.
- Pour une dépendance `opens`, le nom de fichier s'affiche. Si le fichier réside sur un agent étendu et que son nom comporte plus de 25 caractères, seuls les 25 derniers caractères s'affichent.
- Pour une dépendance `needs`, un nom de ressource placé entre tirets (-) s'affiche. Si le nombre d'unités demandées est supérieur à un, le nombre s'affiche avant le premier tiret.
- Pour une échéance définie par l'argument **until**, l'heure précédée du signe inférieur (<) s'affiche.
- Pour une dépendance `prompt`, le numéro d'invite s'affiche au format #*num*. Pour les invites globales, le nom d'invite vient à la suite, entre parenthèses.
- Les flots de travaux annulés portent la mention **[Cancelled]**.
- Les flots de travaux annulés à l'aide de l'option **;pend** portent la mention **[Cancel Pend]** (Annulation en cours).
- Pour une heure d'échéance **deadline**, l'heure précédée du signe inférieur (<) s'affiche.
- Les flots de travaux contenant le mot clé **carryforward** portent la mention **[Carry]**.
- En ce qui concerne les flots de travaux du plan de production précédent qui ont été reportés, le nom et la date d'origine s'affichent entre crochets.
- Lors du signalement des dépendances horaires, la commande **showschedules** indique dans la colonne **Start** :
  - Uniquement l'heure *hh:mm* si le jour où les dépendances horaires sont définies correspond au jour où la commande **showschedules** est exécutée.
  - Uniquement la date *mm/jj* si le jour où les dépendances horaires sont définies ne correspond pas au jour où la commande **showschedules** est exécutée.

  Remarque : La date ou l'heure affichée dans la colonne **Démarrage** est convertie en fonction du fuseau horaire défini sur le poste de travail sur lequel le flot de travaux doit s'exécuter.

Format Keys

Les flots de travaux sont répertoriés, un par ligne.

Format Deps

Les flots de travaux utilisés comme dépendances sont répertoriés, suivis des travaux et flots de travaux dépendants. Les travaux sont répertoriés au format **showjobs** standard. Les flots de travaux sont répertoriés au format **showschedules** standard.

Format Deps;keys

Les flots de travaux qui comportent des dépendances `de prédécesseur/successeur` sont répertoriés un par ligne.

Format Deps;info

Les flots de travaux utilisés sous forme de dépendances sont répertoriés, suivis des travaux et flots de travaux dépendants. Les travaux sont répertoriés au format **showjobs;info**. Les flots de travaux sont répertoriés au format **showschedules** standard.

Format Deps;logon

Les flots de travaux utilisés sous forme de dépendances sont répertoriés, suivis des travaux et flots de travaux dépendants. Les travaux sont répertoriés au format **showjobs;logon**. Les flots de travaux sont répertoriés au format **showschedules** standard.

shutdown

Arrête sans condition tous les processus de production et services Tivoli Workload Scheduler, y compris **batchman**, **jobman**, **netman**, **mailman**, **appservman**, tous les serveurs **mailman** et tous les processus **writer**.

Même si cette commande arrête le service **appservman**, elle n'arrête pas les services WebSphere Application Server. Pour arrêter les services WebSphere Application Server, exécutez la commande **stopappserver**. Pour plus d'informations, voir «stopappserver», à la page 488.

Sur les postes de travail Windows, la commande **shutdown** n'arrête pas le service **tokensrv**.

Remarque : Cette commande n'est pas prise en charge sur les postes de travail de moteur distant.

Vous devez avoir un accès ***shutdown*** au poste de travail.

Syntaxe

{**shutdown** | **shut**} [**;wait**]

Arguments

wait Attend l'arrêt de tous les processus avant d'inviter l'utilisateur à lancer une autre commande.

Commentaires

La commande **shutdown** arrête les processus uniquement sur le poste de travail sur lequel **conman** est en cours d'exécution. Pour réexécuter **netman** uniquement, exécutez la commande **StartUp**. Pour plus d'informations sur la commande **StartUp**, voir «StartUp», à la page 576. Pour réexécuter l'intégralité de l'arborescence de processus, exécutez les commandes **conman**.

```
start
startappserver
startmon
```

Avant d'exécuter une commande **shutdown**, vous devez exécuter la commande **conman unlink @**.

Exemples

Pour arrêter la production sur le poste de travail où vous exécutez le programme **conman**, exécutez la commande suivante :

```
unlink @
shutdown
```

Pour arrêter la production sur le poste de travail où vous exécutez le programme **conman** et attendre la fin de tous les processus, exécutez la commande suivante :

```
unlink@;noask
shut ;wait
```

start

Démarre les processus de production Tivoli Workload Scheduler, sauf pour le moteur de surveillance d'événements et WebSphere Application Server (voir «startappserver», à la page 481 et «startmon», à la page 483 pour connaître les commandes qui démarrent ces processus).

Remarque : Assurez-vous que la commande **conman start** n'est pas émise pendant l'exécution de **JnextPlan** ou de **stageman**.

Vous devez avoir un accès ***start*** au poste de travail.

Syntaxe

start[*domaine***!**]*poste_de_travail*
[**;mgr**]
[**;noask**]
[**;demgr**]

Arguments

domaine
Indique le nom du domaine dans lequel des postes de travail sont lancés. Les caractères génériques sont acceptés.

Cet argument est utile lors du lancement de plusieurs postes de travail dans un domaine. Par exemple, pour lancer tous les agents du domaine **stlouis**, utilisez la commande suivante :

```
start stlouis!@
```

Si l'argument *domaine* n'est pas indiqué et que l'argument *poste_de_travail* comporte des caractères génériques, le domaine par défaut est celui dans lequel **conman** est en cours d'exécution.

poste de travail
Indique le nom du poste de travail à démarrer. Les caractères génériques sont acceptés.

Cette commande n'est pas prise en charge sur les postes de travail de moteur distant.

mgr Ce paramètre ne peut être indiqué que sur le poste de travail où **conman** est en cours d'exécution. Il démarre le poste de travail local en tant que gestionnaire de domaine. Le poste de travail est le nouveau gestionnaire de domaine et le gestionnaire de domaine en cours devient un agent tolérant aux pannes. Ce format de commande suit généralement une commande **stop**.

Remarque : Pour changer de gestionnaire de domaine, il est préférable d'utiliser une commande **switchmgr**. Pour plus d'informations, voir «switchmgr», à la page 507.

noask Indique que l'utilisateur ne doit pas être invité à confirmer avant d'effectuer une opération sur chaque poste de travail éligible.

demgr Cette option interdit toute ouverture des connexions externes durant la phase de transition entre le moment où un agent démarre en tant qu'ancien gestionnaire de domaine et le moment où la commande **switchmgr** est exécutée, privant ainsi l'agent de la fonction de gestionnaire de domaine. Elle est exécutée automatiquement. Tant que l'ancien gestionnaire de domaine n'a pas traité l'événement **switchmgr** (dans le cas, par exemple, d'un redémarrage retardé ou suivant la réparation d'un agent endommagé), l'option *demgr* **doit** être utilisée pour lancer l'ancien gestionnaire de domaine depuis la ligne de commande locale. Pour plus d'informations sur cette option, voir *Tivoli Workload Scheduler - Guide d'administration*.

Commentaires

La commande **start** est utilisée au début chaque période de production pour réexécuter Tivoli Workload Scheduler après le traitement de préproduction. Elle initialise et lance automatiquement les agents tolérants aux pannes et les agents standard, qui sont connectés automatiquement. Les agents qui ne sont pas connectés automatiquement sont initialisés et exécutés via la commande **link**.

En supposant qu'un utilisateur possède un accès ***start*** aux postes de travail en cours de démarrage, les règles suivantes s'appliquent :

- Un utilisateur exécutant **conman** sur le gestionnaire de domaine maître peut démarrer n'importe quel poste de travail du réseau.
- Un utilisateur exécutant **conman** sur un autre gestionnaire que le gestionnaire de domaine maître peut démarrer n'importe quel poste de travail appartenant à ce domaine ou à des domaines subordonnés. L'utilisateur ne peut pas démarrer des postes de travail appartenant à des domaines homologues.
- Un utilisateur exécutant **conman** sur un agent peut démarrer tout poste de travail hébergé par cet agent.

Exemples

La figure 24, à la page 481 et le tableau 68, à la page 481 qui suivent présentent les postes de travail démarrés par les commandes **start** exécutées par les utilisateurs en différents points du réseau.

DM*n* sont des gestionnaires de domaine et **A***nn* sont des agents.

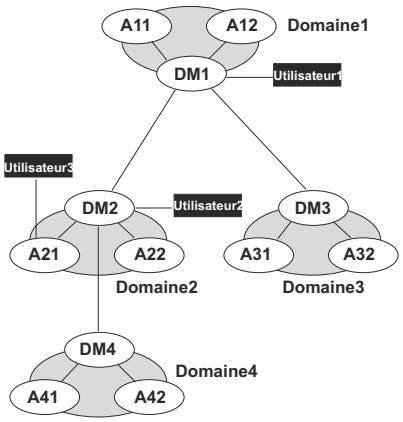


Figure 24. Exemple de réseau

Tableau 68. Postes de travail démarrés

Commande	Démarrage par Utilisateur1	Démarrage par Utilisateur2	Démarrage par Utilisateur3
start @!@	Tous les postes de travail sont démarrés	DM2 A21 A22 DM4 A41 A42	A21
start @	DM1 A11 A12	DM2 A21 A22	A21
start DOMAIN3!@	DM3 A31 A32	Non autorisé	Non autorisé
start DOMAIN4!@	DM4 A41 A42	DM4 A41 A42	Non autorisé
start DM2	DM2	DM2	Non autorisé
start A42	A42	A42	Non autorisé
start A31	A31	Non autorisé	Non autorisé

startappserver

Démarre WebSphere Application Server sur le poste de travail.

Syntaxe

startappserver[*domaine***!**]*poste_de_travail*
[;wait]

Arguments

domaine
Indique le nom du domaine du poste de travail. Indique le nom du domaine du poste de travail. Parce que les postes de travail ont des noms uniques, le domaine n'est pas nécessaire lors du démarrage du serveur WebSphere Application Server sur un poste de travail spécifique. Les caractères génériques sont acceptés.

Si l'argument *domaine* n'est pas indiqué et que l'argument *poste_de_travail* comporte des caractères génériques, le domaine par défaut est celui dans lequel **conman** est en cours d'exécution.

poste de travail
Indique le nom du poste de travail où vous souhaitez démarrer le moteur de surveillance. Les caractères génériques sont acceptés. Si aucun domaine et poste de travail ne sont spécifiés, l'action se fait sur le poste de travail local.

wait Attendez jusqu'au démarrage du serveur WebSphere Application Server avant d'émettre une autre commande.

Commentaires

Le droit de `démarrer` des actions sur des objets `cpu` doit être activé dans le fichier de sécurité pour exécuter cette commande.

WebSphere Application Server peut également être démarré avec l'utilitaire `StartUp`.

startbrokerapp

Démarre l'application Dynamic Workload Broker.

Syntaxe

startbrokerapp [*domaine***!**]*poste_travail*[*;wait*]

Arguments

domaine
Indique le nom du domaine du poste de travail. Dans la mesure où les postes de travail portent des noms uniques, le domaine n'est pas nécessaire lors du démarrage de Dynamic Workload Broker sur un poste de travail spécifique. Les caractères génériques sont acceptés.

Si l'argument *domaine* n'est pas indiqué et que l'argument *poste_de_travail* comporte des caractères génériques, le domaine par défaut est celui dans lequel **conman** est en cours d'exécution.

poste de travail
Indique le nom du poste de travail sur lequel vous souhaitez démarrer Dynamic Workload Broker. Les caractères génériques sont acceptés. Si aucun domaine et poste de travail ne sont spécifiés, l'action se fait sur le poste de travail local.

wait Indique qu'aucune autre commande ne doit être acceptée tant que Dynamic Workload Broker ne s'est pas arrêté.

Commentaires

Le droit de `démarrer` des actions sur des objets `cpu` doit être activé dans le fichier de sécurité pour exécuter cette commande.

Vous pouvez démarrer Dynamic Workload Broker à l'aide de wastool `startBrokerApplication`.

starteventprocessor

Démarre le serveur de traitement des événements sur le gestionnaire de domaine maître, le maître de sauvegarde ou sur un poste de travail installé en tant que maître de sauvegarde, qui fonctionne comme un agent tolérant aux pannes normal.

Syntaxe

{**starteventprocessor** | **startevtp**} [*domaine***!**]*poste de travail*

Arguments

domaine
Indique le nom du domaine du poste de travail.

poste de travail
Indique le nom du poste de travail où vous souhaitez démarrer le serveur de traitement d'événement. Les caractères génériques ne sont pas admis.

Commentaires

Vous pouvez omettre le nom du poste de travail si vous exécutez la commande localement.

Le droit de `démarrer` des actions sur des objets `cpu` doit être activé dans le fichier de sécurité pour exécuter cette commande.

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle qui suit :

1. A partir de la barre de navigation, cliquez sur **Statut et état de santé du système** > **Surveillance de l'environnement** > **Surveillance des postes de travail**.
2. Sélectionnez une tâche pour surveiller les postes de travail.
3. Choisissez un nom pour le moteur ou spécifiez les propriété de connexion, puis cliquez sur **OK**.
4. A partir de la table contenant la liste des postes de travail, sélectionnez un poste de travail et cliquez sur **Plus d'actions** > **Démarrer le processeur d'événement**.

startmon

Démarre le processus monman qui active le moteur de surveillance d'événement sur le poste de travail.

Syntaxe

{**startmon** | **startm**} [*domaine***!**]*poste de travail* [**;noask**]

Arguments

domaine
Indique le nom du domaine du poste de travail. Dans la mesure où les postes de travail portent des noms uniques, le domaine n'est pas nécessaire lors du démarrage du moteur de surveillance sur un poste de travail spécifique. Les caractères génériques sont acceptés.

Si l'argument *domaine* n'est pas indiqué et que l'argument *poste_de_travail* comporte des caractères génériques, le domaine par défaut est celui dans lequel **conman** est en cours d'exécution.

poste de travail
Indique le nom du poste de travail où vous souhaitez démarrer le moteur de surveillance. Les caractères génériques sont acceptés.

noask Indique que l'utilisateur ne doit pas être invité à confirmer avant d'effectuer une opération sur chaque poste de travail éligible.

Commentaires

Le droit de `démarrer` des actions sur des objets `cpu` doit être activé dans le fichier de sécurité pour exécuter cette commande.

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle qui suit :

1. A partir de la barre de navigation, cliquez sur **Statut et état de santé du système** > **Surveillance de l'environnement** > **Surveillance des postes de travail**.
2. Sélectionnez une tâche pour surveiller les postes de travail.
3. Choisissez un nom pour le moteur ou spécifiez les propriété de connexion, puis cliquez sur **OK**.
4. A partir de la table contenant la liste des postes de travail, sélectionnez un poste de travail et cliquez sur **Plus d'actions** > **Démarrer le moniteur d'événements**.

status

Affiche la bannière **conman** ainsi que le statut de production de Tivoli Workload Scheduler.

Syntaxe

{**status** | **stat**}

Résultats

Le mode du plan de production (fichier `Symphony`) est indiqué entre parenthèses, à la suite du mot **schedule** sur la deuxième ligne de la sortie. Le message

d'information **Def** ou **Exp** peut apparaître. **Def** indique que le plan de production est en mode non étendu, et **Exp** indique qu'il est en mode étendu. Le mode du plan de production est déterminé par la valeur de l'option globale ***version étendue***. Avec Tivoli Workload Scheduler version 8.2, les bases de données et les plans sont toujours développés, mais cette information apparaît pour une compatibilité avec les versions antérieures.

Exemples

L'exemple suivant affiche l'état du plan de production en cours.

```
%status
TWS for UNIX/CONMAN 8.4 (1.36.2.22)
Eléments sous licence Propriété d'IBM
5698-WKB
(C) Copyright IBM Corp 1998, 2007
US Government User Restricted Rights
Use, duplication or disclosure restricted by
GSA ADP Schedule Contract with IBM Corp.
Job stream (Exp) 11/26/05 (#34) on site3.
Batchman ACTIF. Limit:19, Fence:0, Audit Level:0
```

stop

Arrête les processus de production Tivoli Workload Scheduler. Pour arrêter le processus **netman**, utilisez la commande **shutdown.** Vous devez avoir un accès ***stop*** au poste de travail.

Syntaxe

stop [*domaine***!**]*poste_de_travail*
[**;wait**]
[**;noask**]

Arguments

domaine
Indique le nom du domaine dans lequel des postes de travail sont arrêtés. Dans la mesure où les postes de travail portent des noms uniques, le domaine n'est pas requis lors de l'arrêt d'un poste de travail spécifique. Les caractères génériques sont acceptés.

Cet argument est utile lors de l'arrêt de plusieurs postes de travail dans un domaine. Par exemple, pour arrêter tous les agents du domaine **stlouis**, utilisez la commande suivante :

```
stop stlouis!@
```

Si l'argument *domaine* n'est pas indiqué et que l'argument *poste_de_travail* comporte des caractères génériques, le domaine par défaut est celui dans lequel **conman** est en cours d'exécution.

poste de travail
Indique le nom du poste de travail à arrêter. Les caractères génériques sont acceptés.

Cette commande n'est pas prise en charge sur les postes de travail de moteur distant.

wait Indique qu'aucune autre commande ne doit être acceptée tant que tous les processus ne sont pas arrêtés.

noask Indique que l'utilisateur ne doit pas être invité à confirmer avant d'effectuer une opération sur chaque poste de travail éligible.

Commentaires

Si la commande **stop** ne peut pas être appliquée à un poste de travail distant (si le chemin TCP/IP n'est pas disponible, par exemple), la commande est stockée en local dans un fichier `pobox` et envoyée au poste de travail une fois celui-ci est connecté.

En supposant qu'un utilisateur possède un accès ***stop*** aux postes de travail en cours d'arrêt, les règles suivantes s'appliquent :

- Un utilisateur exécutant **conman** sur le gestionnaire de domaine maître peut arrêter n'importe quel poste de travail du réseau.
- Un utilisateur exécutant **conman** sur un autre gestionnaire que le gestionnaire de domaine maître peut arrêter n'importe quel poste de travail appartenant à ce domaine ou à des domaines subordonnés. L'utilisateur ne peut pas arrêter des postes de travail appartenant à des domaines homologues.
- Un utilisateur exécutant **conman** sur un agent peut arrêter n'importe quel poste de travail du domaine local.

Lorsque vous lancez une commande **stop @** sur un gestionnaire de domaine, une commande **conman stop** s'exécute en local sur les postes de travail distants. La commande commence son exécution sur les postes de travail situés en bas de la hiérarchie du réseau, pour terminer sur le gestionnaire de domaine. Cependant, le fichier `Symphony` n'est pas mis à jour tant que les postes de travail fonctionnent. C'est pourquoi, si vous lancez une commande **conman sc@!@** depuis un poste de travail, il se peut que les informations résultantes fournissent une image mise à jour des différents états des postes de travail, y compris du gestionnaire de domaine.

Exemples

La figure 25 et le tableau 69, à la page 487 qui suivent présentent les postes de travail arrêtés par les commandes **stop** exécutées par les utilisateurs en différents points du réseau.

DM*n* sont des gestionnaires de domaine et **A***nn* sont des agents.

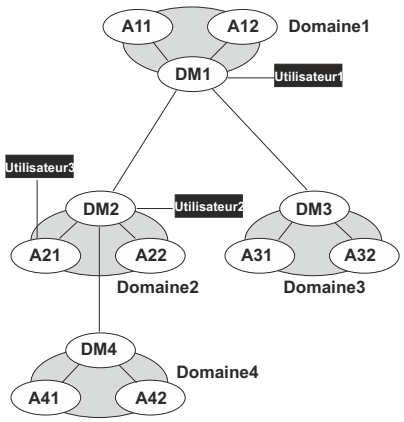


Figure 25. Exemple de réseau

Tableau 69. Postes de travail arrêtés

Commande	Arrêt par Utilisateur1	Arrêt par Utilisateur2	Arrêt par Utilisateur3
stop @!@	Tous les postes de travail sont arrêtés	DM2 A21 A22 DM4 A41 A42	DM2 A21 A22
stop @	DM1 A11 A12	DM2 A21 A22	DM2 A21 A22
stop DOMAIN3!@	DM3 A31 A32	Non autorisé	Non autorisé
stop DOMAIN4!@	DM4 A41 A42	DM4 A41 A42	Non autorisé
stop DM2	DM2	DM2	DM2
stop A42	A42	A42	Non autorisé
stop A31	A31	Non autorisé	Non autorisé

stop ;progressive

Arrête les processus de production Tivoli Workload Scheduler de façon hiérarchique lorsque vous avez défini au moins un poste de travail sur BEHINDFIREWALL dans un réseau Tivoli Workload Scheduler. Similaire à la commande stop @!@, cette commande est plus efficace pour améliorer les performances du plan. La commande ne s'exécute pas à partir du domaine dans lequel elle a été initialement émise pour chaque domaine subordonné, elle s'exécute à chaque niveau de la hiérarchie.

Remarque : Cette commande n'est pas prise en charge sur les postes de travail de moteur distant.

Vous devez avoir un accès ***stop*** au poste de travail.

Syntaxe

stop ;progressive

Commentaires

Lorsque vous émettez la commande sur un gestionnaire de domaine, tous les postes de travail de ce même domaine sont arrêtés. Puis, le gestionnaire de domaine est lui-même arrêté et la commande s'exécute sur chaque domaine subordonné. La commande continue de s'exécuter de la façon hiérarchique suivante : le gestionnaire de domaine arrête les postes de travail du même domaine, s'arrête lui-même, puis la commande s'exécute ensuite sur les domaines subordonnés.

Exemples

La figure 26 et le tableau 70 montrent les postes de travail arrêtés par l'exécution de la commande **stop ;progressive** sur DM2 et DM4.

DM*n* sont des gestionnaires de domaine et **A***nn* sont des agents.

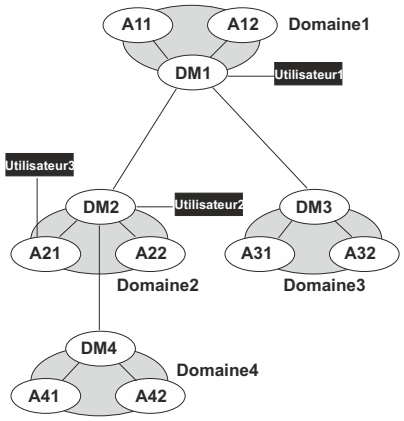


Figure 26. Exemple de réseau

Tableau 70. Postes de travail arrêtés par la commande stop ;progressive

Commande	Arrêt par DM2	Arrêt par DM4
stop ;progressive	A21 A22 DM2	A41 A42 DM4

stopappserver

Arrête WebSphere Application Server sur le poste de travail.

Syntaxe

{**stopappserver**
| **stopapps**} [*domaine***!**]*poste de travail*
[;wait]

Arguments

domaine
Indique le nom du domaine du poste de travail. Etant donné que les postes de travail ont des noms uniques, le domaine n'est pas nécessaire lors de l'arrêt de WebSphere Application Server sur un poste de travail spécifique. Les caractères génériques sont acceptés.

Si l'argument *domaine* n'est pas indiqué et que l'argument *poste_de_travail* comporte des caractères génériques, le domaine par défaut est celui dans lequel **conman** est en cours d'exécution.

poste de travail
Indique le nom du poste de travail sur lequel vous souhaitez arrêter le

moteur de surveillance. Les caractères génériques sont acceptés. Si aucun domaine et poste de travail ne sont spécifiés, l'action se fait sur le poste de travail local.

wait Attendez jusqu'à l'arrêt du serveur WebSphere Application Server avant d'émettre une autre commande.

Commentaires

Le droit d'`arrêter` des actions sur des objets `cpu` doit être activé dans le fichier de sécurité pour exécuter cette commande.

Sous Windows, les systèmes s'abstiennent d'utiliser les services Windows pour arrêter WebSphere Application Server. Si vous utilisez les services Windows, le processus *appserverman*, dont l'exécution se poursuit, redémarre WebSphere Application Server. Utilisez cette commande ou la commande **stopWas** à la place (sans l'option **-direct**).

Lorsque vous exécutez la commande, le processus *appserverman* vérifie d'abord si WebSphere Application Server peut extraire des données d'identification de l'utilisateur (nom d'utilisateur et mot de passe) du fichier `soap.client.props` situé dans le profil WebSphere Application Server. S'ils ne peuvent être extraits, *appserverman* les lit à partir du fichier `useropts` de l'utilisateur et exécute le script `stopServer.sh (bat)` pour les transmettre à WebSphere Application Server.

Pour pouvoir exécuter la commande, vous devez donc terminer l'une des deux procédures de personnalisation afin de fournir les données d'identification de l'utilisateur à WebSphere Application Server :

- Personnalisez les propriétés de nom d'utilisateur (`com.ibm.SOAP.loginUserid`) et de mot de passe (`com.ibm.SOAP.loginPassword`) dans le fichier `soap.client.props` situé dans :

chemin_profil_WAS`/properties` (Version 9.1 et maître et agent ultérieurs)

  où *chemin_profil_WAS* correspond au chemin d'accès du profil WebSphere Application Server que vous avez indiqué durant l'installation. Le chemin d'accès par défaut est *<rép_principal_TWA>*`/WAS/TWSprofile`.

  Vous devez également :

  1. Définir la propriété `com.ibm.SOAP.securityEnabled` par `true` dans le même fichier pour activer la sécurité du client SOAP.
  2. Exécuter le script **encryptProfileProperties.sh** pour chiffrer le mot de passe. Pour plus d'informations sur cet outil du serveur d'applications, voir le manuel *Tivoli Workload Scheduler - Guide d'administration*.

- Personnaliser la section `Attributs des connexions conman` du fichier `localopts` en spécifiant les détails du connecteur ou du gestionnaire de domaine maître.

  Vous devez également :

  1. Créer (ou personnaliser, s'il existe déjà) manuellement le fichier `useropts`, en ajoutant les attributs `USERNAME` et `PASSWORD` de l'utilisateur qui exécutera `stopappserver`. Vérifiez que le nom de fichier `useropts` est entré dans la clé `USEROPTS` de la section `Attributs des connexions conman (CLI)`. Voir *Tivoli Workload Scheduler - Guide d'administration* pour plus de détails.
  2. Pour chiffrer le mot de passe dans le fichier `useropts`, il suffit d'exécuter `conman`.

stopbrokerapp

Arrête l'application Dynamic Workload Broker.

Syntaxe

{**stopbrokerapp**} [*domaine***!**]*poste_de_travail*

Arguments

domaine
Indique le nom du domaine du poste de travail. Dans la mesure où les postes de travail portent des noms uniques, le domaine n'est pas nécessaire lors de l'arrêt de Dynamic Workload Broker sur un poste de travail spécifique. Les caractères génériques sont acceptés.

Si l'argument *domaine* n'est pas indiqué et que l'argument *poste_de_travail* comporte des caractères génériques, le domaine par défaut est celui dans lequel **conman** est en cours d'exécution.

poste de travail
Indique le nom du poste de travail sur lequel vous souhaitez arrêter Dynamic Workload Broker. Les caractères génériques sont acceptés. Si aucun domaine et poste de travail ne sont spécifiés, l'action se fait sur le poste de travail local.

Commentaires

Le droit d'`arrêter` des actions sur des objets `cpu` doit être activé dans le fichier de sécurité pour exécuter cette commande.

Vous pouvez également arrêter Dynamic Workload Broker à l'aide de wastool `stopBrokerApplication`.

stopeventprocessor

Arrête le serveur de traitement d'événements.

Syntaxe

{**stopeventprocessor** | **stopevtp**} [*domaine***!**][*poste de travail*]

Arguments

domaine
Indique le nom du domaine du poste de travail.

poste de travail
Indique le nom du gestionnaire de domaine maître, du maître de sauvegarde ou du poste de travail installé en tant que maître de sauvegarde, qui fonctionne comme un agent tolérant aux pannes normal, sur lequel vous souhaitez arrêter le serveur de traitement des événements. Les caractères génériques ne sont pas admis.

Vous pouvez omettre le nom du poste de travail si vous exécutez la commande localement.

Commentaires

Cette commande ne peut pas être émise de façon asynchrone.

Si vous émettez la commande à partir d'un poste de travail différent de celui sur lequel le processus d'événements est configuré, la commande utilise le client de ligne de commande, de manière à définir correctement les données d'identification de l'utilisateur du client de ligne de commande.

Le droit d'`arrêter` des actions sur des objets `cpu` doit être activé dans le fichier de sécurité pour exécuter cette commande.

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle qui suit :

1. A partir de la barre de navigation, cliquez sur **Statut et état de santé du système** > **Surveillance de l'environnement** > **Surveillance des postes de travail**.
2. Sélectionnez une tâche pour surveiller les postes de travail.
3. Choisissez un nom pour le moteur ou spécifiez les propriété de connexion, puis cliquez sur **OK**.
4. A partir de la table contenant la liste des postes de travail, sélectionnez un poste de travail et cliquez sur **Plus d'actions** > **Arrêter le processeur d'événement**.

stopmon

Arrête le moteur de surveillance sur le poste de travail.

Syntaxe

{**stopmon** |
stopm} [*domaine***!**]*poste de travail*
[**;wait**]
[**;noask**]

Arguments

domaine
Indique le nom du domaine du poste de travail. Dans la mesure où les postes de travail portent des noms uniques, le domaine n'est pas nécessaire lors de l'arrêt du moteur de surveillance sur un poste de travail spécifique. Les caractères génériques sont acceptés.

Si l'argument *domaine* n'est pas indiqué et que l'argument *poste_de_travail* comporte des caractères génériques, le domaine par défaut est celui dans lequel **conman** est en cours d'exécution.

poste de travail
Indique le nom du poste de travail sur lequel vous souhaitez arrêter le moteur de surveillance. Les caractères génériques sont acceptés.

wait Indique qu'aucune autre commande ne doit être acceptée tant que le moteur de surveillance ne s'est pas arrêté.

noask Indique que l'utilisateur ne doit pas être invité à confirmer avant d'effectuer une opération sur chaque poste de travail éligible.

Commentaires

Le moteur de surveillance est redémarré automatiquement lorsque le plan de production suivant est activé (sous Windows également lorsque Tivoli Workload Scheduler est redémarré) à moins que vous ne désactiviez l'option locale `autostart monman`.

La commande est asynchrone, à moins de spécifier le mot-clé `wait`.

Le droit d'`arrêter` des actions sur des objets `cpu` doit être activé dans le fichier de sécurité pour exécuter cette commande.

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle qui suit :

1. A partir de la barre de navigation, cliquez sur **Statut et état de santé du système** > **Surveillance de l'environnement** > **Surveillance des postes de travail**.
2. Sélectionnez une tâche pour surveiller les postes de travail.
3. Choisissez un nom pour le moteur ou spécifiez les propriété de connexion, puis cliquez sur **OK**.
4. A partir de la table contenant la liste des postes de travail, sélectionnez un poste de travail et cliquez sur **Plus d'actions** > **Arrêter le moniteur d'événements**.

submit docommand

Soumet une commande à lancer sous forme de travail.

Pour exécuter cette commande, dans le fichier de sécurité, vous devez disposer d'un accès ***submit*** pour le travail portant le nom spécifié dans sa définition de base de données et, si vous utilisez le mot clé ***alias***, portant également le nom spécifié avec ce mot clé. De plus, si vous utilisez le mot clé ***recoveryjob***, vous devez disposer d'un accès ***submit*** pour le travail spécifié avec ce mot clé.

Pour inclure des dépendances `needs` et `prompt`, vous devez avoir un accès ***use*** aux ressources et aux invites globales.

Si vous soumettez le travail sur un poste de travail autre que le gestionnaire de domaine maître, vous devez vous connecter en tant qu'utilisateur :

- détenant les autorisations d'accès appropriées définies dans le fichier le fichier useropts pour vous connecter au gestionnaire de domaine maître par l'intermédiaire de WebSphere Application Server
- autorisé à exécuter les commandes submit dans le fichier de sécurité stocké sur le gestionnaire de domaine maître

Syntaxe

{**submit**
docommand | **sbd**} [*poste de travail*#]"*cmd*"
[**;alias**[=*nom*]]
[**;into=**[*poste de travail*#]
{*ID_flot_travaux*;**schedid** |*nom_flot_travaux* ([*hhmm*[*date*]])}]
[**;***option_travail*[**;**...]]

Arguments

poste de travail
Indique le nom du poste de travail sur lequel le travail va être lancé. Les caractères génériques sont admis, auquel cas, le travail est lancé sur les postes de travail éligibles. Par défaut, le poste de travail sur lequel **conman** est en cours d'exécution est utilisé. Vous ne pouvez pas indiquer une classe de postes de travail ou un domaine.

Remarque : Du fait d'une limitation dans la façon dont Windows gère le signe égal (=) dans l'environnement shell, vous devez masquer le signe égal (=) comme suit '\='\ lorsque vous soumettez des commandes Windows à l'aide de **submit docommand**. Par exemple, pour définir la variable locale **var1** sur *hello*, vous devez émettre la commande suivante :

```
%sbd "set var1\"=\"hello"
```

commande
Indique une commande système valide comportant jusqu'à 255 caractères. L'intégralité de la commande doit être placée entre guillemets ("). La commande est traitée sous forme de travail et toutes les règles relatives aux travaux s'appliquent.

alias=*nom*
Indique un nom unique à attribuer au travail. Si le mot clé **alias** est entré sans qu'aucun nom ne soit indiqué, un nom est créé à partir des six premiers caractères alphanumériques (en majuscules) de la commande, selon le nombre de caractères de la commande, suivis d'un nombre aléatoire de dix chiffres. Si la commande contient des caractères blancs, le nom est créé à partir des six premiers caractères alphanumériques précédant le caractère blanc. Par exemple, dans le cas de la commande **rm apfile**, le nom généré est similaire à **RM0123456789**. Si la commande comporte plus de six caractères alphanumériques telle que **"wlsinst"**, le nom créé est **WLSINS0396578515**.

Si aucun **alias** n'est indiqué la première fois que vous soumettez la commande, un nom de travail est créé à partir des 255 caractères du nom de la commande. Si vous soumettez une commande une seconde fois à partir du même poste de travail, le mot clé **alias** est obligatoire et doit être unique pour chaque commande soumise.

into=*instance_flot_travaux*
Identifie l'instance de flot de travaux dans laquelle le travail sera placé pour être lancé. Sélectionnez l'instance de flot de travaux comme suit :

[*poste_de_travail***#**]*nom_flot_travaux*([*hhmm*[*date*]])

ou

[*poste_de_travail***#**]*ID_flot_travaux* **;schedid**

Si l'argument **into** n'est pas indiqué, le travail est ajouté à un flot de travaux appelé **JOBS**.

option_travail
Indiquez l'un des arguments suivants :

at=*hhmm* [**timezone** | **tz** *fuseau_horaire*] [**+***n* **days** | *mm/jj[/aa]*] | [**absolute** | **abs**]

confirmed

critical

deadline=*heure* [**timezone|tz** *fuseau_horaire*][**+***n* **day**[**s** | *mm/jj*[*/aa*]]

every=*fréquence*

follows=[*netagent***::**][*workstation***#**]{*jobstreamname*[*hhmm* [*mm/jj*[*/aa*]]][*.travail* | **@**] | *ID_flot_travaux.travail***;schedid**} | *travail*[**;nocheck**][**;wait=***heure*][**,**...]

Remarque : L'argument **;nocheck** n'est pas pris en charge dans les dépendances interréseaux.

interactive

Remarque : Ce mot-clé ne peut être utilisé que dans les environnements Windows.

maxdur=*heure*[**onmaxdur** *action*]

mindur=*heure*[**onmindur** *action*]

logon=*utilisateur*.

needs=[*num*] [*poste_de_travail***#**]*ressource*[**,**...]

opens=[*poste_de_travail***#**]"*file_name*"[**(***qualifiant***)**][**,**...]

priority=[*pri* | **hi** | **go**]

prompt="[**:** | **!**]*texte***"** | *nom_invite*[**,**...]

rccondsucc"*Condition de succès*"

recovery=stop | continue | rerun

recoveryjob= [*poste_de_travail*#]*nom_travail*

Nom du travail de reprise différent de celui spécifié dans la définition de travail de la base de données (le cas échéant).

after [*poste_de_travail*#]*nom_travail*

abendprompt "*texte*"

until *heure* [**timezone|tz** *fuseau_horaire*][**+***n* **day**[**s**] | [**absolute** | **abs**]] [**;onuntil** *action*]

La valeur par défaut de ***option_travail*** est l'utilisateur du poste de travail sur lequel la commande est exécutée.

Utilisation des paramètres locaux

Vous pouvez utiliser les paramètres locaux comme des valeurs associées aux mots clés suivants :

- `commande`
- `opens`
- `logon`
- `invite`
- `abendprompt`

Les paramètres locaux sont définis et gérés à l'aide de la commande d'utilitaire parms dans une base de données locale du poste de travail où le travail est exécuté. Les paramètres sont résolus sur le poste de travail pendant l'exécution de la commande **submit**.

Commentaires

Les travaux soumis en production à partir de la ligne de commande **conman** ne sont pas inclus dans le plan de préproduction et ne peuvent donc pas être pris en compte lors de l'identification des prédécesseurs des dépendances de prédécesseur/successeur externes.

Si aucun *poste de travail* n'est indiqué avec les options `follows`, `needs` ou `opens` ou `into`, le poste de travail sur lequel se trouve le travail est utilisé par défaut.

L'agenda classe les dépendances de prédécesseur/successeur comme *internes* lorsqu'elles ne sont spécifiées que par leur nom de travail dans le flot de travaux. Il les classe en temps que dépendances *externes* lorsqu'elles sont spécifiées dans le format *jobStreamName.workstationName.jobName*.

Lorsque vous soumettez l'objet dans un flot de travaux et que vous ajoutez une dépendance de prédécesseur/successeur qui a le même nom de flot de travaux (par exemple, vous soumettez l'objet dans le flot de travaux `schedA` et vous définissez une dépendance de prédécesseur/successeur sur `schedA.job2`), la dépendance est traitée comme une dépendance de prédécesseur/successeur *externe*. Depuis la version 8.3, contrairement aux versions précédentes, étant donné que l'agenda utilise le critère de correspondance `sameday` pour résoudre des dépendances externes, les dépendances générées de cette manière ne sont jamais ajoutées lors de la première soumission de l'objet.

Exemples

Pour soumettre une commande **rm** dans le flot de travaux `JOBS` avec une dépendance de `prédécesseur/successeur`, exécutez la commande suivante :

```
submit docommand="rm apfile";follows sked3
```

Pour soumettre une commande **sort** caractérisée par l'alias **sortit** et placer le travail dans le flot de travaux `reports` dont l'heure d'exécution **at** est définie sur 17:30, exécutez la commande suivante :

```
sbd "sort < file1 > file2";alias=sortit;into=reports;at=1730
```

Pour soumettre la commande **chmod** sur tous les postes de travail dont le nom commence par `site`, exécutez la commande suivante :

```
sbd site@#"chmod 444 file2";alias
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle décrite dans :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Soumission de travaux ad hoc".

submit file

Soumet un fichier à lancer sous forme de travail.

Pour exécuter cette commande, dans le fichier de sécurité, vous devez disposer d'un accès ***submit*** pour le travail portant le nom spécifié dans sa définition de base de données et, si vous utilisez le mot clé ***alias***, portant également le nom spécifié

avec ce mot clé. De plus, si vous utilisez le mot clé ***recoveryjob***, vous devez disposer d'un accès ***submit*** pour le travail spécifié avec ce mot clé.

Pour inclure des dépendances `needs` et `prompt`, vous devez avoir un accès ***use*** aux ressources et aux invites globales.

Si vous soumettez le travail sur un poste de travail autre que le gestionnaire de domaine maître, vous devez vous connecter en tant qu'utilisateur :

- Détenant les autorisations d'accès appropriées définies dans le fichier le fichier useropts pour vous connecter au gestionnaire de domaine maître par l'intermédiaire de WebSphere Application Server
- Autorisé à exécuter les commandes submit dans le fichier de sécurité stocké sur le gestionnaire de domaine maître

Syntaxe

{**submit**
file | **sbf**} **"***file_name***"**
[**;alias**[**=***nom*]]
[**;into=**[*poste_de_travail***#**]{*ID_flot_travaux*
;schedid | *nom_flot_travaux*([*hhmm*[*date*]])}]
[**;***option_travail*[**;...**]]
[**;noask**]

Arguments

file_name
Indique le nom du fichier (jusqu'à 255 caractères). Les caractères génériques sont acceptés. Ce nom doit être placé entre guillemets (") s'il contient des caractères autres que les suivants : tirets (-), barres obliques (/) et traits de soulignement (_). Reportez-vous aux exemples.

alias=*nom*
Indique un nom unique à attribuer au travail. Si le mot clé **alias** est entré sans qu'aucun nom ne soit indiqué, un nom est créé à partir des six premiers caractères alphanumériques (en majuscules) du nom de fichier, selon le nombre de caractères du nom de fichier, suivis d'un nombre aléatoire de dix chiffres. Par exemple, dans le cas où le nom de fichier est `jclttx5`, le nom généré est similaire à `JCLTTX0123456789`.

Si vous n'indiquez pas d'**alias**, un nom de fichier est créé à partir des 255 caractères alphanumériques du nom de base du fichier, en majuscules.

Dans l'un ou l'autre des cas ci-dessus, si le nom du fichier ne commence pas par une lettre, vous êtes invité à utiliser **alias=** *nom*.

Si vous soumettez un fichier une seconde fois à partir du même poste de travail, le mot clé **alias** est obligatoire et doit être unique pour chaque fichier soumis.

into=*instance_flot_travaux*
Identifie l'instance de flot de travaux dans laquelle le travail sera placé pour être lancé. Sélectionnez l'instance de flot de travaux comme suit :

[*poste_de_travail***#**]*nom_flot_travaux*([*hhmm*[*date*]])

ou

[*poste_de_travail***#**]*ID_flot_travaux* **;schedid**

Si l'argument **into** n'est pas indiqué, le travail est ajouté à un flot de travaux appelé **JOBS**.

option_travail
Indiquez l'un des arguments suivants :

at=*hhmm* [**timezone**|**tz** *fuseau_horaire*] [**+***n* **days** | *mm/jj[/aa]*] | [**absolute** | **abs**]

confirmed

critical

deadline=*heure*[**timezone** | **tz** *fuseau_horaire*][**+***n* **days** | *mm/jj[/aa]*]

every=*fréquence*

follows=[*agent_réseau***::**][*poste_de_travail***#**]{*nom_flot_travaux*(*hhmm* [*mm/jj[/aa]*]) [*.travail* | **@**] | *ID_flot_travaux.travail***;schedid**}| *travail*[**;nocheck**][**;wait**=*heure*][**,**...]

Remarque : L'argument **;nocheck** n'est pas pris en charge dans les dépendances interréseaux.

interactive

Remarque : Ce mot-clé ne peut être utilisé que dans les environnements Windows.

logon=*utilisateur*

maxdur=*heure*[**onmaxdur** *action*]

mindur=*heure*[**onmindur** *action*]

needs=[*num*] [*poste_de_travail***#**]*ressource*[**,**...]

opens=[*poste_de_travail***#**]"*file_name*"[**(***qualifiant***)**][**,**...]

priority=[*pri* | **hi** | **go**]

prompt="[**:** | **!**]*texte***"** | *nom_invite*[**,**...]

rccondsucc"*Condition de succès*"

recovery=stop | continue | rerun

recoveryjob= [*poste_de_travail*#]*nom_travail*

Nom du travail de reprise différent de celui spécifié dans la définition de travail de la base de données (le cas échéant).

after [*poste_de_travail*#]*nom_travail*

abendprompt *"texte"*

until *heure* [**timezone**|**tz** *fuseau_horaire*][**+***n* **day**[**s**] | [**absolute** | **abs**]] [**;onuntil** *action*]

noask Indique que l'utilisateur ne doit pas être invité à confirmer avant d'effectuer une opération sur chaque fichier éligible.

Utilisation des paramètres locaux

Vous pouvez utiliser les paramètres locaux comme des valeurs associées aux mots clés suivants :

- `opens`

- `logon`
- `invite`
- `abendprompt`

Les paramètres locaux sont définis et gérés à l'aide de la commande d'utilitaire parms dans une base de données locale du poste de travail où le travail est exécuté. Les paramètres sont résolus sur le poste de travail pendant l'exécution de la commande **submit**.

Commentaires

Les travaux soumis en production à partir de la ligne de commande **conman** ne sont pas inclus dans le plan de préproduction et ne peuvent donc pas être pris en compte lors de l'identification des prédécesseurs des dépendances de prédécesseur/successeur externes.

Si vous ne spécifiez pas de poste de travail avec `follows`, `needs`, `opens` ou `into`, la valeur par défaut est le poste de travail sur lequel est exécuté **conman**.

L'agenda classe les dépendances de prédécesseur/successeur comme *internes* lorsqu'elles ne sont spécifiées que par leur nom de travail dans le flot de travaux. Il les classe en temps que dépendances *externes* lorsqu'elles sont spécifiées dans le format *jobStreamName.workstationName.jobName*.

Lorsque vous soumettez l'objet dans un flot de travaux et que vous ajoutez une dépendance de prédécesseur/successeur qui a le même nom de flot de travaux (par exemple, vous soumettez l'objet dans le flot de travaux `schedA` et vous définissez une dépendance de prédécesseur/successeur sur `schedA.job2`), la dépendance est traitée comme une dépendance de prédécesseur/successeur *externe*. Depuis la version 8.3, contrairement aux versions précédentes, étant donné que l'agenda utilise le critère de correspondance `sameday` pour résoudre des dépendances externes, les dépendances générées de cette manière ne sont jamais ajoutées lors de la première soumission de l'objet.

Exemples

Pour soumettre un fichier dans le flot de travaux `jobs` en utilisant le nom de travail `myjcl`, exécutez la commande suivante :

```
submit file=d:\jobs\lib\daily\myjcl
```

où la séquence `;into` a été omise.

Pour soumettre un fichier dans le flot de travaux `missked` en utilisant le nom de travail `misjob4`, exécutez la commande suivante :

```
sbf /usr/lib/mis/jcl4;alias=misjob4;into=missked ;needs=2 slots
```

Le travail requiert deux unités de la ressource `slots`.

Pour soumettre tous les fichiers dont le nom commence par `back` dans le flot de travaux `bkup`, exécutez la commande suivante :

```
sbf "/usr/lib/backup/back@";into=bkup
```

Pour soumettre le fichier `tws_env.cmd` dont le chemin d'accès contient un espace, sur un poste de travail Windows, exécutez :

- En mode interactif :

```
sbf "\"C:\Program Files\IBM\TWS\lucaMDM\tws_env.cmd\"";alias=MYJOB
```

Sous Windows, les guillemets (") doivent être gérés par la séquence de caractères "\.

- En mode ligne de commande :

```
conman sbf "\"\\\"C:\Program Files\IBM\TWS\lucaMDM\tws_env.cmd\\\"\"";alias=MYJOB
```

Sous Windows, lorsque la commande est exécutée en externe à partir de l'environnement conman, la séquence d'échappement est plus longue.

où "\" est le caractère d'échappement de l'espace dans le chemin d'accès au fichier.

submit job

Soumet un travail à lancer.

Pour exécuter cette commande, dans le fichier de sécurité, vous devez disposer d'un accès ***submit*** (***submitdb***) pour le travail portant le nom spécifié dans sa définition de base de données et, si vous utilisez le mot clé ***alias***, portant également le nom spécifié avec ce mot clé. De plus, si vous utilisez le mot clé ***recoveryjob***, vous devez disposer d'un accès ***submit*** pour le travail spécifié avec ce mot clé.

Notez que si vous disposez uniquement des droits de sécurité ***submitdb***, vous êtes limité à soumettre les travaux définis dans la base de données. Vous ne pouvez pas soumettre de travaux ad-hoc.

Pour inclure des dépendances `needs` et `prompt`, vous devez avoir un accès ***use*** aux ressources et aux invites globales.

Si vous soumettez le travail sur un poste de travail autre que le gestionnaire de domaine maître, vous devez vous connecter en tant qu'utilisateur :

- Détenant les autorisations d'accès appropriées définies dans le fichier le fichier useropts pour vous connecter au gestionnaire de domaine maître par l'intermédiaire de WebSphere Application Server
- Autorisé à exécuter les commandes **submit** dans le fichier de sécurité stocké sur le gestionnaire de domaine maître

Si vous soumettez un travail reflet, voir Chapitre 19, «Définition et gestion des dépendances croisées», à la page 681 pour plus détails.

Syntaxe

{**submit**
job | **sbj**} [*poste de travail*#]*nom_travail*
[**;alias**[**=***nom*]]
[**;into=**[*poste_de_travail***#**]{*ID_flot_travaux*
;schedid | *nom_flot_travaux*([*hhmm*[*date*]])}]
[**;***option_travail*[**;**...]]
[**;vartable=***nom_table*]
[**;noask**]

Arguments

poste de travail
Indique le nom du poste de travail sur lequel le travail va être lancé. Les caractères génériques sont admis, auquel cas, le travail est lancé sur les

postes de travail éligibles. Par défaut, le poste de travail sur lequel `conman` est en cours d'exécution est utilisé. Vous ne pouvez pas indiquer une classe de postes de travail ou un domaine.

nom_travail
Indique le nom du travail. Les caractères génériques sont admis, auquel cas, tous les travaux éligibles sont soumis. Si le travail figure déjà dans le plan de production et qu'il est soumis dans le même flot de travaux, vous devez utiliser le même argument **alias** pour attribuer un nom unique.

alias=*nom*
Indique un nom unique à attribuer au travail à la place de l'option *nom_travail*. Si le mot clé **alias** est entré sans spécifier de nom, un nom est créé à partir des cinq premiers caractères alphanumériques du *nom_travail*, suivis par un nombre aléatoire à dix chiffres. Le nom apparaît systématiquement en majuscules. Par exemple, si *nom_travail* est défini sur `jcrttx5`, le nom généré sera similaire à `JCRTT1234567890`.

into=*instance_flot_travaux*
Identifie l'instance de flot de travaux dans laquelle le travail sera placé pour être lancé. Sélectionnez l'instance de flot de travaux comme suit :

[*poste_de_travail***#**]*nom_flot_travaux*([*hhmm*[*date*]])

ou

[*poste_de_travail***#**]*ID_flot_travaux* **;schedid**Si l'argument **into** n'est pas indiqué, le travail est ajouté à un flot de travaux appelé **JOBS**.

option_travail
Indiquez l'un des arguments suivants :

at=*hhmm* [**timezone**|**tz** *fuseau_horaire*] [**+***n* **days** | *mm/jj[/aa]*] | [**absolute** | **abs**]

confirmed

critical

deadline=*heure*[**timezone** | **tz** *fuseau_horaire*][**+***n* **days** | *mm/jj[/aa]*]

every=*fréquence*

follows=[*agent_réseau***::**][*poste_de_travail***#**]{*nom_flot_travaux*(*hhmm* [*mm/jj[/aa]*]) [.*travail* | **@**] | *ID_flot_travaux*.*travail***;schedid**}| *travail*[**;nocheck**][**;wait=***heure*][**,**...]

Remarque : L'argument **;nocheck** n'est pas pris en charge dans les dépendances interréseaux.

maxdur=*heure*[**onmaxdur** *action*]

mindur=*heure*[**onmindur** *action*]

needs=[*num*] [*poste_de_travail***#**]*ressource*[**,**...]

opens=[*poste_de_travail***#**]"*file_name*"[**(***qualifiant***)**][**,**...]

priority=[*pri* | **hi** | **go**]

prompt="[**:** | **!**]*texte***"** | *nom_invite*[**,**...]

rccondsucc"*Condition de succès*"

recovery=stop | continue | rerun

recoveryjob= [*poste_de_travail***#**]*nom_travail*

Nom du travail de reprise différent de celui spécifié dans la définition de travail de la base de données (le cas échéant).

after [*poste_de_travail*#]*nom_travail*

abendprompt "*texte*"

until *heure* [**timezone** | **tz** *fuseau_horaire*][+*n* **day**[**s**] | [**absolute** | **abs**]] [**;onuntil** *action*]

vartable=*nom_table*
Indique le nom de la table de variables, s'il est différent du nom par défaut, dans laquelle sont définies les variables que vous souhaitez utiliser.

A faire :

- Avec cette commande, vous pouvez utiliser une substitution de variable pour les mots clés suivants :
  - `opens`
  - `invite`
  - `abendprompt`
- Placez la variable entre accents circonflexes (^), puis l'ensemble de la chaîne entre guillemets. Si la variable contient une partie d'un chemin, vérifiez que les accents circonflexes ne sont pas immédiatement précédés d'une barre oblique inversée (\). En effet, la séquence \^ pourrait être interprétée comme une séquence d'échappement et serait alors résolue par le programme d'analyse syntaxique comme un accent circonflexe. Si nécessaire, placez la barre oblique inversée dans la définition de la variable entre accents circonflexes.
- Les variables spécifiées dans la définition du travail avec le format ${*nom de variable*} ne sont pas résolues.

Si vous soumettez un travail contenant des variables définies dans une table de variables qui n'est pas la table de variables par défaut et que vous n'indiquez pas cette table dans le cycle d'exécution, dans le flot de travaux ou sur le poste de travail, les variables ne sont pas résolues. Pour plus d'informations, voir Chapitre 6, «Personnalisation de votre charge de travail à l'aide des tables de variables», à la page 121.

noask Indique que l'utilisateur ne doit pas être invité à confirmer avant d'effectuer une opération sur chaque travail éligible.

Commentaires

Les travaux soumis en production à partir de la ligne de commande **conman** ne sont pas inclus dans le plan de préproduction et ne peuvent donc pas être pris en compte lors de l'identification des prédécesseurs des dépendances de prédécesseur/successeur externes.

Si aucun poste de travail n'est indiqué avec les options `follows`, `needs` ou `opens` ou `into`, le poste de travail sur lequel se trouve le travail est utilisé par défaut.

at indique à quel moment peut être soumis le travail. Si le mot clé **at** est utilisé, le travail ne peut pas démarrer avant l'heure définie dans ce mot clé. Notez que si le gestionnaire de domaine maître de votre réseau s'exécute lorsque la valeur `yes` est attribuée aux options `enLegacyStartOfDayEvaluation` et `enTimeZone` afin de convertir l'heure `startOfDay` du gestionnaire de domaine maître en fuseau horaire local sur chaque poste de travail du réseau, vous devez ajouter le mot clé **absolute** pour le faire fonctionner.

L'agenda classe les dépendances de prédécesseur/successeur comme *internes* lorsqu'elles ne sont spécifiées que par leur nom de travail dans le flot de travaux. Il les classe en temps que dépendances *externes* lorsqu'elles sont spécifiées dans le format *jobStreamName.workstationName.jobName*.

Lorsque vous soumettez l'objet dans un flot de travaux et que vous ajoutez une dépendance de prédécesseur/successeur qui a le même nom de flot de travaux (par exemple, vous soumettez l'objet dans le flot de travaux `schedA` et vous définissez une dépendance de prédécesseur/successeur sur `schedA.job2`), la dépendance est traitée comme une dépendance de prédécesseur/successeur *externe*. Depuis la version 8.3, contrairement aux versions précédentes, étant donné que l'agenda utilise le critère de correspondance `sameday` pour résoudre des dépendances externes, les dépendances générées de cette manière ne sont jamais ajoutées lors de la première soumission de l'objet.

Exemples

Pour soumettre les travaux `test` dans le flot de travaux `JOBS`, exécutez la commande suivante :

```
sbj test
```

Pour soumettre un travail caractérisé par l'alias `rptx4` et placer le travail dans le flot de travaux `reports` dont l'heure d'exécution **at** est définie sur 17:30, exécutez la commande suivante :

```
sbj rjob4;alias=rptx4;into=reports;at=1730
```

Pour soumettre le travail `txjob3` sur tous les postes de travail dont le nom commence par `site`, exécutez la commande suivante :

```
sbj site@#txjob3;alias
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle décrite dans :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Soumission de travaux prédéfinis".

submit sched

Soumet un flot de travaux pour le traitement.

Pour exécuter cette commande, dans le fichier de sécurité, vous devez disposer d'un accès ***submit*** pour le flot de travaux portant le nom spécifié dans sa définition de base de données et, si vous utilisez le mot clé ***alias***, portant également le nom spécifié avec ce mot clé. Pour inclure des dépendances `needs` et `prompt`, vous devez avoir un accès ***use*** aux ressources et aux invites globales.

La commande `submit schedule` utilise les données d'identification définies dans le fichier `useropts` appartenant à l'utilisateur *utilisateur_TWS* qui a installé ce poste de travail.

Si vous soumettez le flot de travaux à partir d'un poste de travail différent du gestionnaire de domaine maître, vous devez être connecté en tant qu'utilisateur :

- détenant les autorisations d'accès appropriées définies dans le fichier le fichier useropts pour vous connecter au gestionnaire de domaine maître par l'intermédiaire de WebSphere Application Server
- autorisé à exécuter les commandes submit dans le fichier de sécurité stocké sur le gestionnaire de domaine maître

Syntaxe

{**submit**
sched | **sbs**} [*poste de travail*#]*nom_flot_travaux*
[**;alias**[**=***nom*]]
[**;***option_flot_travaux*[**;**...]]
[**;vartable=***nom_table*]
[**;noask**]

Arguments

poste de travail
Indique le nom du poste de travail sur lequel le flot de travaux va être lancé. Les caractères génériques sont admis, auquel cas, le flot de travaux est lancé sur les postes de travail éligibles. Par défaut, le poste de travail sur lequel **conman** est en cours d'exécution est utilisé. Vous ne pouvez pas indiquer une classe de postes de travail ou un domaine.

nom_flot_travaux
Indique le nom du flot de travaux. Les caractères génériques sont admis, auquel cas, tous les flots de travaux éligibles sont soumis. Si le flot de travaux figure déjà dans le plan de production, vous devez utiliser l'argument **alias** pour attribuer un nom unique.

alias=*nom*
Indique un nom unique à attribuer au flot de travaux à la place de *nom_flot_travaux*. Si une valeur lui est attribuée, elle correspond également à l'*ID_flot_travaux*. Si le mot clé **alias** est entré sans spécifier de nom, un nom est créé à partir des cinq premiers caractères alphanumériques du *nom_flot_travaux*, suivis par un nombre aléatoire à dix chiffres. Le nom apparaît systématiquement en majuscules. Par exemple, si *nom_flot_travaux* est défini sur `sttrom`, le nom généré sera similaire à `STTRO1234567890`.

L'autorisation de soumettre le calendrier est vérifiée dans le fichier de sécurité à l'aide du nom d'origine et non du nom d'alias.

option_flot_travaux
Entrez l'un des éléments suivants (voir «Détail des mots clés pour la définition des flots de travaux», à la page 241 pour rechercher les options mutuellement exclusives) :

[at=*hhmm* **[timezone|tz** *fuseau_horaire***] [+***n* **days** | *date*] **[absolute** | **abs]]** | **[schedtime=[***hhmm* **[***date***]** | **[+***n* **days]]**
où :

at indique à quel moment le flot de travaux peut être lancé. Si le mot-clé at est utilisé, le flot de travaux ne peut pas démarrer avant l'heure définie par ce mot-clé (voir la rubrique relative aux mots-clés de définition du flot de travaux dans le chapitre "Définition d'objets dans la base de données" dans le manuel "*IBM Tivoli Workload Scheduler - Guide d'utilisation et de référence*" pour plus d'informations sur le mot-clé "at"). Notez que si le gestionnaire de domaine maître de votre réseau s'exécute lorsque la

valeur yes est attribuée aux options enLegacyStartOfDayEvaluation et enTimeZone afin de convertir l'heure startOfDay du gestionnaire de domaine maître en fuseau horaire local sur chaque poste de travail du réseau, vous devez ajouter le mot clé **absolute** pour le faire fonctionner.

schedtime représente le jour et l'heure de positionnement du flot de travaux dans le plan. Le flot de travaux est lancé si, à cet instant il ne comporte aucune dépendance et qu'aucune restriction horaire **at** n'est définie à son égard. La valeur affectée à **schedtime** ne représente pas une dépendance pour le flot de travaux. Sa valeur s'affiche alors dans les colonnes *Heure_calend* de la sortie des commandes d'affichage. Si une restriction **at** est définie, la valeur affectée à **schedtime** est remplacée par la valeur **at**. Lorsque le flot de travaux démarre effectivement, la valeur affectée à **schedtime** est remplacée par l'heure de démarrage réelle du flot de travaux.

Le format utilisé pour *date* dépend de la valeur affectée à la variable ***format_date*** spécifiée dans le fichier localopts.

Si aucun fuseau horaire supplémentaire n'est spécifié, celui défini sur le poste de travail exécutant la commande est pris en compte.

carryforward

deadline=*heure***[timezone | tz** *fuseau_horaire***][+***n* **days | ***date***]**
Si aucun fuseau horaire supplémentaire n'est spécifié, celui défini sur le poste de travail exécutant la commande est pris en compte.

follows=[*netagent***::][***workstation***#]{***jobstreamname***[***hhmm* **[***mm/jj[/aa]***]][.***travail* **| @] | ***ID_flot_travaux***.***travail***;schedid}| ***travail***[;nocheck][;wait=***heure***][,...]**
Les critères de correspondance utilisés lors de la soumission de flots de travaux en production sont différents de ceux qui sont utilisés pour résoudre les dépendances de **prédécesseur/successeur** dans le plan de préproduction. Lorsqu'un flot de travaux (JS_A, par exemple) contenant une dépendance de **prédécesseur/successeur** provenant d'un travail ou d'un flot de travaux (JS_B, par exemple) est soumis à partir du programme de ligne de commande **conman**, l'instance de prédécesseur JS_B est définie en fonction des critères suivants :

1. L'instance la plus proche de JS_B précédant JS_A.
2. Si aucune instance remplacée de JS_B n'existe, l'instance de prédécesseur est l'instance la plus proche de JS_B suivant JS_A.
3. Sinon, une erreur s'affiche et la commande échoue si le mot clé **;nocheck** n'est pas utilisé.

L'instance de flot de travaux prédécesseur est recherchée parmi les instances ajoutées au plan de production lorsque **JnextPlan** a été exécuté et les instances soumises en production à l'aide de la commande **sbs**, y compris celles qui ont été soumises avec un alias.

Avertissement : L'argument **;nocheck** n'est pas pris en charge dans les dépendances interréseaux.

limit=*nombre_max_travaux*

needs=[*num***] [***poste_de_travail***#]***ressource***[,...]**

opens=[*poste_de_travail***#]"***file_name***"[(***qualifiant***)][,...]**

priority=[*pri* **| hi | go]**

prompt="[: | !]*texte***" | ***nom_invite***[,...]**

until *heure* **[timezone|tz** *fuseau_horaire***][+***n* **day[s] | [absolute | abs]] [;onuntil** *action***]**

Si aucun fuseau horaire supplémentaire n'est spécifié, celui défini sur le poste de travail exécutant la commande est pris en compte.

vartable=*nom_table*
Indique le nom de la table de variables, s'il est différent du nom par défaut, dans laquelle sont définies les variables que vous souhaitez utiliser.

A faire :

- Avec cette commande, vous pouvez utiliser une substitution de variable pour les mots clés suivants :
  - `opens`
  - `invite`
- Placez la variable entre accents circonflexes (^), puis l'ensemble de la chaîne entre guillemets. Si la variable contient une partie d'un chemin, vérifiez que les accents circonflexes ne sont pas immédiatement précédés d'une barre oblique inversée (\). En effet, la séquence \^ pourrait être interprétée comme une séquence d'échappement et serait alors résolue par le programme d'analyse syntaxique comme un accent circonflexe. Si nécessaire, placez la barre oblique inversée dans la définition de la variable entre accents circonflexes.

Si vous soumettez un flot de travaux contenant des variables définies dans une table de variables qui n'est pas la table de variables par défaut et que vous n'indiquez pas cette table dans le cycle d'exécution, dans le flot de travaux ou sur le poste de travail, les variables ne sont pas résolues. Pour plus d'informations, voir Chapitre 6, «Personnalisation de votre charge de travail à l'aide des tables de variables», à la page 121.

noask Indique que l'utilisateur ne doit pas être invité à confirmer avant d'effectuer une opération sur chaque flot de travaux éligible.

Commentaires

Les flots de travaux soumis en production à partir de la ligne de commande **conman** ne sont pas inclus dans le plan de préproduction et ne peuvent donc pas être pris en compte lors de l'identification des prédécesseurs des dépendances de prédécesseur/successeur externes.

Si aucun poste de travail n'est indiqué avec les options `follows`, `needs` ou `opens`, le poste de travail sur lequel se trouve le flot de travaux est utilisé par défaut.

L'agenda classe les dépendances de prédécesseur/successeur comme *internes* lorsqu'elles ne sont spécifiées que par leur nom de travail dans le flot de travaux. Il les classe en temps que dépendances *externes* lorsqu'elles sont spécifiées dans le format *jobStreamName.workstationName.jobName*.

Lorsque vous soumettez un flot de travaux qui inclut un travail possédant une dépendance de prédécesseur/successeur qui porte le même nom de flot de travaux (par exemple, le flot de travaux `schedA` inclut un travail nommé `job6` qui possède une dépendance de prédécesseur/successeur sur `schedA.job2`), la dépendance est ajoutée en tant que dépendance de prédécesseur/successeur *externe*. Depuis la version 8.3, contrairement aux versions précédentes, étant donné que l'agenda

utilise le critère de correspondance `sameday` pour résoudre des dépendances externes, les dépendances générées de cette manière ne sont jamais ajoutées lors de la première soumission de l'objet.

Exemples

Pour soumettre le flot de travaux `adhoc` sur le poste de travail `site1` et le définir en tant que flot de travaux **carryforward**, exécutez la commande suivante :

```
submit sched=site1#adhoc;carryforward
```

Pour soumettre le flot de travaux `fox4` caractérisé par un nombre maximal de **2** travaux, une priorité égale à **23** et une échéance fixée à minuit par l'option **until**, exécutez la commande suivante :

```
sbs fox4;limit=2;pri=23;until=0000
```

Pour soumettre le flot de travaux `sched3` sur tous les postes de travail dont le nom commence par `site`, exécutez la commande suivante :

```
sbs site@#sched3
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle décrite dans :

le manuel Dynamic Workload Console - Guide d'utilisation, section "Soumission de flots de travaux prédéfinis".

switcheventprocessor

permute le serveur de traitement d'événement du gestionnaire de domaine maître au maître de sauvegarde ou inversement.

Notez que vous pouvez exécuter le serveur de traitement d'événement également sur un poste de travail installé en tant que maître de sauvegarde qui s'exécute en tant que simple agent tolérant aux pannes.

Syntaxe

{**switcheventprocessor** | **switchevtp**} *poste de travail*

Arguments

poste de travail
Indique le nom du gestionnaire de domaine maître ou du maître de sauvegarde sur lequel vous souhaitez faire passer le serveur de traitement d'événement. Les caractères génériques ne sont pas admis.

Commentaires

Si vous émettez la commande à partir d'un poste de travail différent de celui sur lequel le processus d'événements est configuré, la commande utilise le client de ligne de commande, de manière à définir correctement les données d'identification de l'utilisateur du client de ligne de commande.

Dans le cas des maîtres de sauvegarde, le poste de travail doit avoir l'attribut `full-status` défini sur `on`.

Le droit de `démarrer` et d'`arrêter` des actions sur des objets `cpu` doit être activé dans le fichier de sécurité pour pouvoir exécuter cette commande.

L'état de corrélation des instances de règle de corrélation en cours est perdu lorsque le serveur est désactivé ou migré. Si la mise en cache des événements reçus est activée dans le fichier de configuration de l'écouteur EIF, les événements mis en cache sont perdus après permutation du traitement d'événement.

Important :

- Avant d'exécuter cette commande, prenez la précaution de lancer **`planman deploy`**. Pour ce faire, prenez soin de déployer les dernières modifications ou les derniers ajouts apportés aux règles d'événement actives avant de basculer le processeur d'événement. Il s'agit d'éviter que l'ancien processeur d'événement ne reçoive les dernière mises à jours (envoyées automatiquement en fonction de la configuration de l'option globale **`deploymentFrequency`**) à la place du nouveau processeur suite à une non-concordance horaire.
- Les horloges du maître et des maîtres de sauvegarde conçus pour exécuter le processeur d'événement doivent être synchronisées en permanence afin d'éviter toutes les incohérences lors du calcul de l'intervalle de temps des règles d'événement en cours d'exécution. En effet, si le processeur d'événement bascule vers un ordinateur non synchronisé, les actions suite au dépassement du délai d'attente du processus de déclenchement doivent subir les délais imprévus. Utilisez un serveur NTP (Network Time Protocol) pour maintenir toutes les horloges synchronisées.

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle qui suit :

1. A partir de la barre de navigation, cliquez sur **Statut et état de santé du système** > **Surveillance de l'environnement** > **Surveillance des postes de travail**.
2. Sélectionnez une tâche pour surveiller les postes de travail.
3. Choisissez un nom pour le moteur ou spécifiez les propriété de connexion, puis cliquez sur **OK**.
4. A partir de la table contenant la liste des postes de travail, sélectionnez un poste de travail et cliquez sur **Plus d'actions** > **Devenir processeur d'événement**.

switchmgr

Remplace le gestionnaire de domaine en cours par un gestionnaire de domaine de sauvegarde.

Vous devez avoir un accès ***start*** et ***stop*** au gestionnaire de domaine de sauvegarde.

La commande **switchmgr** doit uniquement être utilisée dans le cadre d'une procédure spécifique destinée à permuter de façon définitive ou temporaire les capacités de gestion de domaine d'un gestionnaire à son gestionnaire de domaine de sauvegarde. Pour plus d'informations sur ces procédures, voir *Tivoli Workload Scheduler - Guide d'administration*.

Syntaxe

{**switchmgr** | **switchm**} *domaine***;***nouv_gest*

Arguments

domaine
Indique le domaine dans lequel vous souhaitez effectuer le changement de gestionnaire.

nouv_gest
Indique le nom du nouveau gestionnaire de domaine. Il doit correspondre à un poste de travail du même domaine et être défini par avance comme agent tolérant aux pannes, les options Résolution des dépendances et Statut intégral étant activées.

Commentaires

Cette commande arrête le poste de travail indiqué et le relance ensuite en tant que gestionnaire de domaine. Tous les postes de travail faisant partie du domaine sont informés du changement et l'ancien gestionnaire de domaine est converti en agent tolérant aux pannes dans le domaine.

Lors de la prochaine exécution de `JnextPlan` sur l'ancien gestionnaire de domaine, le domaine agit comme si une autre commande **switchmgr** avait été exécutée et l'ancien gestionnaire de domaine reprend automatiquement ses responsabilités de gestionnaire de domaine.

Les agents tolérants aux pannes définis avec `securitylevel = on` peuvent ne pas réussir à utiliser le port SSL pour se connecter au nouveau gestionnaire de domaine maître après l'exécution de la commande **`switchmgr`**. Dans ce cas, pour laisser l'agent démarrer correctement, effectuez l'une des deux opérations suivantes :

- Interrompez puis rétablissez la liaison à l'agent à partir du nouveau gestionnaire de domaine maître.
- Utilisez l'option `securitylevel = force` sur l'agent.

Exemples

Pour basculer le gestionnaire de domaine sur le poste de travail `orca` dans le domaine `masterdm`, exécutez la commande suivante :

```
switchmgr masterdm;orca
```

Pour basculer le gestionnaire de domaine sur le poste de travail `ruby` dans le domaine `bldg2`, exécutez la commande suivante :

```
switchmgr bldg2;ruby
```

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle qui suit :

1. A partir de la barre de navigation, cliquez sur **Statut et état de santé du système** > **Surveillance de l'environnement** > **Surveillance des postes de travail**.
2. Sélectionnez une tâche pour surveiller les postes de travail.
3. Choisissez un nom pour le moteur ou spécifiez les propriété de connexion, puis cliquez sur **OK**.

4. A partir de la table contenant la liste des postes de travail, sélectionnez un poste de travail et cliquez sur **Plus d'actions** > **Devenir gestionnaire de domaine maître**.

Commande système

Exécute une commande système.

Syntaxe

[**:** | **!**] *commande système*

Arguments

commande système
Désigne une commande système valide. Le préfixe (: ou !) est requis lorsqu'un nom de commande est orthographié comme une commande **conman**.

Exemples

Pour exécuter une commande **ps** sous UNIX, exécutez la commande suivante :

```
ps -ef
```

Pour exécuter une commande **dir** sousWindows, exécutez la commande suivante :

```
dir \bin
```

tellop

Envoie un message à la console Tivoli Workload Scheduler.

Syntaxe

{**tellop** | **to**} [*texte*]

Arguments

texte Indique le texte du message. Celui-ci peut comporter jusqu'à 900 caractères.

Commentaires

Si la commande **tellop** est émise sur le gestionnaire de domaine maître, le message est envoyé à tous les postes de travail connectés. Si elle est émise sur un gestionnaire de domaine, le message est envoyé à tous les agents connectés de son domaine et des domaines subordonnés. Si elle est émise sur un poste de travail autre qu'un gestionnaire de domaine, le message est envoyé uniquement à son gestionnaire de domaine si celui-ci est connecté. Le message s'affiche uniquement si le niveau de message écran est supérieur à zéro. Voir «console», à la page 410.

Si la commande **tellop** est entrée seule, vous êtes invité à entrer le texte du message. A l'invite, tapez chaque ligne et appuyez sur la touche Retour. A la fin du message, tapez deux barres obliques (//) ou un point (.)et appuyez sur la touche Retour. Vous pouvez utiliser la séquence de retour ligne (\n) pour la mise en forme des messages. Vous pouvez quitter la commande **tellop** à tout moment sans envoyer de message en appuyant sur **Control+c**.

Exemples

Pour envoyer un message, exécutez la commande suivante :

```
tellop TWS sera arrêté à\n4:30 pendant 15 minutes.
```

Pour inviter l'utilisateur à saisir du texte avant d'envoyer un message, exécutez la commande suivante :

```
to
TELLOP>********************************
TELLOP>*  TWS sera arrêté à            *
TELLOP>*  4:30 pendant 15 minutes.     *
TELLOP>********************************
TELLOP>//
```

unlink

Ferme les liaisons de données entre les postes de travail.

Vous devez avoir un accès ***unlink*** au poste de travail cible.

Syntaxe

unlink [*domaine***!**]*poste_de_travail*
[**;noask**]

Arguments

domaine
Indique le nom du domaine dans lequel les liaisons doivent être arrêtées. Il n'est pas nécessaire d'indiquer le nom de domaine d'un poste de travail appartenant au domaine maître. Les caractères génériques sont acceptés.

Remarque : Vous devez toujours indiquer le nom de domaine lors de la suppression d'un lien d'un poste de travail n'appartenant pas au domaine maître.

Cet argument est utile lors de la suppression des liaisons de plusieurs postes de travail dans un domaine. Par exemple, pour déconnecter tous les agents du domaine `stlouis`, utilisez la commande suivante :

```
unlink stlouis!@
```

Si vous ne précisez pas *domaine* et que *poste_de_travail* contient des caractères génériques, le domaine par défaut est celui dans lequel s'exécute **conman**.

poste de travail
Indique le nom du poste de travail dont la liaison doit être supprimée. Les caractères génériques sont acceptés.

Cette commande n'est pas prise en charge sur les postes de travail de moteur distant.

noask Indique que l'utilisateur ne doit pas être invité à confirmer avant d'effectuer une opération sur chaque poste de travail éligible.

Commentaires

En supposant qu'un utilisateur possède un accès **unlink** aux postes de travail dont la liaison est en cours de suppression, les règles suivantes s'appliquent :

- Un utilisateur exécutant **conman** sur le gestionnaire de domaine maître peut supprimer la liaison de n'importe quel poste de travail du réseau.
- Un utilisateur exécutant **conman** sur un autre gestionnaire que le gestionnaire de domaine maître peut supprimer la liaison de n'importe quel poste de travail de son domaine et des domaines subordonnés. L'utilisateur ne peut pas supprimer la liaison des postes de travail dans les domaines homologues.
- Un utilisateur exécutant **conman** sur un agent peut supprimer la liaison de n'importe quel poste de travail dans son domaine local à condition que le poste de travail soit un gestionnaire de domaine ou un hôte. Un agent homologue appartenant au même domaine ne peut pas être privé de liaisons.

Pour des informations supplémentaires, voir «link», à la page 424.

Exemples

La figure 27 et le tableau 71 présentent les liaisons fermées par les commandes **unlink** exécutées par les utilisateurs en différents points du réseau.

DM*n* sont des gestionnaires de domaine et **A***nn* sont des agents.

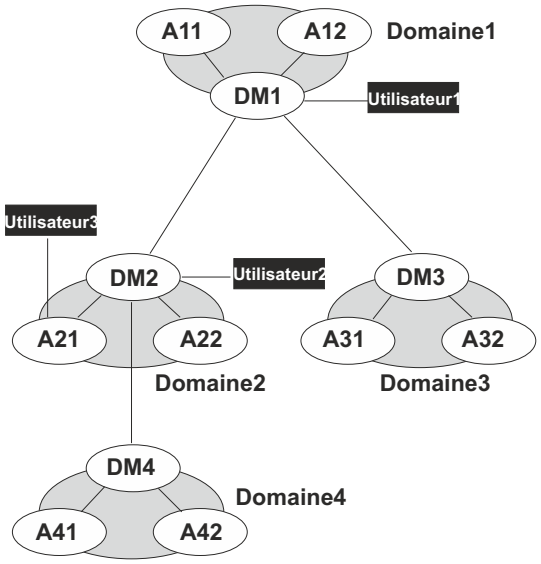


Figure 27. Postes de travail réseau non liés

Tableau 71. Postes de travail non liés

Commande	Fermeture par Utilisateur1	Fermeture par Utilisateur2	Fermeture par Utilisateur3
unlink@!@	Toutes les liaisons sont fermées	DM1-DM2 DM2-A21 DM2-A22 DM2-DM4 DM4-A41 DM4-A42	DM2-A21

Tableau 71. Postes de travail non liés (suite)

Commande	Fermeture par Utilisateur1	Fermeture par Utilisateur2	Fermeture par Utilisateur3
unlink @	DM1-A11 DM1-A12 DM1-DM2 DM1-DM3	DM1-DM2 DM2-A21 DM2-A22 DM2-DM4	DM2-A21
unlink DOMAIN3!@	DM3-A31 DM3-A32	Non autorisé	Non autorisé
unlink DOMAIN4!@	DM4-A41 DM4-A42	DM4-A41 DM4-A42	Non autorisé
unlink DM2	DM1-DM2	Non applicable	DM2-A21
unlink A42	DM4-A42	DM4-A42	Non autorisé
unlink A31	DM3-A31	Non autorisé	Non autorisé

Voir aussi

A partir de Dynamic Workload Console, vous pouvez effectuer la même tâche que celle qui suit :

1. A partir de la barre de navigation, cliquez sur **Statut et état de santé du système** > **Surveillance de l'environnement** > **Surveillance des postes de travail**.
2. Sélectionnez une tâche pour surveiller les postes de travail.
3. Choisissez un nom pour le moteur ou spécifiez les propriété de connexion, puis cliquez sur **OK**.
4. A partir de la table contenant la liste des postes de travail, sélectionnez un poste de travail et cliquez sur **Plus d'actions** > **Supprimer le lien**.

version

affiche la bannière de programme **conman**, avec la version allant jusqu'au niveau du groupe de correctifs installé.

Syntaxe

{**version** | **v**}

Exemples

Pour afficher la bannière du programme **conman**, exécutez la commande suivante :

```
%version
```

La sortie est similaire à ce qui suit :

```
Tivoli Workload Scheduler (UNIX)/CONMAN 9.1.0.00 (20130516)
Licensed Materials  Property of IBM*
5698-WSH
(C) Copyright IBM Corp 1998, 2013 All rights reserved.
* Trademark of International Business Machines
```

```
|                       Installed for user "twsuser".
|                       Locale LANG set to the following: "en"
|                       Scheduled for (Exp) 05/20/13 (#8) on LB001542_MASTER.Batchman LIVES.
|                       Limit:55,Fence:0,Audit Level:0
```

Chapitre 12. Activation des fonctions de planification dynamique dans votre environnement

Cette section explique comment vous pouvez activer des fonctions de planification dynamique dans votre environnement afin de planifier des travaux Tivoli Workload Scheduler existants et des types de travaux avec options avancées ; ceux-ci sont fournis avec le produit et les types supplémentaires sont implémentés à l'aide de plug-ins personnalisés.

Les fonctions dynamiques vous aident à conserver des stratégies métier et à vérifier les contrats de service en :

- reconnaissant automatiquement des ressources d'environnement de planification ;
- faisant correspondre des exigences de travaux avec des ressources disponibles ;
- contrôlant et optimisant l'utilisation de ressources ;
- suivant automatiquement des modifications de ressources ;
- demandant des ressources supplémentaires, le cas échéant.

Vous pouvez activer des fonctions dynamiques dans votre environnement en définissant un ensemble de types de poste de travail :

Agent dynamique
Poste de travail qui gère une grande variété de types de travaux, comme par exemple les travaux d'une base de données ou d'un protocole FTP spécifiques, en plus des types de travaux existants. Ce poste de travail est automatiquement défini et enregistré dans la base de données Tivoli Workload Scheduler lorsque vous installez le agent dynamique. Vous pouvez regrouper les agents dynamiques dans des pools et des pools dynamiques.

Dans un réseau simple, les agents dynamiques se connectent directement au gestionnaire de domaine maître ou via un gestionnaire de domaine dynamique. Dans des topologies de réseau plus complexes, où le gestionnaire de domaine maître ou le gestionnaire de domaine dynamique ne peut pas communiquer directement avec l'agent dynamique, vous pouvez configurer vos agents dynamiques pour utiliser une passerelle locale ou distante.

Pool Un poste de travail regroupant un ensemble d'agents dynamiques doté de caractéristiques matérielles ou logicielles similaires auxquelles sont soumis les travaux. Tivoli Workload Scheduler équilibre les travaux entre les agents dynamiques du pool et réaffecte automatiquement les travaux aux agents dynamiques disponibles lorsqu'un agent dynamique n'est plus disponible. Pour créer un pool d'agents dynamiques dans votre environnement Tivoli Workload Scheduler, définissez un poste de travail de type **pool** hébergé par le poste de travail Workload Broker, puis sélectionnez les agents dynamiques que vous souhaitez ajouter au pool. Vous pouvez définir le pool à l'aide de Dynamic Workload Console ou de la commande **composer**.

Pool dynamique
Poste de travail qui regroupe un ensemble de agents dynamiques, défini de manière dynamique en fonction des exigences de ressource que vous indiquez et hébergé par le poste de travail Workload Broker. Par exemple,

si vous avez besoin d'un poste de travail sollicitant peu l'unité centrale et que Windows est installé pour exécuter votre travail, vous spécifiez les exigences suivantes à l'aide de Dynamic Workload Console ou de la commande **composer**. Lorsque vous sauvegardez l'ensemble d'exigences, un nouveau poste de travail est automatiquement créé dans la base de données Tivoli Workload Scheduler. Ce poste de travail est hébergé par le poste de travail Workload Broker. Ce poste de travail mappe tous les agents dynamiques de votre environnement qui répondent aux exigences spécifiées. Le pool généré est mis à jour dynamiquement dès qu'un nouvel agent dynamique approprié devient disponible. Les travaux s'exécutent sur le premier poste de travail du pool dynamique qui répond à toutes les exigences. Les travaux planifiés sur ce poste de travail héritent automatiquement des exigences définies pour le poste de travail.

Pour plus d'informations sur la création de pools et de pools dynamiques à l'aide de l'Dynamic Workload Console, voir

la section consacrée à la création d'un pool d'agents dans *Tivoli Dynamic Workload Console - Guide d'utilisation*. Pour plus d'informations sur la création de pools et de pools dynamiques à l'aide de la commande **composer**, voir *Guide d'utilisation et de référence*, *SC11-2009*.

Les pools agents dynamiques et les pools dynamiques optimisent la fonction dynamique intégrée à Tivoli Workload Scheduler et donnent la possibilité d'associer de façon dynamique votre charge de travail soumise (ou une partie) aux meilleures ressources disponibles lors de l'exécution. Vous pouvez activer des fonctions de planification dynamique sur les postes de travail lors de l'installation. Pour plus d'informations sur l'installation de agents dynamiques, voir la section relative à l'installation d'un nouvel agent dans le *Guide de planification et d'installation*, *SC11-2008*.

Vous pouvez agents dynamiques, les pools et les pools dynamiques pour planifier les types de travaux avec options avancées. Les types de travaux avec options avancées incluent ceux fournis avec le produit et les types supplémentaires implémentés via les plug-ins personnalisés. Les deux types de travail s'exécutent uniquement sur agents dynamiques, les pools et les pools dynamiques. Pour plus d'informations sur la planification des types de travaux avec options avancées, voir «Création de types de travaux avec options avancées», à la page 518. Pour plus d'informations sur la création de plug-ins personnalisés, voir *Extension de Tivoli Workload Automation*, *SC14-7623*.

Vous pouvez également utiliser agents dynamiques, des pool et des pools et des pools dynamiques, pour exécuter les travaux que vous avez créés pour les types de poste de travail Tivoli Workload Scheduler existants. Pour exécuter ces travaux sur les types de poste de travail dynamique, il suffit de modifier la spécification du poste de travail sur lequel vous souhaitez procéder à l'exécution. Pour plus d'informations sur la planification de travaux Tivoli Workload Scheduler existants, voir «Ajout de fonctions dynamiques aux travaux Tivoli Workload Scheduler existants», à la page 533.

Si vous souhaitez optimiser la fonction dynamique lors de la planification des types de travaux avec options avancées, vous les planifiez sur des pools et des pools dynamiques, ce qui affecte dynamiquement le travail aux meilleures ressources disponibles. Si n'avez besoin de définir que des types de travaux avec options avancées, sans utiliser la fonction de planification dynamique, vous planifiez ces travaux sur un agent dynamique spécifique, sur lequel le travail s'exécute de façon statique.

Avantages des types de travaux avec options avancées

Cette section décrit les avantages obtenus via l'implémentation de types de travaux avec options avancées, ceux fournis avec le produit ou les types supplémentaires implémentés à l'aide de plug-in personnalisés, et leur planification sur des agents dynamiques, des pools et pools dynamiques.

Bien que le travail Tivoli Workload Scheduler standard soit un script ou une commande générique, vous pouvez définir des travaux pour la réalisation de tâches spécifiques, telles que des opérations Java, de base de données, de transfert de fichiers et de service Web, à l'aide des types de travaux avec options avancées disponibles à partir de Dynamic Workload Console ou de la commande Composer. Vous pouvez définir ces types de travaux simplement, sans avoir de connaissances particulières sur les applications exécutant les travaux.

Les types de travaux avec options avancées incluent ceux fournis avec le produit mais également les types supplémentaires implémentés à l'aide de plug-ins personnalisés.

Les types de travaux avec options avancées suivants sont disponibles

Travaux de transfert de fichiers
Permettent de transférer des fichiers depuis et vers un serveur accessible via FTP, SSH ou n'importe quel autre protocole.

JCL Permettent d'exécuter un travail JCL, par référence ou par définition. La spécification d'un travail par référence permet d'indiquer la référence du travail à soumettre, sans avoir à écrire ou à importer l'intégralité du travail dans le JCL. La spécification d'un travail par définition permet d'indiquer la définition JCL à soumettre. Ce type de travail ne s'exécute que sur Tivoli Workload Scheduler distributed - Agent for z/OS.

travaux de services Web
Permettent d'appeler un service Web.

Travaux de base de données
Permettent d'exécuter des requêtes, des instructions SQL et des travaux sur un certain nombre de bases de données, y compris les bases de données personnalisées. Vous pouvez également créer et exécuter des procédures mémorisées sur des bases de données DB2, Oracle et MSSQL.

Travaux exécutables
Permettent d'exécuter un script ou une commande à l'aide d'options avancées, telles que la redirection d'une entrée ou d'une sortie standard vers un fichier.

Travaux Java
Permettent d'exécuter une classe Java.

travaux MSSQL
Permettent d'exécuter un travail Microsoft SQL.

Travaux de la méthode d'accès
Permettent d'étendre les fonctions de planification des travaux deTivoli Workload Scheduler vers d'autres systèmes et applications à l'aide des méthodes d'accès. Ces méthodes communiquent avec le système externe pour lancer le travail et renvoyer son statut. Les méthodes d'accès disponibles sont les suivantes :

- Oracle E-Business Suite

- PeopleSoft
- SAP
- MVS
- Méthodes personnalisées

travaux J2EE
Permettent aux applications Java d'un même réseau d'envoyer et de recevoir des messages depuis et vers une destination JMS.

travaux IBM i
Permettent d'exécuter une commande sur un système IBM i.

Outre Dynamic Workload Console ou la commande **composer**, vous pouvez utiliser les fichiers de configuration associés pour configurer les types de travaux avec options avancées. Pour plus d'informations, voir la section sur la configuration de planification des types de travaux avec options avancées dans le *Guide d'administration*, *SC11-6396*.

Pour plus d'informations sur la procédure de définition des types de travaux avec options avancées, voir les sections sur les étapes préalables de création des types de travaux avec options avancées et sur la création de définitions de travail dans *Tivoli Dynamic Workload Console - Guide d'utilisation*.

Pour plus d'informations sur la création de travaux à l'aide de la commande **composer**, voir la section relative à la définition des travaux dans le *Guide d'utilisation et de référence*, *SC11-2009*.

Vous pouvez également créer des plug-ins personnalisés pour implémenter vos propres types de travaux avec options avancées pour les applications non prises en charge par Tivoli Workload Scheduler. Pour plus d'informations sur la création de plug-ins personnalisés, voir *Extension de Tivoli Workload Automation*, *SC14-7623*.

Les types de travaux avec options avancées, fournis avec le produit ou les types supplémentaires implémentés via les plug-ins personnalisés, ne peuvent être exécutés que sur les agents dynamiques, les pools et les pools dynamiques.

Création de types de travaux avec options avancées

Cette section décrit le mode de création d'un type de travail spécifique à l'aide de types de travaux avec options avancées fourni avec Dynamic Workload Console.

Vous pouvez aisément définir des types de travaux avec options avancées, sans avoir de connaissances particulières sur les applications exécutant les travaux. Vous pouvez définir ensuite ces types de travaux uniquement sur des pools agents dynamiques et des pools dynamiques. La procédure suivante décrit la manière de créer un travail à l'aide de l'Dynamic Workload Console. La procédure utilisée pour créer les types de travaux avec options avancées est similaire, mais chaque type de travail contient des options spécifiques au travail. Pour plus d'informations sur chaque type de travail, voir l'aide en ligne de l'Dynamic Workload Console.

Pour créer un travail à l'aide de l'Dynamic Workload Console, procédez comme suit :

1. A partir de la barre d'outils de navigation, cliquez sur  **Administration** > **Conception de la charge de travail** > **Gestion des définitions de charge de travail**
2. Spécifiez un nom de moteur, distribué ou z/OS. Le concepteur de charge de travail s'ouvre. Les types et les caractéristiques des travaux varient selon que vous sélectionnez un moteur distribué ou z/OS.
3. Dans la sous-fenêtre Liste de travail, cliquez sur **Nouveau > Définition de travail**.
4. Sélectionnez la catégorie et le type de travail que vous souhaitez créer.
5. Dans le panneau des propriétés, spécifiez les attributs de la définition de travail que vous créez. Pour des détails complets sur les zones et les options disponibles, reportez-vous à l'aide en ligne en cliquant sur "**?**" dans l'angle supérieur droit.
6. Cliquez sur **Sauvegarder** pour sauvegarder la définition de travail dans la base de données.

Pour plus d'informations sur les types de travaux et les catégories disponibles, reportez-vous au tableau suivant :

Tableau 72. Types de travaux

Catégorie	Type de travail	Description	Types de travail avec fonctions avancées
Natif	Windows	Travaux exécutés sur les systèmes d'exploitation Windows.	NON
	UNIX	Travaux exécutés sur les plateformes UNIX.	NON
	Autre	Travaux exécutés sur des agents étendus. Pour plus d'informations sur les types de tâches personnalisées pour les applications acquises chez un fournisseur et prises en charge, voir *Tivoli Workload Scheduler for Applications - Guide d'utilisation*. Travaux exécutés sur les agents tolérants aux pannes limités pour IBM i.	NON
	Executable	Travaux qui exécutent un script ou une commande à l'aide d'options avancées, telles que le réacheminement des entrées et sorties standard vers un fichier.	OUI
	Commande à distance	Travaux qui peuvent s'exécuter sur des ordinateurs distants ; aucune installation d'agent requise. **Remarque :** Sous les systèmes z/OS, ils peuvent être créés uniquement à l'aide de Dynamic Workload Console	OUI
	z/OS	Travaux qui exécutent la commande indiquée dans l'onglet JCL sur un système JCL.	OUI
	IBM i	Travaux qui exécutent des commandes sur des systèmes IBM i.	OUI
ERP	Travail SAP sur les postes de travail d'agent étendu	Travaux exécutés sur un agent étendu SAP. Trois types de définitions de travaux SAP R/3 sont concernés : • Travail R/3 standard • Travail de chaîne de processus BW • Travail InfoPackage BW	NON
	Travail SAP sur les postes de travail dynamiques	Travaux exécutés sur des postes de travail d'agent dynamique, sur des pools, sur des pools dynamiques et sur des agents z-centric. Les types suivants de définition de travail SAP sont disponibles : • Travail R/3 standard • Travail de chaîne de processus BW • Travail InfoPackage BW	NON
	Méthode d'accès	Les travaux étendent les fonctions de planification de travaux de Tivoli Workload Scheduler à d'autres systèmes et applications à l'aide de méthodes d'accès. Ces méthodes communiquent avec le système externe pour lancer le travail et renvoyer son statut. Les méthodes d'accès disponibles sont les suivantes : • Oracle E-Business Suite • PeopleSoft • SAP • MVS • Méthodes personnalisées	NON
	Canal PI	Travaux qui exécutent des travaux des canaux SAP Process Integration (PI) pour contrôler les canaux de communication entre l'intégrateur de processus et un système dorsal SAP R/3. **Remarque :** Fourni avec Tivoli Workload Scheduler for Applications 8.6 ou version ultérieure et disponible seulement en cas d'installation spécifique.	OUI

Tableau 72. Types de travaux (suite)

Catégorie	Type de travail	Description	Types de travail avec fonctions avancées
Cloud	Workload Broker	Travaux qui gèrent le cycle de vie d'un travail Dynamic Workload Broker. Pour plus d'informations sur l'utilisation de Dynamic Workload Broker, voir *Tivoli Workload Scheduler - Planification dynamique de la charge de travail*.	NON
	Application des accès	Travaux qui concernent des ordinateurs physiques, des machines virtuelles et des environnements cloud privés et publics qui créent un environnement à la demande. Ce type de travail s'intègre à IBM SmartCloud Provisioning. **Remarque :** Sous les systèmes z/OS, ils peuvent être créés uniquement à l'aide de Dynamic Workload Console	OUI
Transfert de fichier et coordination	Transfert de fichier	Travaux qui exécutent un programme pour transférer les fichiers vers et à partir d'un serveur accessible via FTP, SSH ou d'autres protocoles.	OUI
	Reflet distribué	Travaux exécutés en local et mappage d'autres travaux dans un environnement Tivoli Workload Scheduler distribué.	NON
	Reflet z/OS	Travaux exécutés en local et mappage d'autres travaux dans un environnement Tivoli Workload Scheduler for z/OS distant.	NON
Base de données et intégrations	Base de données	Travaux qui réalisent des requêtes, des instructions SQL et des travaux sur un certain nombre de bases de données, y compris des bases de données personnalisées. Vous pouvez également créer et exécuter des procédures mémorisées sur des bases de données DB2, Oracle et Microsoft SQL Server.	OUI
	Services Web	Travaux qui exécutent un service Web.	OUI
	MS SQL	Travaux qui exécutent un travail Microsoft SQL Server.	OUI
	J2EE	Travaux qui permettent aux applications Java d'un même réseau d'envoyer et de recevoir des messages vers et à partir d'une destination JMS.	OUI
	Java	Travaux qui exécutent une classe Java.	OUI
Analyse métier	Rapports Cognos	Travaux qui exécutent des rapports, des rapports interactifs, une requête et des vues de rapport IBM Cognos. **Remarque :** Fourni avec Tivoli Workload Scheduler for Applications 8.6 ou version ultérieure et disponible seulement en cas d'installation spécifique.	OUI
	InfoSphere DataStage	Travaux qui exécutent des travaux IBM InfoSphere DataStage. **Remarque :** Fourni avec Tivoli Workload Scheduler for Applications 8.6 ou version ultérieure et disponible seulement en cas d'installation spécifique.	OUI
OSLC	OSLC Automation	Travaux qui appellent n'importe quel fournisseur OSLC qui implémente la spécification d'automatisation OSLC. Les ressources d'automatisation définissent des plans d'automatisation, des requêtes d'automatisation et des résultats d'automatisation du cycle de vie de développement, de test et de déploiement du logiciel.	OUI
	OSLCTPM (Provisioning)	Travaux qui appellent n'importe quel fournisseur OSLC qui implémente la spécification d'application des accès OSLC. Les ressources mises à disposition définissent des plans de mise à disposition, des requêtes de mise à disposition et des résultats de mise à disposition du cycle de vie de développement, de test et de déploiement du logiciel.	OUI

Codes retour

La liste suivante répertorie les codes retour pour les types de travaux avec options avancées

```
Travaux de base de données :
CR =  0 -> Le travail s'est terminé avec succès
CR = -1 -> L'instruction SQL a été exécutée avec un code de sortie différent de 1
CR = -2 -> Erreur de travail MSSQL
CR = -3 -> L'instruction SQL n'a pu être exécutée en raison d'une erreur dans l'instruction

Travaux de transfert de fichiers :
RC =  0 -> Le transfert de fichier a réussi.

RC = -1 -> Le transfert de fichier n'est pas effectué. Le travail échoue avec
le code d'erreur suivant : AWKFTE007E

Explication : une erreur s'est produite lors du transfert de fichier.

Causes possibles : fichier distant introuvable ou droits refusés.

RC = -2 -> Le transfert de fichier n'est pas effectué. Le travail échoue avec
le code d'erreur suivant : AWKFTE020E

Explication : uniquement pour les protocoles SSH ou WINDOWS. Une erreur a été
renvoyée pendant la tentative de conversion de la page de codes.

Causes possibles : pour les protocoles SSH ou Windows, la page de codes est
automatiquement détectée et convertie. Dans ce cas, une erreur se trouve dans la
page de code du fichier à transférer et la
page de codes du système local n'est pas respectée.
```

```
RC = -3 -> Le transfert de fichier n'est pas effectué. Le travail échoue avec
le code d'erreur suivant : AWKFTE015E

Explication : une erreur s'est produite lors du transfert de fichier.

Causes possibles : fichier local introuvable.

RC = -4 -> Le transfert de fichier est effectué avec la page de codes par défaut. Le
travail échoue avec le code d'erreur suivant : AWKFTE023E

Explication : la conversion de la page de codes spécifiée n'a pas été effectuée.
Le transfert de fichier a été effectué avec les pages de code par défaut.

Causes possibles : la page de codes spécifiée est indisponible.

Travaux IBM i :
Code retour = code retour utilisateur lors de la récupération
Code retour = 0 -> Le travail s'est terminé avec succès
Code retour > -1 -> travail terminé avec succès

Travaux Java :
CR =  0 -> Le travail s'est terminé avec succès
CR = -1 -> En raison d'une exception rencontrée, l'application Java lancée
par le travail a échoué

Travaux de services Web :
CR =  0 -> Le travail s'est terminé avec succès
CR = -1 -> Le nom d'hôte du serveur contenu dans l'URL du service Web est inconnu
CR = -2 -> Appel d'erreur du service Web
```

Lorsque le code retour utilisateur est récupéré, IBM i l'agent de surveillance lui assigne une priorité.

Définition des variables et mots de passe pour la résolution locale sur des agents dynamiques

Pour les types de travaux avec des options avancées, vous pouvez autoriser la définition et la résolution locales des variables et des mots de passe sur des agents dynamiques (y compris les pools et les pools dynamiques).

Cela est particulièrement utile dans le cas des mots de passe car il n'est pas nécessaire de les indiquer dans la définition du travail. L'avantage réside dans le fait que, si le mot de passe doit être changé, vous ne modifiez pas la définition du travail mais changez le mot de passe à l'aide de la commande param en local sur les agents (ou sur les agents du pool) qui exécutent ou peuvent exécuter le travail. Si le travail doit être soumis sur un pool ou un pool dynamique, vous pouvez copier le fichier avec les définitions de variable sur tous les agents membre de ce pool, de sorte que les variables sont résolues en local quel que soit l'emplacement d'exécution du travail.

Cette fonction n'est pas limitée aux postes de travail Windows. Vous pouvez l'utiliser également sous UNIX, à condition de l'appliquer à des types de travaux avec options avancées.

Pour définir une variable ou un mot de passe en local sur un agent dynamique, utilisez la commande d'utilitaire param. Cette commande à le pouvoir de créer, supprimer et répertorier des variables locales sur des agents dynamiques. Pour en savoir plus sur son utilisation, reportez-vous aux détails de la commande.

Indication de variables et de mots de passe locaux dans les définitions de travail

Après avoir défini une variable et sa valeur avec la commande `param` pour l'ajouter dans une définition de travail de sorte qu'elle est résolue localement sur l'agent lors de l'exécution, utilisez la syntaxe suivante :

`${agent:`*variable_name*`}`

Après avoir défini un mot de passe avec la commande `param` pour l'ajouter dans une définition de travail de sorte qu'il est résolu localement sur l'agent lors de l'exécution, utilisez la syntaxe suivante :

`${agent:password.`*user_name*`}`

Vous pouvez imbriquer des variables dans les mots de passe. Si vous avez utilisé une variable pour définir un utilisateur, entrez le mot de passe comme suit dans la définition du travail :

`${agent:password.${agent:`*variable_name*`}}`

où *variable_name* a été défini précédemment comme ayant la valeur *user_name* avec la commande `param`.

Exemple

Un administrateur Tivoli Workload Scheduler doit définir une tâche de transfert de fichier qui télécharge un fichier depuis un serveur distant vers l'un des agents dynamiques constituant un pool d'agents. L'administrateur souhaite paramétrer dans la définition du travail :

- Le nom avec lequel le fichier distant va être enregistré localement
- L'utilisateur distant et son mot de passe

L'administrateur procède de la manière suivante sur l'un des agents :

1. Il définit une variable nommée `localfile`. La variable est égale à `./npp.5.1.1.Installer.DW.exe` et elle est créée dans un fichier nommé `FTPvars` (aucune section). La commande exécute :

   ```
   E:\IBM\TWA\TWS\CLI\bin>param -c FTPvars..localfile ./npp.5.1.1.Installer.DW.exe
   ```

2. Il définit une variable nommée `remoteUser`. Elle est égale à `FTPuser` et elle est créée dans le fichier `FTPvars` (aucune section). La commande exécute :

   ```
   E:\IBM\TWA\TWS\CLI\bin>param -c FTPvars..remoteUser FTPuser
   ```

3. Il définit le mot de passe pour `FTPuser`. Le mot de passe est égal à `tdwb8nxt` et il est créé dans la section `password` du fichier `FTPvars`. La commande exécute :

   ```
   E:\IBM\TWA\TWS\CLI\bin>param -c FTPvars.password.FTPuser tdwb8nxt
   ```

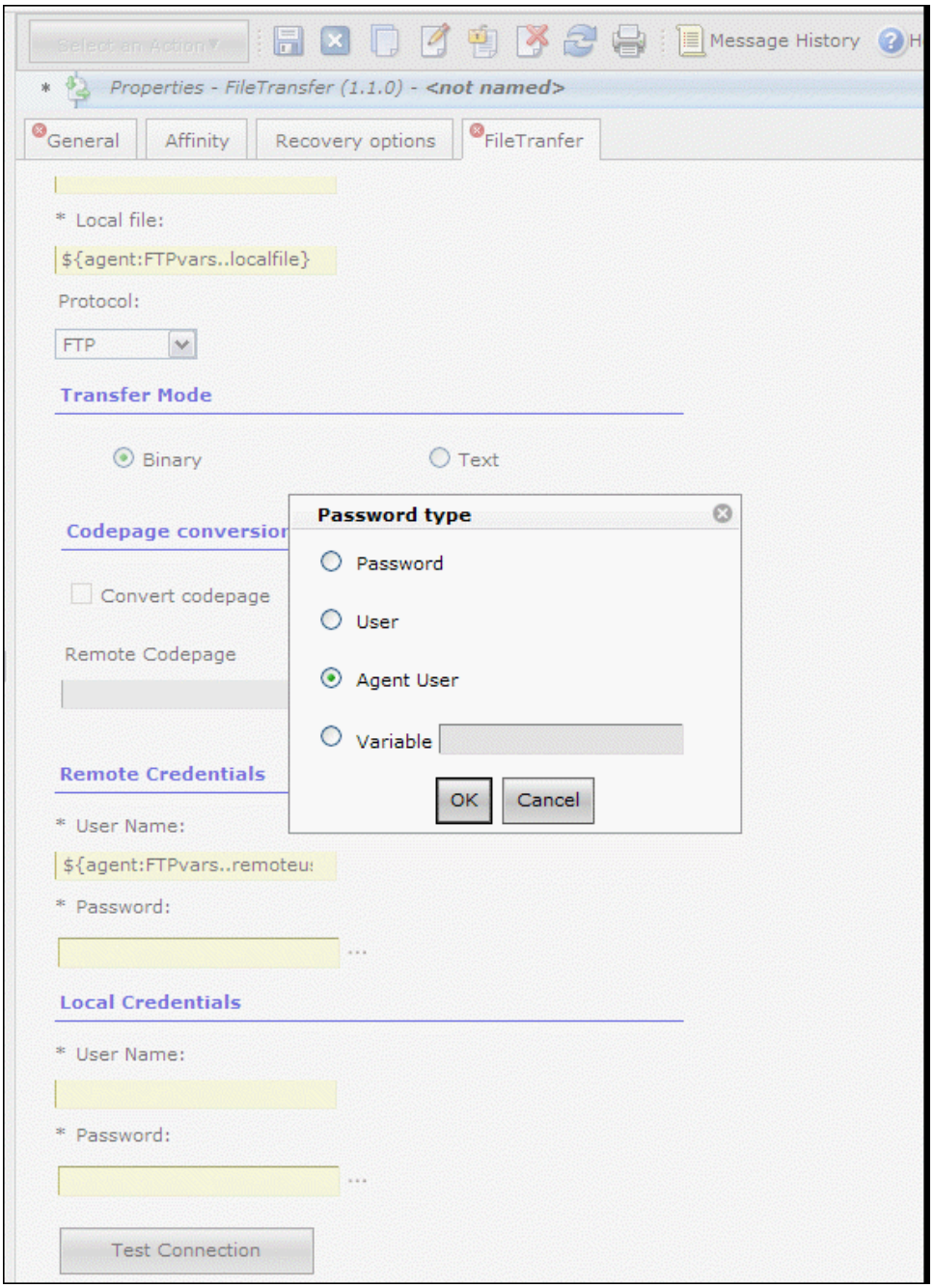
4. Avec un éditeur de texte, il ouvre le fichier `E:\IBM\TWA\TWS\ITA\cpa\config\jm_variables_files\FTPvars` et il vérifie son contenu :

   ```
   localfile = ./npp.5.1.1.Installer.DW.exe
   remoteuser = FTPuser

   [password]
   FTPuser = {aes}XMMYMY2zBHvDEDBo5DdZVmwOJao60pX1K6x2HhRcovA=
   ```

5. Il copie le fichier `FTPvars` dans le chemin d'accès *agent_installation_path*`\TWA\TWS\ITA\cpa\config\jm_variables_files>` de chaque autre agent défini dans le pool.
6. Il commence la définition d'un nouveau travail de transfert de fichier dans la fenêtre Concepteur de charge de travail de Dynamic Workload Console. Dans la fenêtre `FileTransfer` :

a. Il entre ${agent:FTPvars..localfile} dans la zone Fichier local.
b. Il entre ${agent:FTPvars..remoteuser} dans la zone Données d'identification distante →Nom d'utilisateur.
c. Cliquez sur le bouton ... en regard de la zone Données d'identification distante →Mot de passe. La fenêtre Type de mot de passe apparaît et l'administrateur sélectionne Utilisateur d'agent.



d. Après que l'administrateur clique sur le bouton OK pour confirmation dans la fenêtre en incrustation, la zone Données d'identification distante →Mot de passe est renseignée avec la valeur ${agent:password.${agent:FTPvars..remoteuser}}.

7. Il définit toutes les autres zones pour terminer la définition du travail.

Lors de l'exécution du travail, les entités et les mots de passe saisis comme variables sont résolus avec les valeurs définies dans le fichier `FTPvars`.

Utilisation de variables dans les travaux Dynamic Workload Broker

Cette section explique comment ajouter des variables aux travaux que vous envisagez d'exécuter avec Dynamic Workload Broker.

Vous pouvez inclure des variables dans votre définition de travail. La résolution des variables a lieu au moment de la soumission.

Les variables prises en charge sont les suivantes :

Tableau 73. Variables Tivoli Workload Scheduler prises en charge dans les définitions JSDL.

Variables pouvant être insérées dans la définition de travail Dynamic Workload Broker	Description
tws.host.workstation	Nom du poste de travail hôte.
tws.job.date	Date du travail soumis.
tws.job.fqname	Nom complet du travail (UNISON_JOB).
tws.job.ia	Heure d'arrivée des données du travail.
tws.job.interactive	Le travail est interactif. Une valeur peut être `true` (vraie) ou `false` (fausse). S'applique uniquement aux travaux compatibles avec les versions antérieures.
tws.job.logon	Droits d'accès de l'utilisateur exécutant le travail (LOGIN). S'applique uniquement aux travaux compatibles avec les versions antérieures.
tws.job.name	Nom du travail soumis.
tws.job.num	UNISON_JOBNUM.
tws.job.priority	Priorité du travail soumis.
tws.job.promoted	Le travail est promu. Les valeurs peuvent être `YES` ou `No`. Pour plus d'informations sur la promotion des travaux dynamiques, voir «Promotion des travaux planifiés sur des pools dynamiques», à la page 532.
tws.job.recnum	Numéro d'enregistrement du travail.
tws.job.resourcesForPromoted	Quantité des ressources logiques requises affectées sur un pool dynamique à un travail promu. La valeur est soit `1` si le travail est promu, soit `10` si le travail n'est pas promu. Pour plus d'informations sur la promotion de travaux dynamiques, voir «Promotion des travaux planifiés sur des pools dynamiques», à la page 532.
tws.job.taskstring	Chaîne de tâche du travail soumis. S'applique uniquement aux travaux compatibles avec les versions antérieures.
tws.job.workstation	Nom du poste de travail sur lequel le travail est défini.

Tableau 73. Variables Tivoli Workload Scheduler prises en charge dans les définitions JSDL. (suite)

Variables pouvant être insérées dans la définition de travail Dynamic Workload Broker	Description
tws.jobstream.id	ID du flot de travaux incluant le travail (UNISON_SCHED_ID).
tws.jobstream.name	Nom du flot de travaux incluant le travail (UNISON_SCHED).
tws.jobstream.workstation	Nom du poste de travail sur lequel le flot de travaux contenant le travail est défini.
tws.master.workstation	Nom du gestionnaire de domaine maître (UNISON_MASTER).
tws.plan.date	Date de début du plan de production (UNISON_SCHED_DATE).
tws.plan.date.epoch	Date de début du plan de production, dans le format époque (UNISON_SCHED_EPOCH).
tws.plan.runnumber	Numéro d'exécution du plan de production (UNISON_RUN).

Transmission de variables entre des travaux appartenant à la même instance de flot de travaux

Dans de nombreux scénarios, la sortie ou une propriété du premier travail d'un flot de travaux peut désigner l'entrée correspondant à l'exécution des travaux successifs dans la même instance de flot de travaux. Ce principe vaut également pour le flot de travaux `JOBS`.

Dans ce scénario, les travaux *JobA* et *JobB* appartiennent à la même instance de flot de travaux, et *JobA* est un prédécesseur de *JobB*. *JobA* transmet certaines valeurs de variables à *JobB* lors de l'exécution.

Vous pouvez transmettre les variables suivantes de *JobA* à *JobB* :

- *JobA* exporte certaines propriétés et *JobB* y fait référence en tant que variables dans sa définition dans un format prédéfini. Lors de l'exécution, les variables de *JobB* sont automatiquement résolues. Les propriétés de travail que vous pouvez exporter sont liées au type de travail que vous définissez. Voir «Transmission des propriétés de travail d'un travail à l'autre dans la même instance de flot de travaux», à la page 526.
- *JobA* exporte sa valeur de sortie standard et *JobB* y fait référence en tant que variable. Lors de l'exécution, la variable de *JobB* est automatiquement résolue. Voir «Transmission de la sortie standard de travail d'un travail à l'autre dans la même instance de flot de travaux», à la page 528.
- *Seulement pour les travaux exécutables. JobA* exporte sa valeur de sortie standard et *JobB* y fait référence en tant que valeur d'entrée standard. Voir «Transmission de la sortie standard de travail d'un travail à l'autre en tant qu'entrée standard dans la même instance de flot de travaux», à la page 529.
- *JobA* définit la valeur de certaines variable à l'aide de l'utilitaire **jobprop** sur les systèmes d'exploitation UNIX et de l'utilitaire **jobprop.exe** sur les systèmes d'exploitation Windows installés sur les agents dynamiques. *JobB* fait référence à la valeur de ces variable dans sa définition. Lors de l'exécution, les variables de

JobB sont automatiquement résolues. Cette option n'est valide que pour les travaux natifs et exécutables. Voir «Transmission d'un jeu de variables d'un travail à un autre dans la même instance de flot de travaux à l'aide de l'utilitaire **jobprop**», à la page 530.

Remarque : Le flot de travaux `USERJOBS` créé par les processus Tivoli Workload Scheduler, ne prend pas en charge la transmission de variables entre les travaux de ce flot.

Transmission des propriétés de travail d'un travail à l'autre dans la même instance de flot de travaux

Les propriétés de travail que vous pouvez exporter d'un travail dynamique au suivant dans la même instance de flot de travaux sont liées au type de travail que vous définissez. Pour ajouter une propriété de travail dans une autre définition de travail de sorte qu'elle est résolue localement sur l'agent lors de l'exécution, utilisez la syntaxe suivante :

`${job:<NOM_TRAVAIL>.<nom_propriété>}`

où <*NOM_TRAVAIL*> est la valeur de nom ou d'alias du travail à partir duquel vous exportez les valeurs de propriété, et où <*nom_propriété*> est la propriété en référence. La valeur <*nom_propriété*> est insensible à la casse.

Seuls certains types de travaux peuvent transmettre les valeurs de propriétés au travail successeur. C'est le cas des types de travaux suivants :

Travaux InfoSphere DataStage
Le tableau 74, à la page 527 présente la liste des propriétés qui sont transmissibles d'un travail InfoSphere DataStage à un autre ; il indique également le mappage entre les propriétés `Informations supplémentaires` du travail et les propriétés utilisables. Pour plus d'informations sur les travaux InfoSphere DataStage, voir *IBM Tivoli Workload Scheduler for Applications - Guide d'utilisation*.

Travaux reflet
Le tableau 75, à la page 528 présente la liste des propriétés qui sont transmissibles d'un travail reflet à un autre ; il indique également le mappage entre les propriétés `Informations supplémentaires` du travail et les propriétés utilisables.

Travaux OSLC
Le tableau 76, à la page 528 présente la liste des propriétés qui sont transmissibles d'un travail OSLC à un autre ; il indique également le mappage entre les propriétés `Informations supplémentaires` du travail et les propriétés utilisables.

Exemple

L'exemple ci-dessous décrit comment la spécification de variables dans différents formats permet aux variables d'avoir plusieurs valeurs, étant donné qu'elles sont résolues à différents moments. Il explique également comment les variables peuvent être transmises d'un travail à un autre dans une instance de flot de travaux. Le flot de travaux `WIN92MAS_REW#VP_JS_141800058` contient les travaux `JOBA` et `JOBB`. Le travail exécutable `JOBB` fait référence aux propriétés suivantes du travail reflet `JOBA` :

- `ScheduledTime`
- `dJobNAme`
- `dJobStreamName`

- dJobStreamWorkstation

Définitions de la base de données :

```
SCHEDULE WIN92MAS_REW#VP_JS_141800058
:
WIN92MAS_REW#JOBA
 TASK
    <?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:dshadow=
"http://www.ibm.com/xmlns/prod/scheduling/1.0/dshadow" xmlns:
 jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl">
  <jsdl:application name="distributedShadowJob">
    <dshadow:DistributedShadowJob>
      <dshadow:JobStream>VPJS_141800058</dshadow:JobStream>
      <dshadow:Workstation>nc125133</dshadow:Workstation>
      <dshadow:Job>VP_JOBMON_141800058</dshadow:Job>
      <dshadow:matching>
        <dshadow:previous/>
      </dshadow:matching>
    </dshadow:DistributedShadowJob>
  </jsdl:application>
</jsdl:jobDefinition>
 DESCRIPTION "Sample Job Definition for DISTRIBUTED environment"
 RECOVERY STOP
NC125133#JOBB
 TASK
    <?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:XMLSchema=
"http://www.w3.org/2001/XMLSchema" xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
jsdle="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdle" XMLSchema:text="resolveVariableTable" r
  <jsdl:application name="executable">
    <jsdle:executable>
            <jsdle:script>
echo ScheduledTime:${job:JOBA.ScheduledTime}
echo JobName:${job:JOBA.dJobName}
echo JobStreamName:${job:JOBA.dJobStreamName}
echo JobStreamWorkstation:${job:JOBA.dJobStreamWorkstation}
</jsdle:script>
        </jsdle:executable>
  </jsdl:application>
</jsdl:jobDefinition>
 DESCRIPTION "Added by composer."
 RECOVERY STOP
 FOLLOWS JOBA

END
```

Tivoli Workload Scheduler utilise les propriétés des travaux InfoSphere DataStage en l'état. Les valeurs des propriétés Informations supplémentaires des travaux InfoSphere DataStage sont liées à l'environnement local du poste de travail sur lequel InfoSphere DataStage est installé. Le tableau 74 présente les valeurs des propriétés Informations supplémentaires des travaux InfoSphere DataStage sur un poste de travail dont l'environnement local est en_US.

Tableau 74. Propriétés des travaux InfoSphere DataStage

Propriétés des travaux InfoSphere DataStage pouvant être transmises dans une autre définition de travail	**Propriétés Informations supplémentaires des travaux InfoSphere DataStage**
${job:<*NOM_TRAVAIL*>.Interim Status}	Interim Status
${job:<*NOM_TRAVAIL*>.Invocation ID}	Invocation ID
${job:<*NOM_TRAVAIL*>.Invocation List}	Invocation List

Tableau 74. Propriétés des travaux InfoSphere DataStage (suite)

Propriétés des travaux InfoSphere DataStage pouvant être transmises dans une autre définition de travail	Propriétés Informations supplémentaires des travaux InfoSphere DataStage
${job:<*NOM_TRAVAIL*>.Job Control}	Job Control
${job:<*NOM_TRAVAIL*>.Job Controller}	Job Controller
${job:<*NOM_TRAVAIL*>.Job Process ID}	Job Process ID
${job:<*NOM_TRAVAIL*>.Job Restartable}	Job Restartable
${job:<*NOM_TRAVAIL*>.Job Start Time}	Job Start Time
${job:<*NOM_TRAVAIL*>.Job Status}	Job Status
${job:<*NOM_TRAVAIL*>.Job Wave Number}	Job Wave Number
${job:<*NOM_TRAVAIL*>.Last Run Time}	Last Run Time
${job:<*NOM_TRAVAIL*>.User Status}	User Status

Tableau 75. Propriétés des travaux reflet

Propriétés des travaux reflet pouvant être transmises dans une autre définition de travail	Propriétés Informations supplémentaires des travaux reflet
${job:<*NOM_TRAVAIL*>.ScheduledTime}	Heure planifiée du travail distant
${job:<*NOM_TRAVAIL*>.dJobName}	Travail distant
${job:<*NOM_TRAVAIL*>.dJobStreamName}	Flot de travaux distant
${job:<*NOM_TRAVAIL*>.dJobStreamWorkstation}	Poste de travail du flot de travaux distant

Tableau 76. Propriétés des travaux OSLC

Propriétés des travaux OSLC pouvant être transférées à un travail	Propriétés Informations supplémentaires des travaux OSLC
${job:<*NOM_TRAVAIL*>.RESULT_URI}	RESULT_URI

Transmission de la sortie standard de travail d'un travail à l'autre dans la même instance de flot de travaux

Vous pouvez exporter la sortie standard de travail d'un travail dynamique à son successeur dans la même instance de flot de travaux. La variable de sortie standard du travail est utilisée dans la zone `script` de la définition du travail

Pour ajouter une sortie standard du travail dans une autre définition de travail, de sorte qu'elle soit résolue localement sur l'agent lors de l'exécution, utilisez la syntaxe suivante :

${job:<*NOM_TRAVAIL*>.stdlist}

où <*NOM_TRAVAIL*> est le nom ou l'alias du travail à partir duquel vous exportez la sortie standard du travail.

Exemple

Dans cet exemple, le flot de travaux `WIN92MAS_REW#VP_JS_141800058` contient `JOBA_ALIAS`, qui est l'alias de `JOBA`, et les travaux `JOBB`. Le travail exécutable `JOBB` fait référence à la sortie standard `JOBA_ALIAS`.

Définitions de la base de données :

```
SCHEDULE WIN92MAS_REW#VP_JS_141800058
:
WIN92MAS_REW#JOBA as JOBA_ALIAS
 TASK
    <?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:dshadow=
"http://www.ibm.com/xmlns/prod/scheduling/1.0/dshadow" xmlns:
 jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl">
  <jsdl:application name="distributedShadowJob">
    <dshadow:DistributedShadowJob>
      <dshadow:JobStream>VPJS_141800058</dshadow:JobStream>
      <dshadow:Workstation>nc125133</dshadow:Workstation>
      <dshadow:Job>VP_JOBMON_141800058</dshadow:Job>
      <dshadow:matching>
        <dshadow:previous/>
      </dshadow:matching>
    </dshadow:DistributedShadowJob>
  </jsdl:application>
</jsdl:jobDefinition>
 DESCRIPTION "Sample Job Definition for DISTRIBUTED environment"
 RECOVERY STOP

NC125133#JOBB
 TASK
    <?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:XMLSchema="http://www.w3.org/2001/XMLSchema"
xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl" xmlns:
jsdle="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdle"
XMLSchema:text="resolveVariableTable" name="executable">
  <jsdl:application name="executable">
    <jsdle:executable>
    <jsdle:script>echo &quot;stdlist: ${job:JOBA_ALIAS.stdlist}&quot;
</jsdle:script>
    </jsdle:executable>
  </jsdl:application>
</jsdl:jobDefinition>
 DESCRIPTION "Added by composer."
 RECOVERY STOP
 FOLLOWS JOBA_ALIAS
END
```

Transmission de la sortie standard de travail d'un travail à l'autre en tant qu'entrée standard dans la même instance de flot de travaux

Vous pouvez exporter la sortie standard de travail d'un travail dynamique à un travail exécutable successeur en tant qu'entrée standard dans la même instance de flot de travaux. La variable de sortie standard du travail est utilisée dans la zone `input` de la définition du travail exécutable

Pour ajouter une sortie standard du travail dans une autre définition de travail exécutable, de sorte qu'elle soit résolue localement sur l'agent lors de l'exécution, utilisez la syntaxe suivante :

```
${job:<NOM_TRAVAIL>.stduri}
```

où *<NOM_TRAVAIL>* est la valeur de nom ou d'alias du travail à partir duquel vous exportez la sortie standard du travail.

Remarque : Le transfert de la variable *stduri* n'est pas pris en charge pour les travaux reflet.

Exemple

Dans cet exemple, le flot de travaux NC112019#JS_PROP contient JOBALIAS_A, qui est l'alias de NC112019#JOBA, et les travaux NC112019#JOBB. Le travail exécutable NC112019#JOBB fait référence à la sortie standard JOBALIAS_A comme l'entrée standard.

Définitions de la base de données :

```
SCHEDULE NC112019#JS_PROP
:
NC112019#JOBA AS JOBALIAS_A
 TASK
    <?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl" xmlns:
jsdle="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdle" name="executable">
  <jsdl:application name="executable">
    <jsdle:executable interactive="false" path="ls"/>
  </jsdl:application>
</jsdl:jobDefinition>
 DESCRIPTION "Ajouté par Composer pour le flot de travaux : NC112019#JS_PROP."
 RECOVERY STOP

NC112019#JOBB
 TASK
    <?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:XMLSchema="http://www.w3.org/2001/XMLSchema"
xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl" xmlns:
jsdle="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdle" XMLSchema:text="resolveVariableTable" name
  <jsdl:application name="executable">
    <jsdle:executable input="${job:JOBALIAS_A.stduri}" interactive="false" path="cat"/>
  </jsdl:application>
</jsdl:jobDefinition>
 DESCRIPTION "Ajouté par Composer pour le flot de travaux : WIN92MAS#JS_PROP."
 RECOVERY STOP
 FOLLOWS JOBALIAS_A
END
```

Transmission d'un jeu de variables d'un travail à un autre dans la même instance de flot de travaux à l'aide de l'utilitaire jobprop

Vous pouvez utiliser l'utilitaire **jobprop** installé sur agents dynamiques pour définir la variable et sa valeur dans un travail et transmettre la variable au travail successif dans la même instance de flot de travaux.

Pour définir la variable et sa valeur dans le premier travail, utilisez la syntaxe suivante :

```
jobprop <NOM_VAR> <valeur>
```

où <*NOM_VAR*> désigne la variable que vous pouvez exporter dans un autre travail et <valeur> désigne la valeur affectée à <*NOM_VAR*>. Pour plus d'informations sur l'utilitaire **jobprop**, voir «jobprop», à la page 588.

Pour définir les variables dans un autre travail, utilisez la syntaxe suivante :

```
${job:<NOM_TRAVAIL>.<NOM-VAR>}
```

où <*NOM_TRAVAIL*> est la valeur de nom ou d'alias du travail à partir duquel vous exportez la valeur de variable <*NOM-VAR*>.

Exemple

Dans cet exemple, le flot de travaux WIN92MAS#JS_PROP contient les travaux exécutables NC125133#JOBA et NC125133#JOBB. Le travail NC125133#JOBB fait référence aux valeurs de variables NC125133#JOBA suivantes définies à l'aide de l'utilitaire **jobprop** :

- Variable *VAR1* définie sur la valeur value1.
- Variable *VAR2* définie sur la valeur value2.
- Variable *VAR3* définie sur la valeur value3.
- Variable *VAR4* définie sur la valeur value4.

Définitions de la base de données :

```
SCHEDULE WIN92MAS#JS_PROP
:
NC125133#JOBA
 TASK
    <?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl" xmlns:
jsdle="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdle">
  <jsdl:application name="executable">
    <jsdle:executable interactive="false">
    <jsdle:script>#!/bin/sh
. /home/ITAuser/TWA/TWS/tws_env.sh
jobprop VAR1 value1
jobprop VAR2 value2
jobprop VAR3 value3
jobprop VAR4 value4
</jsdle:script>
    </jsdle:executable>
  </jsdl:application>
</jsdl:jobDefinition>
 DESCRIPTION "Sample Job Definition"
 RCCONDSUCC "RC>=0"
 RECOVERY STOP

NC125133#JOBB
 TASK
    <?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl" xmlns:
jsdle="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdle">
  <jsdl:application name="executable">
    <jsdle:executable interactive="false">
    <jsdle:script>
echo VAR1=${job:joba.VAR1}
echo VAR2=${job:joba.VAR2}
echo VAR3=${job:joba.VAR3}
echo VAR4=${job:joba.VAR4}
</jsdle:script>
    </jsdle:executable>
  </jsdl:application>
</jsdl:jobDefinition>
 DESCRIPTION "Sample Job Definition"
 RCCONDSUCC "RC>=0"
 RECOVERY STOP
 FOLLOWS JOBA
END
```

Définition de relations d'affinité

Les relations d'affinité déclenchent l'exécution de travaux sur le même poste de travail. Le poste de travail sur lequel le premier travail s'exécute est choisi de façon dynamique par Dynamic Workload Broker, le travail ou les travaux d'affinité s'exécutant sur le même poste de travail. Cette section s'applique aux types de travaux avec options avancées et aux travaux Workload Broker.

Dans Dynamic Workload Broker, vous pouvez définir des relations d'affinité entre deux travaux ou plus lorsque vous souhaitez les exécuter sur le même poste de travail. Lors de la soumission du travail à partir de l'environnement Tivoli Workload Scheduler, vous pouvez définir l'affinité qui sera résolue par Dynamic Workload Broker en ajoutant une définition d'affinité à la section **Chaîne de la tâche** du travail Tivoli Workload Scheduler à l'aide du nom de travail Tivoli Workload Scheduler comme suit :

```
jobName [-var varName=varValue,...,]-twsaffinity jobname=twsJobName
```

où

twsJobName
Correspond au nom de l'instance du travail Tivoli Workload Scheduler avec lequel vous souhaitez établir une relation d'affinité.

Remarque : Les travaux doivent appartenir au même flot de travaux

Promotion des travaux planifiés sur des pools dynamiques

Cette section explique comment promouvoir un travail critique planifié sur un pool dynamique. Un travail promu peut s'exécuter sur de nombreux agents dynamiques dans le pool dynamique, à l'inverse d'un travail non promu. Cela permet de s'assurer qu'un travail important s'exécute avant d'autres travaux qui sont moins importants.

Pour garantir qu'un travail critique obtient les ressources nécessaires et qu'il est traité dans les délais, spécifiez les variables suivantes :

tws.job.promoted
Cette variable indique si le travail est promu. Les valeurs prises en charge sont **YES** et **NO**. La valeur de cette variable s'applique à tous les travaux soumis dans l'environnement spécifié.

tws.job.resourcesForPromoted
Cette variable est définie dans la définition du pool dynamique et indique la quantité de ressources logiques requises affectée sur un pool dynamique pour promouvoir un travail. La valeur peut être **1** si le travail est promu ou **10** si le travail n'est pas promu. La quantité est indiquée avec cette notation : **${tws.job.resourcesForPromoted}**.

Lorsqu'un travail est planifié sur un pool dynamique, la valeur de la variable **tws.job.promoted** dans le travail détermine le comportement du pool dynamique :

- Si la valeur de la variable **tws.job.promoted** est **NO**, la valeur de la variable **tws.job.resourcesForPromoted** sur le pool dynamique est 10, ce qui signifie que peu de ressources correspondent au besoin.
- Si la valeur de la variable **tws.job.promoted** est **YES**, la valeur de la variable **tws.job.resourcesForPromoted** sur le pool dynamique est 1, ce qui signifie que plusieurs ressources correspondent au besoin car le pool dynamique inclut les

postes de travail avec une quantité de ressource supérieure ou égale à 1 et pas uniquement les postes de travail avec une valeur supérieure ou égale à 10.

Par exemple, vous pouvez écrire un script qui vérifie la valeur affectée à la variable **tws.job.promoted** dans le travail et effectue différentes actions selon que le travail est promu ou non.

Ajout de fonctions dynamiques aux travaux Tivoli Workload Scheduler existants

Cette section décrit la façon de modifier un travail existant pour utiliser les fonctionnalités dynamiques fournies avec les agents dynamiques, les pools et les pools dynamiques.

Vous pouvez modifier vos travaux Tivoli Workload Scheduler existants pour utiliser les fonctionnalités dynamiques fournies avec les agents dynamiques, les pools et les pools dynamiques. Pour modifier un travail existant, procédez comme suit :

1. Installez le nombre de agents dynamiques requis.
2. Le cas échéant, affectez des agents dynamiques aux pools ou créez des pools dynamiques en fonction de vos exigences.
3. Analysez vos travaux Tivoli Workload Scheduler existants et décidez lesquels permettraient d'obtenir les meilleurs résultats lors de l'utilisation de la fonctionnalité dynamique.
4. Connectez-vous à Dynamic Workload Console.
5. A partir de la barre d'outils de navigation, cliquez sur  **Administration** > **Conception de la charge de travail** > **Gestion des définitions de charge de travail**
6. Spécifiez un nom de moteur, distribué ou z/OS. Le concepteur de charge de travail s'ouvre. Les types et les caractéristiques des travaux varient selon que vous sélectionnez un moteur distribué ou z/OS.
7. Dans le panneau Liste de travail, sélectionnez **Rechercher > Définition du travail**. Le panneau de recherche s'affiche.
8. Entrez vos critères de recherche et cliquez sur **Rechercher**.
9. Sélectionnez un ou plusieurs travaux parmi les résultats de recherche et cliquez sur **Editer**. Les travaux sélectionnés s'affichent dans la partie droite du panneau pour modification.
10. Dans l'onglet **Général**, cliquez sur le bouton d'exploration de la zone **Poste de travail**. Le panneau de recherche s'affiche.
11. Entrez vos critères de recherche et cliquez sur **Rechercher**.
12. Sélectionnez le agent dynamique, pool ou pool dynamique approprié, puis cliquez sur **OK**. Le travail est à présent affecté au poste de travail spécifié et s'exécutera dessus lorsqu'il sera planifié.

Scénario métier sur fonction dynamique

Cette section illustre un exemple de scénario métier qui expose les avantages de types de travaux avec options avancées et des fonctions dynamiques.

Une compagnie d'assurances exécute un certain nombre de travaux dans la nuit afin d'enregistrer les données traitées au cours de la journée dans la base de données de sauvegarde. Elle doit également recueillir toutes les données concernant les transactions effectuées pendant la journée sur tous les postes de travail des filiales. Elle utilise des bases de données DB2. A l'aide de types de travaux avec options avancées fournis dans le concepteur de charge de travail, elle crée un travail pour effectuer une sauvegarde de base de données et un autre travail pour extraire les données des transactions quotidiennes. Pour effectuer ces opérations, elle utilise la nouvelle base de données type de travail avec options avancées.

Après le regroupement des données de tous les postes de travail de l'entreprise, elle copie les données de résultat sur un poste de travail unique et les traite afin de générer un rapport. Elle choisit de manière dynamique le meilleur poste de travail disponible en définissant les exigences nécessaires pour exécuter le travail : un poste de travail disposant d'un espace disque important, une UC puissante et le programme requis pour générer le rapport.

Si l'administrateur ne souhaite pas modifier le flot de travaux qu'il a utilisé avant Tivoli Workload Scheduler version 8.6 pour exécuter un travail Java, par exemple, il peut modifier le nom du poste de travail sur lequel le travail doit s'exécuter en insérant le nom d'un pool ou d'un pool dynamique de agents dynamiques où l'exécutable Java est installé. Tivoli Workload Scheduler traduit la syntaxe du travail de sorte qu'il peut être exécuté par le programme Java et affecte le travail aux meilleures ressources disponibles dans le pool.

Le rapport met en évidence le nombre de nouveaux contrat signés et le nombre de clients en retard de paiement. Le message qui est envoyé au chef comptable répertorie le nombre de nouveaux contrats et le nombre de clients en retard.

Une entreprise peut atteindre ses objectifs de la manière suivante :

- Utiliser les nouveaux postes de travail avec des fonctions dynamiques pour exécuter les travaux que l'administrateur a créés pour les postes de travailTivoli Workload Scheduler existants. Pour exécuter ces travaux sur les nouveaux postes de travail, l'administrateur change uniquement le poste de travail sur lequel il souhaite exécuter le travail. Le principal avantage réside dans le fait qu'il peut utiliser les flux de travaux qu'il a précédemment créés sans action supplémentaire.
- Définir plusieurs types de travaux avec options avancées sans avoir de connaissances particulières sur les applications exécutant les travaux.

Ces types de travaux avec options avancées s'exécutent sur les postes de travail suivants :

agents dynamiques
: Poste de travail pouvant exécuter des travaux existants et des types de travaux avec options avancées.

Pools
: Groupes pour lesquels vous pouvez ajouter des agents dynamiques en fonction de vos besoins. Les travaux sont attribués de manière dynamique au meilleur agent disponible.

Pools dynamiques
: Groupes de agents dynamiques pour lesquels vous indiquez vos exigences et laissez Tivoli Workload Scheduler sélectionner les agents dynamiques correspondant à vos besoins. Les travaux sont affectés de manière dynamique au meilleur agent dynamique disponible.

Scénario : création d'une définition de travail et soumission à un pool dynamique

Dans ce scénario, vous devez définir les exigences pour l'exécution du travail lors de la création du pool dynamique, par exemple, vous pouvez inclure dans le pool dynamique tous les postes de travail doté du système d'exploitation Windows, de DB2 et d'une UC utilisée à 50 % au maximum. Le pool dynamique est ensuite rempli avec les postes de travail qui correspondent à vos exigences et il est alors prêt pour le travail qui sera soumis.

Pour créer un pool dynamique et lui soumettre un travail, procédez comme suit :

1. Connectez-vous à Dynamic Workload Console.
2. A partir de la barre d'outils de navigation, cliquez sur  **Administration > Conception de l'environnement de charge de travail > Création de postes de travail**.
3. Sélectionnez un moteur et cliquez sur **OK**.
4. La page **Propriétés du poste de travail** s'affiche.
5. Sélectionnez **Pool dynamique** dans le menu **Type de poste de travail**.
6. Complétez les zones requises.
7. Cliquez sur **Editer les exigences**. La page **Exigences** s'affiche.
8. Spécifiez les exigences suivants :
   - Sélectionnez le système d'exploitation dans la page **Système d'exploitation**.
   - Sélectionnez l'utilisation de l'UC dans la page **Utilisation de l'UC**.
   - Sélectionnez la ressource logique requise dans la page **Ressources logiques**. Pour inclure les postes de travail dotés de DB2, cliquez sur **Ajouter** et spécifiez votre exigence dans la page **Exigences**.
   - Le cas échéant, sélectionnez la politique d'optimisation requise.
9. Cliquez sur **OK** pour sauvegarder vos exigences.
10. Cliquez sur **Sauvegarder** pour enregistrer le pool dynamique.
11. A partir de la barre d'outils de navigation, cliquez sur  **Administration** > **Conception de la charge de travail** > **Gestion des définitions de charge de travail**
12. Spécifiez un nom de moteur, distribué ou z/OS. Le concepteur de charge de travail s'ouvre. Les types et les caractéristiques des travaux varient selon que vous sélectionnez un moteur distribué ou z/OS.
13. Sélectionnez **Nouveau > Définition du travail > Base de données et intégrations > Base de données**.
14. Remplissez les zones obligatoires et spécifiez le pool dynamique que vous avez créé précédemment dans la zone **Poste de travail**.
15. Entrez vos instructions SQL dans l'onglet **SQL**.
16. Cliquez sur **Sauvegarder** pour sauvegarder le travail.

17. A partir de la barre d'outils de navigation, cliquez sur  **Administration** > **Soumission de la charge de travail** > **Soumission de travaux prédéfinis**.

Scénario : création d'une définition de travail et soumission à un pool

Dans ce scénario, vous avez besoin d'exécuter un script de mise à jour d'inventaire. Par conséquent, vous créez un pool pour regrouper les postes de travail dans lequel le programme requis est installé et vous créez un travail pour exécuter le script. Vous sélectionnez le système cible pour le travail du pool `invadmin`.

Pour créer un pool et lui soumettre un travail, procédez comme suit :

1. Connectez-vous à Dynamic Workload Console.
2. A partir de la barre d'outils de navigation, cliquez sur  **Administration > Conception de l'environnement de charge de travail > Création de postes de travail**.
3. Sélectionnez un moteur et cliquez sur **OK**.
4. La page **Propriétés du poste de travail** s'affiche.
5. Sélectionnez **Pool** dans le menu **Type de poste de travail**.
6. Nommez le pool `invadmin` et complétez les zones obligatoires.
7. Sélectionnez les postes de travail que vous souhaitez ajouter au pool. Sélectionnez les postes de travail sur lesquels le programme requis est installé.
8. Cliquez sur **Sauvegarder** pour enregistrer le pool.
9. A partir de la barre d'outils de navigation, cliquez sur  **Administration** > **Conception de la charge de travail** > **Gestion des définitions de charge de travail**
10. Spécifiez un nom de moteur, distribué ou z/OS. Le concepteur de charge de travail s'ouvre. Les types et les caractéristiques des travaux varient selon que vous sélectionnez un moteur distribué ou z/OS.
11. Sélectionnez **Nouveau > Définition du travail > Natif > Exécutable**.
12. Spécifiez un nom pour le travail.
13. Dans la zone **Poste de travail**, spécifiez le pool `invadmin` que vos avez défini précédemment.
14. Accédez à l'onglet **Tâche** et sélectionnez **Commande**.
15. Entrez le nom et le chemin d'accès du fichier exécutable que vous souhaitez exécuter, situé sur chaque poste de travail du pool.
16. Cliquez sur **Sauvegarder** pour sauvegarder le travail.
17. A partir de la barre d'outils de navigation, cliquez sur  **Administration** > **Soumission de la charge de travail** > **Soumission de travaux prédéfinis**.

Limitations spécifiques aux travaux appartenant au flot de travaux USERJOBS lors de la planification dynamique

Fonctions et propriétés partiellement ou pas du tout prises en charge pour le flot de travaux *USERJOBS* lors de la planification dynamique.

Les travaux du flot de travaux *USERJOBS* prennent en charge la plupart des fonctions ou propriétés de la planification dynamique. Le tableau 77 liste certaines fonctions ou propriétés qui sont partiellement ou pas du tout prises en charge.

Tableau 77. Fonctions partiellement ou pas du tout prises en charge pour les travaux du flot de travaux USERJOBS

Fonction/Propriété	*USERJOBS*
Les types de travaux avec des options avancées peuvent perdre certaines informations lorsqu'ils sont déplacés vers le flot de travaux *USERJOBS*.	Il est possible que le type de travail soit manquant ou différent de celui d'origine.
Affichage des types de travaux avec des propriétés d'options avancées à l'aide de la ligne de commande **conman**.	Si vous exécutez la commande **sj <nom_travail>; props**, où <*nom_travail*> désigne les types de travaux avec le nom des options avancées, vous rencontrez les problèmes suivants : • La valeur de la zone `Tâche` de la section `Informations générales` est tronquée. **Remarque :** Ce problème concerne uniquement l'affichage du travail mais pas le travail lui-même. • La section `Informations supplémentaires` affiche le nom interne des propriétés à la place du nom externe traduit.
Affichage des types de travaux avec des options avancées à l'aide de Dynamic Workload Console.	Si vous utilisez les vues graphiques de Dynamic Workload Console, les travaux reflet ne s'affichent pas avec les points standard.
Transmission de variables entre des travaux appartenant à la même instance du flot de travaux. Pour plus d'informations sur cette fonction, voir «Transmission de variables entre des travaux appartenant à la même instance de flot de travaux», à la page 525.	Cette fonction n'est pas prise en charge. La transmission de variables de travail n'est pas résolue lorsque les travaux sont déplacés du flot de travaux initial vers le flot de travaux *USERJOBS*.

Chapitre 13. Utilisation des commandes d'utilitaire

Le présent chapitre décrit les commandes d'utilitaire Tivoli Workload Scheduler. Ces commandes, à l'exception de celles répertoriées ci-dessous, sont installées dans le répertoire *TWS_home*/bin. Les commandes d'utilitaire sont exécutées à partir de l'invite de commande du système d'exploitation.

StartUp est installée dans le répertoire *racine_TWS* et **version** dans le répertoire *racine_TWS*/version.

Description des commandes

Le tableau 78 contient la liste des commandes d'utilitaire avec, pour chaque commande, sa description et les systèmes d'exploitation qu'elle prend en charge.

Tableau 78. Liste des commandes d'utilitaire

Commande	Description	Système d'exploitation
at	Soumet un travail à exécuter à un moment donné.	UNIX
batch	Soumet un travail à exécuter dès que possible.	UNIX
cpuinfo	Renvoie les informations indiquées dans une définition de poste de travail.	UNIX, Windows
datecalc	Convertit la date et l'heure au format requis.	UNIX, Windows
supprimer	Supprime les fichiers script et les fichiers de liste standard par nom.	UNIX, Windows
evtdef	Importe/exporte des définitions d'événements personnalisées.	UNIX, Windows
evtsize	Définit la taille maximale des fichiers de messages d'événements.	UNIX, Windows
exportservedata	Télécharge la liste des instances de courtier de charge de travail à partir de la base de données Tivoli Workload Scheduler et modifie un numéro de port ou un nom d'hôte.	UNIX, Windows
importservedata	Télécharge la liste des instances de Dynamic Workload Broker dans la base de données Tivoli Workload Scheduler après l'édition du fichier temporaire pour changer un numéro de port ou un nom d'hôte.	UNIX, Windows
jobinfo	Renvoie des informations sur un travail.	UNIX, Windows
jobstdl	Renvoie le nom des chemins d'accès des fichiers de liste standard.	UNIX, Windows
listproc	Répertorie les processus. Cette commande n'est pas prise en charge.	Windows
killproc	Supprime les processus. Cette commande n'est pas prise en charge.	Windows
maestro	Renvoie le répertoire principal Tivoli Workload Scheduler.	UNIX, Windows
makecal	Permet de créer des agendas personnalisés.	UNIX, Windows

Tableau 78. Liste des commandes d'utilitaire (suite)

Commande	Description	Système d'exploitation
metronome.pl	Est remplacé par **tws_inst_pull_info**.	UNIX, Windows
morestdl	Affiche le contenu des listes standard.	UNIX, Windows
movehistorydata	Déplace les données présentes dans la base de données Tivoli Workload Scheduler vers les tables d'archivage.	UNIX, Windows
param	Crée, affiche et supprime des variables et des mots de passe utilisateur sur des agents dynamiques.	UNIX, Windows
parms	Affiche, modifie et ajoute des paramètres.	UNIX, Windows
release	Libère les unités d'une ressource.	UNIX, Windows
rmstdlist	Supprime les fichiers de liste standard en fonction de leur âge.	UNIX, Windows
sendevent	Envoie des événements génériques au serveur de processeur d'événement actif.	UNIX, Windows
showexec	Affiche des informations sur l'exécution des travaux.	UNIX
shutdown	Arrête le processus **netman** et, facultativement, WebSphere Application Server.	UNIX, Windows
ShutDownLwa	Arrête l'agent localement.	UNIX, Windows **Remarque :** Sur les systèmes UNIX, peut être exécuté par l'*utilisateur_TWS* ou par l'utilisateur racine uniquement.
StartUp	Démarre le processus **netman** et WebSphere Application Server éventuellement.	UNIX, Windows
StartUpLwa	Démarre l'agent localement.	UNIX, Windows **Remarque :** Sur les systèmes UNIX, peut être exécuté par l'*utilisateur_TWS* ou par l'utilisateur racine uniquement.
tws_inst_pull_info	Collecte des données sur l'instance Tivoli Workload Scheduler locale, le poste de travail WebSphere Application Server et DB2 à des fins de diagnostic. Cette commande est décrite dans le manuel *Tivoli Workload Scheduler - Guide d'identification et de résolution des problèmes*.	UNIX, Windows
version	Affiche les informations de version.	UNIX

at et batch

Soumettez les commandes et travaux ad hoc devant être lancés par Tivoli Workload Scheduler.

Ces commandes s'exécutent uniquement sous UNIX.

Pour plus d'informations sur la disponibilité pour les utilisateurs, reportez-vous aux commandes **at.allow** et **at.deny** ci-dessous.

Syntaxe

at -V | -U

at {-s*flot_travaux* | **-q** *file_attente***}** *spéc_heure*

batch -V | -U

batch [**-s** *flot_travaux*]

Arguments

-V Affiche la version de la commande et quitte l'application.

-U Affiche des informations sur la syntaxe de la commande et quitte l'application.

-s *flot_travaux*

Indique l'ID *ID_flot_travaux* de l'instance du flot de travaux dans laquelle le travail est soumis. Si aucune instance de flot de travaux avec cet *ID_flot_travaux* n'existe, un flot de travaux est créé avec *flot_travaux* comme alias et *ID_flot_travaux*. Le nom du flot doit commencer par une lettre et peut contenir des caractères alphanumériques et des tirets. Il peut comporter jusqu'à 16 caractères.

Si les arguments **-s** et **-q** sont omis, un nom de flot de travaux est sélectionné sur la base de la valeur de la variable d'environnement ***ATSCRIPT***. Si ***ATSCRIPT*** contient le mot **maestro**, l'alias du flot de travaux correspond aux huit premiers caractères du nom de groupe de l'utilisateur. Si ***ATSCRIPT*** n'est pas définie ou comporte une valeur autre que **maestro**, l'alias du flot de travaux sera **at** (pour les travaux soumis avec **at**) ou **batch** (pour les travaux soumis avec **batch**).

Pour plus d'informations sur les flots de travaux, voir «Autres remarques», à la page 543.

Les mots clés suivants s'appliquent uniquement aux travaux **at** :

-q*file_attente*

Indique que le travail doit être soumis dans un flot de travaux portant le nom *file-attente*, lequel ne peut comporter qu'une seule lettre (comprise entre a et z). Pour plus d'informations sur les flots de travaux, voir «Autres remarques», à la page 543.

spéc-heure

Indique l'heure à laquelle le travail va être lancé. La syntaxe est identique à celle de la commande UNIX **at**.

Commentaires

Après avoir entré **at** ou **batch**, lancez les commandes qui constituent le travail. Pour terminer chaque ligne d'entrée, appuyez sur la touche de retour. Le programme se termine par une marque de fin de fichier (généralement **Ctrl+d**) ou par une ligne avec un point (.). Vous avez également la possibilité d'utiliser le signe inférieur (<) pour lire des commandes à partir d'un fichier. Voir «Exemples».

Les informations relatives aux travaux **at** et **batch** sont transmises au gestionnaire de domaine maître où les travaux sont ajoutés dans des flots de travaux du plan de production, `Symphony`. Les travaux sont lancés en fonction des dépendances incluses dans les flots de travaux.

Le shell UNIX utilisé pour exécuter les travaux soumis via les commandes **at** et **batch** est déterminé par la variable ***SHELL_TYPE*** du script de configuration `jobmanrc`. N'utilisez pas le shell C. Pour plus d'informations, voir «Personnalisation du traitement des travaux sur un poste de travail UNIX - jobmanrc», à la page 51.

Une fois soumis, les travaux sont lancés de la même manière que d'autres travaux planifiés. Chaque travail est exécuté dans l'environnement de l'utilisateur qui le soumet. Pour vous assurer que l'environnement est complet, des commandes **set** sont insérées dans le script pour que celui-ci corresponde aux valeurs des variables de l'environnement de l'utilisateur.

Exemples

Pour soumettre un travail dans le flot de travaux *ID_flot_travaux* `sched8` qui sera lancé dès que possible, exécutez la commande suivante :

```
batch -s sched8
commande <Retour>
...
<Control d>
```

Pour soumettre un travail qui sera lancé deux heures après la saisie de la commande, exécutez la commande suivante :

```
at now + 2 hours
commande <Retour>
...
<Control d>
```

Si la variable ***ATSCRIPT*** est indéfinie (null), le travail est soumis dans un flot de travaux dont le nom est identique à celui du groupe de l'utilisateur. Sinon, il est soumis dans un flot de travaux intitulé `at`.

Pour soumettre un travail dans une instance de flot de travaux avec *ID_flot_travaux* `sked-mis` en vue d'un lancement à 17:30, exécutez la commande suivante :

```
at -s sked-mis 17h30
commande <Retour>
...
<Control d>
```

La commande suivante est identique à celle de l'exemple précédent, excepté que le programme lit les commandes du travail à partir d'un fichier :

```
at -s sked-mis 17h30 < ./mon_travail
```

Le fait que les commandes soient lues à partir d'un fichier ne modifie pas la façon dont elles sont traitées. Cela signifie que les commandes sont copiées du fichier `./mon_travail` dans un fichier script.

Remplacement des commandes UNIX

Les commandes UNIX **at** et **batch** standard peuvent être remplacées par des commandes Tivoli Workload Scheduler. Dans l'exemple suivant, nous expliquons comment remplacer les commandes UNIX **at** et **batch** :

```
$ mv /usr/bin/at /usr/bin/uat
$ mv /usr/bin/batch /usr/bin/ubatch
$ ln -s racine_TWS/bin/at /usr/bin/at
$ ln -s racine_TWS/bin/batch /usr/bin/batch
```

Fichiers at.allow et at.deny

Les commandes **at** et **batch** utilisent les fichiers `/usr/lib/cron/`**`at.allow`** et `/usr/lib/cron/`**`at.deny`** pour limiter l'accès. Si le fichier `at.allow` existe, seuls les utilisateurs répertoriés dans le fichier sont autorisés à utiliser **at** et **batch**. Si le fichier n'existe pas, `at.deny` est examiné pour vérifier si les droits d'accès de l'utilisateur sont explicitement refusés. S'il n'existe aucun fichier, seul l'utilisateur **root** est autorisé à utiliser les commandes.

Fichiers script

Les commandes entrées avec **at** ou **batch** sont stockées dans des fichiers script. Ces derniers sont créés par Tivoli Workload Scheduler à l'aide de la convention d'appellation suivante :

racine_TWS`/atjobs/`***epoch.sss***

où :

epoch Indique le nombre de secondes écoulées depuis 00:00, 1/1/70.

sss Représente les trois premiers caractères du nom du flot de travaux.

Remarque : Tivoli Workload Scheduler supprime les fichiers script des travaux non reportés. Vous devez toutefois surveiller l'espace disponible dans le répertoire `atjobs` et supprimer les fichiers caducs si nécessaire.

Noms de travail

Tivoli Workload Scheduler attribue un nom unique à chaque travail **at** et **batch** lors de la soumission. Les noms sont constitués de l'ID processus de l'utilisateur, précédé du nom d'utilisateur, tronqué pour ne plus comporter que huit caractères. Le nom résultant est mis en majuscules.

Autres remarques

- Il est conseillé de créer à l'avance, via le programme composer, les flots de travaux dans lesquels les travaux **at** et **batch** sont soumis. Les flots de travaux peuvent contenir des dépendances qui déterminent l'heure de lancement des travaux. Les flots de travaux doivent contenir au moins le mot clé **carryforward**. Ainsi, les travaux qui n'aboutissent pas ou qui ne sont pas lancés pendant le plan de production courant sont reportés au plan de production suivant.
- Indiquez l'expression **on everyday** pour que les flots de travaux soient sélectionnés chaque jour.
- Utilisez le mot clé **limit** pour limiter le nombre des travaux soumis qui peuvent être exécutés simultanément.
- Utilisez le mot clé **priority** pour définir la priorité des travaux soumis par rapport aux autres travaux.

Si la valeur de temps est antérieure à l'heure courante, elle est prise en compte pour le jour suivant. Si la valeur de temps est postérieure à l'heure courante, elle est prise en compte pour le jour courant.

cpuinfo

Renvoie les informations indiquées dans une définition de poste de travail.

Syntaxe

cpuinfo -V | -U

cpuinfo *poste_de_travail* [*type_info*] [...]

Arguments

-V Affiche la version de la commande et quitte l'application.

-U Affiche des informations sur la syntaxe de la commande et quitte l'application.

poste de travail

Nom du poste de travail.

type_info

Type d'information à afficher. Indiquez un ou plusieurs des éléments suivants :

os_type

Renvoie la valeur de la zone **os** : **UNIX**, **WNT**, **ZOS**, **OTHER** et **IBM i**. La valeur **ZOS** s'applique uniquement aux postes de travail distants utilisés pour communiquer avec un contrôleur Tivoli Workload Scheduler for z/OS.

node Renvoie la valeur de la zone **node**. Pour un serveur courtier de charge de travail, il s'agit du nom d'hôte ou de l'adresse TCP/IP du poste de travail sur lequel vous avez installé Tivoli Workload Scheduler Bridge. Pour un poste de travail distant, il s'agit du nom d'hôte du poste de travail sur lequel le moteur distant est installé. Dans les autres cas, indiquez le nom d'hôte ou l'adresse TCP/IP du poste de travail.

port Renvoie la valeur de la zone **tcpaddr**. Si vous définissez un poste de travail Workload Broker, spécifiez la valeur de la propriété **TWS.Agent.Port** du fichier TWSAgentConfig.properties. Pour les postes de travail de moteur distant, la valeur de cette zone est le numéro de port HTTP utilisé par le moteur distant. Si le protocole HTTPS est utilisé, la valeur de cette zone est 31111.

sslport

Renvoie la valeur de la zone **secureaddr**. Il s'agit du port utilisé pour écouter les connexions SSL entrantes. Pour les postes de travail de moteur distant, la valeur de cette zone est le numéro de port HTTPS utilisé par le moteur distant. Si le protocole HTTP est utilisé, la valeur de cette zone est 31113.

engineaddr

Pour tous types de postes de travail, la valeur de cette zone est 0.

protocole

Renvoie la valeur de la zone **protocol** : HTTP ou HTTPS. Lorsque

le type de poste de travail est un moteur distant, cette valeur indique le protocole utilisé pour communiquer entre le serveur courtier et le moteur distant.

sec_level
Renvoie la valeur de la zone **securitylevel** : NONE, ENABLED, ON ou FORCE.

autolink
Renvoie la valeur de la zone **autolink** : ON ou OFF.

fullstatus
Renvoie la valeur de la zone **fullstatus** : ON ou OFF.

resolvedep
Renvoie ON ou OFF. N'est plus utilisé dans la version 8.6.

behindfirewall
Renvoie la valeur de la zone **behindfirewall** : ON ou OFF.

host Renvoie la valeur de la zone **host**. Il s'agit du nom du poste de travail hébergeant l'agent.

domaine
Renvoie la valeur de la zone **domain**.

ID Renvoie l'identificateur d'agent utilisé par le poste de travail lorsqu'il est connecté au serveur courtier. Pour les postes de travail de type : AGENT, REM-ENG, POOL, D-POOL.

method
Pour les agents étendus et réseau uniquement. Renvoie la valeur de la zone **access**.

serveur
Renvoie la valeur de la zone **server**.

type Renvoie la valeur de la zone **type**. Indique le type de poste de travail : MASTER, MANAGER, FTA, S-AGENT, REM-ENG, AGENT, POOL, D-POOL et X-AGENT.

fuseau_horaire
Renvoie la valeur de la zone **timezone**. Indique le fuseau horaire du poste de travail. Pour un agent étendu, la zone est vide. Pour un poste de travail de moteur distant, il s'agit du fuseau horaire du moteur distant.

version
Renvoie la version de Tivoli Workload Scheduler en cours d'exécution sur le poste de travail. Pour un agent étendu, la zone est vide.

info Renvoie la version du système d'exploitation et le modèle du poste de travail. Pour les agents étendus, la zone est vide. Pour les postes de travail de moteur distant, cette zone affiche le moteur distant.

Commentaires

Les valeurs sont renvoyées, à raison d'une par ligne, dans l'ordre de saisie des arguments sur la ligne de commande. Si aucun argument n'est indiqué, toutes les informations applicables sont renvoyées avec des libellés, à raison d'une par ligne.

Exemples

Les exemples ci-dessous sont basés sur la définition de poste de travail suivante :

```
Nom du poste de travail  Type     Domaine    Mis à jour le  Verrouillé par
---------------  -------  ---------------  ----------  ------------------
RE-ZOS           REM-ENG  -                09/06/2010  -

CPUNAME RE-ZOS
  OS ZOS
  NODE 9.168.119.189 TCPADDR 635
  FOR MAESTRO HOST NC123162_DWB
    TYPE REM-ENG
    PROTOCOL HTTP
END
```

Pour imprimer le **type** et le **protocole** du poste de travail RE-ZOS, exécutez la commande suivante :

```
>cpuinfo RE-ZOS type protocol
REM-ENG
HTTP
```

Pour imprimer toutes les informations du poste de travail RE-ZOS, exécutez la commande suivante :

```
>cpuinfo RE-ZOS
 OS_TYPE: ZOS
 NODE: 9.168.119.189
 PORT: 635
 SSLPORT: 31113
 ENGINEADDR: 0
 PROTOCOLE: HTTP
 AUTOLINK: OFF
 FULLSTATUS: OFF
 RESOLVEDEP: OFF
 BEHINDFIREWALL: OFF
 HOST: NC123162_DWB
 DOMAIN: MASTERDM
 ID: D795263CBCD2365CA7B5C5BC0C3DD363
 SERVER:
 TYPE: REM-ENG
 TIME_ZONE: Europe/Rome
 VERSION: 8.6
 INFO: Remote Engine
```

datecalc

Résout les expressions de date et renvoie les dates au format de votre choix.

Syntaxe

datecalc -V | -U

datecalc *date-base*
[*décalage*]
[**pic** *format*]
[**freedays** *nom_agenda* [**-sa**] [**-su**]]

datecalc -t *heure*
[*date-base*]
[*décalage*]
[**pic** *format*]

datecalc *aaaammjjhhtt*
[*décalage*]
[**pic** *format*]

Arguments

-V Affiche la version de la commande et quitte l'application.

-U Affiche des informations sur la syntaxe de la commande et quitte l'application.

base-date

Indiquez l'un des arguments suivants :

jour | *date* | **today** | **tomorrow** | **scheddate**

où :

jour Indique un jour de la semaine. Les valeurs admises sont les suivantes : **su**, **mo**, **tu**, **we**, **th**, **fr** ou **sa**.

date Indique une date au format *élément*/*élément*[/*élément*], où *élément* représente : *j*[*j*], *m*[*m*] et *aa*[*aa*]. Tout autre forma de date n'est pas valide.

Si deux chiffres sont utilisés pour l'année (*aa*), un nombre supérieur à 70 correspond à une date du 20e siècle et un nombre inférieur à 70 correspond à une date du 21e siècle.

Le paramètre se réfère à la date en cours, et non à la commande *date* d'UNIX. L'exemple ci-après illustre une option utilisant la sortie *date* d'UNIX comme entrée pour le paramètre *date* de Tivoli Workload Scheduler.

```
hdate=´date +"%m/%d/%y"´
echo $hdate
datecalc $hdate pic mm/dd/yyyy
```

Les valeurs admises pour le mois (*m*[*m*]) sont **jan**, **feb**, **mar**, **apr**, **may**, **jun**, **jul**, **aug**, **sep**, **oct**, **nov** ou **dec**.

Les barres obliques (/) peuvent être remplacées par des tirets (-), des points (.), des virgules (,) ou des espaces. Par exemple, vous pouvez entrer une des dates suivantes pour le 28 mars 2005 :

03/28/05
3-28-2005
28.mar.05
05,28,3
mar 28 2005
28 3 05

Si des nombres sont utilisés, il est possible d'entrer une date ambiguë, par exemple, 2,7,04. Dans ce cas, **datecalc** utilise le format de date défini dans le catalogue de messages Tivoli Workload Scheduler pour interpréter la date. Si celle-ci ne respecte pas le format, **datecalc** génère un message d'erreur.

today Indique la date système en cours.

tomorrow
Indique la date système en cours plus un jour ou, dans le cas des calculs d'heure, plus 24 heures.

scheddate
Indique la date du plan de production. Celle-ci peut être différente de la date système. Si vous l'utilisez dans les travaux d'un flot de travaux qui n'est pas reporté, le programme renvoie la date à laquelle le travail doit être exécuté. Elle peut être différente de la date de production du flot de travaux si le travail possède une dépendance `at`.

Si vous l'utilisez dans les travaux d'un flot de travaux qui est reporté, le programme renvoie la date à laquelle le travail aurait dû être exécuté. Elle peut être différente de la date de production du flot de travaux reporté si le travail possède une dépendance `at`. Si la dépendance `at` respecte la syntaxe suivante : **at**=*hhmm* + *n* **days**, les *n* **days** ne seront pas ajoutés à la variable ***TIVOLI_JOB_DATE***. Par conséquent, la commande **datecalc** ne reportera pas les jours en question.

-t *heure* **[***date-base***]**
Indiquez *heure* dans l'un des formats suivants :

now | **noon** | **midnight** | [*h*[*h*][[:]*mm*] **[am** | **pm]** [**zulu]**

où :

now Indique la date et l'heure système en cours.

noon Indique midi (ou 1200).

midnight
Indique minuit (ou 0000).

*h***[***h***][[:]***mm***]**
Indique l'heure au format 12 heures (si des heures **am** ou **pm** sont utilisées) ou au format 24 heures. Le délimiteur (:) facultatif peut être remplacé par un point (.), une virgule (,), une apostrophe ('), la lettre **h** ou un espace. Par exemple, vous pouvez utiliser les formats suivants pour représenter l'heure 20:00 :

8:00pm
20:00
0800pm
2000
8pm
20
8,00pm
20.00
8\'00pm
20 00

zulu Indique que l'heure entrée correspond à l'heure de Greenwich (heure universelle). **datecalc** la convertit en heure locale.

aaaammjjhhtt
Indique l'année, le mois, le jour, l'heure et les minutes, exprimés en douze chiffres exactement. Par exemple, tapez **200505070915** pour représenter le 7 mai 2005, 9 heures 15.

décalage
Indique un décalage par rapport à la *date-base* au format suivant :

{**[+** | **>** | **-** | **<** *nombre* | **nearest]** | **next}** **day[s]** | **weekday[s]** | **workday[s]** | **week[s]** | **month[s]** | **year[s]** | **hour[s]** | **minute[s]** | *jour* | *agenda*

où :

+ | > Indique un décalage par rapport à une date ou une heure ultérieures. Utilisez + (signe plus) sous Windows ; utilisez > (supérieur à) sous UNIX. Veillez à ajouter une barre oblique inversée (\) avant le crochet (>).

- | < Indique un décalage par rapport à une date ou une heure antérieures. Utilisez - (signe moins) sous Windows ; utilisez < (inférieur à) sous UNIX. Veillez à ajouter une barre oblique inversée (\) avant le crochet (>).

nombre Nombre d'unités du type indiqué.

nearest
Indique un décalage par rapport à l'occurrence la plus proche du type d'unité (antérieure ou ultérieure).

next Indique l'occurrence suivante du type d'unité.

day[s] Indique tous les jours.

weekday[s]
Indique tous les jours sauf le samedi et le dimanche.

workday[s]
Identique à **weekday[s]**, mais exclut également les dates de l'agenda **holidays**.

week[s]
Indique sept jours.

month[s]
Indique des mois de l'agenda.

year[s]
Indique des années de l'agenda.

hour[s]
Indique des heures d'horloge.

minute[s]
Indique des minutes d'horloge.

jour Indique un jour de la semaine. Les valeurs admises sont les suivantes : **su**, **mo**, **tu**, **we**, **th**, **fr** ou **sa**.

agenda Indique les entrées d'un agenda portant ce nom.

pic *format*
Indique le format de renvoi de la date et de l'heure. Les caractères du *format* ont les significations suivantes :

m Mois.

d Jour du mois.

y Année.

j Jour du mois du calendrier julien.

h Heure.

t Minute.

^|/ Un espace. Utilisez / (barre oblique) sous Windows ; utilisez ^ (caret) sous UNIX (ajoutez une barre oblique inversée (\) avant le caret (^) si vous vous trouvez dans le shell Bourne).

Vous pouvez également inclure des signes de ponctuation. Il s'agit des délimiteurs utilisés dans *date* et *heure*.

Si aucun format n'est défini, **datecalc** renvoie la date et l'heure au format défini par les variables d'environnement NLS (Native Language Support). Si les variables NLS ne sont pas définies, la langue nationale C est utilisée par défaut.

freedays

Indique le nom d'un agenda de jours chômés *nom_agenda* devant remplacer les **holidays** dans l'évaluation des *jours ouvrés*.

Dans le cas présent, les *jours ouvrés* sont considérés comme étant *tous les jours de la semaine* sauf *samedi*, *dimanche* et toutes les dates répertoriées dans *Nom_agenda*.

Par défaut, *saturday* et *sunday* ne sont pas considérés comme des *jours ouvrés*, sauf si vous indiquez explicitement le contraire en ajoutant **-sa** et **-su** après *nom_agenda*.

Vous pouvez également affecter le nom **holidays** à l'agenda des jours chômés.

Exemples

Pour renvoyer la date suivante (à compter du jour courant) de l'agenda de fin de mois `monthend`, exécutez la commande suivante :

```
>datecalc today next monthend
```

Dans les exemples suivants, la date système en cours est le vendredi 16 avril 2006.

```
>datecalc today +2 days pic mm/jj/aaaa
04/16/2006
```

```
>datecalc today next tu pic aaaa\^mm\^jj
2006 04 16
```

```
>LANG=american;export LANG
>datecalc -t 14:30 tomorrow
Sat, Apr 17, 2006 02:30:00 PM
```

```
>LANG=french;datecalc -t 14:30 tomorrow
Samedi 17 avril 2006 14:30:00
```

Dans l'exemple suivant, l'heure système en cours est 10:24.

```
>datecalc -t now \> 4 hours pic hh:tt
14:24
```

datamigrate

La commande d'utilitaire **datamigrate** du gestionnaire de domaine maître vous permet d'importer dans la base de données les données enregistrées dans les fichiers à plat créés à l'aide de la commande **composer** dans une autre instance de Tivoli Workload Scheduler.

Syntaxe

Utilisez la syntaxe suivante pour importer des données :

datamigrate -u | -v

datamigrate -<*type_objet*> <*fichier_type_objet*> [-tmppath <répertoire_temporaire]

Arguments

-u Affiche des informations sur la syntaxe de la commande et quitte l'application.

-v Affiche la version de la commande et quitte l'application.

-<*type_objet*> **<fichier_type_objet>**

-topology *nom_fichier_topologie*
Importe toutes les définitions de domaine, de poste de travail et de classe de poste de travail enregistrées dans le fichier *nom_fichier_topologie*, créé à l'aide de la commande **composer** dans l'instance de Tivoli Workload Scheduler.

-prompts *nom_fichier_invites*
Importe toutes les définitions d'invite enregistrées dans le fichier *nom_fichier_invites* créé à l'aide de la commande **composer** dans l'instance de Tivoli Workload Scheduler.

-calendars *nom_fichier_calendriers*
Importe toutes les définitions de calendrier enregistrées dans le fichier *nom_fichier_calendriers* créé à l'aide de la commande **composer** dans l'instance de Tivoli Workload Scheduler.

-parms *nom_fichier_params*
Importe toutes les définitions de paramètre enregistrées dans le fichier *nom_fichier_params* créé à l'aide de la commande **composer** dans l'instance de Tivoli Workload Scheduler.

-resources *nom_fichier_ressources*
Importe toutes les définitions de ressource enregistrées dans le fichier *nom_fichier_ressources* créé à l'aide de la commande **composer** dans l'instance de Tivoli Workload Scheduler.

-rcgroups *nom_fichier_groupece*
Importe toutes les définitions de groupe de cycle d'exécution enregistrées dans le fichier *nom_fichier_groupece* créé à l'aide de la commande **composer** dans l'instance de Tivoli Workload Scheduler.

-users *nom_fichier_utilisateurs*
Importe toutes les définitions d'utilisateurs enregistrées dans le fichier *nom_fichier_utilisateurs* créé à l'aide de la commande **composer** dans l'instance de Tivoli Workload Scheduler.

Remarque : Veillez à remplacer toutes les entrées `"*********"` du fichier d'utilisateurs par les véritables mots de passe avant de lancer la commande d'utilitaire **datamigrate**.

-jobs *nom_fichier_travail*
Importe toutes les définitions de travail enregistrées dans le fichier *nom_fichier_travail* créé à l'aide de la commande **composer** dans l'instance de Tivoli Workload Scheduler.

-scheds *nom_fichier_planning*
Importe toutes les définitions de flot de travaux (calendrier) enregistrées dans le fichier *nom_fichier_planning* créé à l'aide de la commande **composer** dans l'instance de Tivoli Workload Scheduler.

-tmppath *chemin_temp*
chemin_temp désigne le chemin d'accès temporaire où la commande **datamigrate** peut insérer ses fichiers de travail temporaires. Le chemin d'accès par défaut est `<rép_installation_TWS>\tmp` sur les systèmes d'exploitation Windows, et `<rép_installation_TWS>/tmp` sur les systèmes UNIX.

Commentaires

Si vous préférez l'utilisation de la commande **datamigrate** à celle de la commande **composer** avec les options add ou replace, veillez à suivre l'ordre d'importation des données qui convient.

Exemples

Pour importer les données exportées de Tivoli Workload Scheduler version 8.6 dans la base de données Tivoli Workload Scheduler version 9.2, lancez les commandes suivantes dans l'ordre indiqué :

```
datamigrate -topology topology86.txt

datamigrate -prompts prompt86.txt

datamigrate -calendars cals86.txt

datamigrate -parms parms86.txt

datamigrate -resources res86.txt

datamigrate -rcgroups rcs86.txt

datamigrate -users users86.txt

datamigrate -jobs jobs86.txt

datamigrate -scheds js86.txt
```

delete

Supprime des fichiers. Bien que cette commande permette de supprimer des fichiers de la liste standard, il est conseillé d'utiliser plutôt la commande `rmstdlist`. Les utilisateurs **maestro** et **root** (sous UNIX) et **Administrateur** (sous Windows) peuvent supprimer tous les fichiers. Les autres utilisateurs ne peuvent supprimer que les fichiers associés à leurs propres travaux.

Syntaxe

delete -V | -U

delete *file_name*

Arguments

-V Affiche la version de la commande et quitte l'application.

-U Affiche des informations sur la syntaxe de la commande et quitte l'application.

file_name

Indique le nom du fichier ou du groupe de fichiers à supprimer. Ce nom doit être placé entre guillemets (") s'il contient des caractères autres que les suivants : caractères alphanumériques, tirets (-), barres obliques (/), barres obliques inversées (\) et traits de soulignement (_). Les caractères génériques sont acceptés.

Remarque : Utilisez cette commande avec précaution. L'utilisation incorrecte de caractères génériques peut entraîner la suppression accidentelle des fichiers.

Exemples

Pour supprimer tous les fichiers de liste standard du 4 novembre 2004, exécutez la commande suivante :

```
delete d:\win32app\maestro\stdlist\2004.4.11\@
```

Le script suivant, inclus dans un travail planifié sous UNIX, supprime le fichier de liste standard de ce travail s'il n'y a pas d'erreur :

```
...
#Suppression de la stdlist de ce travail :
if grep -i error $UNISON_STDLIST
then
exit 1
else
`maestro`/bin/delete $UNISON_STDLIST
fi
...
```

Le script de configuration standard, `jobmanrc`, affecte à la variable ***UNISON_STDLIST*** le nom du fichier de liste standard du travail. Pour plus d'informations sur `jobmanrc`, voir «Personnalisation du traitement des travaux sur un poste de travail UNIX - jobmanrc», à la page 51.

evtdef

Importe/exporte un fichier de définition XML pour le fournisseur d'événements génériques, dans lequel vous pouvez ajouter ou modifier les types d'événements personnalisés. Vous pouvez ensuite, via la commande **sendevent**, envoyer ces événements au serveur de traitement d'événements. Pour plus d'informations, voir aussi «Définition d'événements personnalisés», à la page 145.

Syntaxe

evtdef -U | -V

evtdef [*paramètres de connexion*] **dumpdef** *chemin d'accès au fichier*

evtdef [*paramètres de connexion*] **loaddef** *chemin d'accès au fichier*

Arguments

-U Affiche des informations sur la syntaxe de la commande et quitte l'application.

-V Affiche la version de la commande et quitte l'application.

paramètres de connexion

Si vous utilisez **evtdef** à partir du gestionnaire de domaine maître, les paramètres de connexion ont été configurés lors de l'installation et n'ont pas besoin d'être fournis, à moins que vous ne vouliez pas utiliser les valeurs par défaut.

Si vous utilisez **evtdef** à partir du client de ligne de commande sur un autre poste de travail, les paramètres de connexion peuvent être fournis par l'une des méthodes suivantes :

- Ils sont stockés dans le fichier `localopts`
- Ils sont stockés dans le fichier `useropts`
- Ils sont fournis à la commande dans un fichier de paramètres
- Ils sont fournis à la commande au sein d'une chaîne de commande

Pour une présentation de ces options, voir «Configuration des options pour l'utilisation des interfaces utilisateur», à la page 59. Pour des détails complets sur les paramètres de configuration, voir la rubrique relative à la configuration de l'accès du client de ligne de commande dans *Tivoli Workload Scheduler - Guide d'administration*.

dumpdef *chemin d'accès au fichier*

Télécharge le fichier XML du fournisseur d'événements génériques. Le fichier est téléchargé avec le nom et le chemin d'accès spécifiés par la variable *chemin_d'accès_au_fichier*. Vous pouvez éditer le fichier pour y ajouter des types d'événements personnalisés.

Le nom du fournisseur d'événements génériques fourni avec le produit est `GenericEventPlugIn`. Vous pouvez modifier ce nom en intervenant sur la balise `name` du mot clé `eventPlugin`.

Important : Vous devez utiliser ce nom comme valeur :

- du mot clé `source` de la commande «sendevent», à la page 573
- du mot clé `eventProvider` dans la définition des règles d'événement déclenchées par ces événements personnalisés.

loaddef *chemin d'accès au fichier*

Charge le fichier XML modifié du fournisseur d'événements génériques à partir du nom de fichier et du chemin d'accès spécifiés dans la variable *chemin_d'accès_au_fichier*.

Commentaires

Les schémas linguistiques suivants sont appliqués pour valider les définitions de règles et, selon l'éditeur XML dont vous êtes équipé, pour fournir une aide à la syntaxe::

- eventDefinitions.xsd
- common.xsd

Les fichiers sont situés dans le sous-répertoire `schemas` du répertoire d'installation de Tivoli Workload Scheduler.

Lorsque vous téléchargez le fichier modèle du fournisseur d'événements génériques, ce dernier se présente comme suit :

```
<?xml version="1.0" encoding="UTF-8"?>
<eventDefinitions
xmlns="http://www.ibm.com/xmlns/prod/tws/1.0/event-management/plugins/events"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ibm.com/xmlns/prod/tws/1.0/event-management/
```

```
plugins/events/eventDefinitions.xsd" >
<eventPlugin>
<complexName displayName="Evénement personnalisé" name="GenericEventPlugIn" />
<scopes>
<scope name="Generic">
<scopedef text="{Param1} on {Workstation}" />
</scope>
</scopes>
<!--  Generic Event -->
<event baseAliasName="genericEvt" scope="Generic">
<complexName displayName="Evénement générique" name="Event1" />
<displayDescription>L'événement est envoyé lorsqu'une correspondance avec
l'expression spécifiée est trouvée.</displayDescription>
<property type="string" required="true" wildcardAllowed="true"
multipleFilters="true" minlength="1">
<complexName displayName="Paramètre 1" name="Param1" />
<displayDescription>Valeur de Paramètre 1</displayDescription>
</property>
<property type="string" required="true" wildcardAllowed="false"
multipleFilters="false" minlength="1>
<complexName displayName="Poste de travail" name="Workstation" />
<displayDescription>Poste de travail pour lequel l'événement est
généré.</displayDescription>
</property>
</property>
</event>
</eventPlugin>
</eventDefinitions>
```

Vous pouvez ensuite éditer ce fichier pour ajouter les types de propriété dont vous avez besoin afin de définir un événement spécifique. Les types de propriété suivants sont disponibles :

Tableau 79. Propriétés supplémentaires pouvant être utilisées pour la définition d'événements personnalisés.

Type de propriété	A ajouter dans le fichier d'événements XML comme illustré
boolean	`<property type="boolean" required="false" wildcardAllowed="false" multipleFilters="true" minlength="1"> <complexName displayName="Boolean field" name="Boolean"/> <displayDescription>Add a boolean field</displayDescription> </property>`
date	`<property type="date" required="false" wildcardAllowed="false" multipleFilters="true" minlength="1"> <complexName displayName="Date field" name="Date"/> <displayDescription>Add a date field</displayDescription> </property>`
datetime	`<property type="datetime" required="false" wildcardAllowed="false" multipleFilters="true" minlength="1"> <complexName displayName="Date Time field" name="Datetime"/> <displayDescription>Add a date time field</displayDescription> </property>`
datetimeutc	`<property type="datetimeutc" required="false" wildcardAllowed="false" multipleFilters="true" minlength="1"> <complexName displayName="Date Time UTC field" name="Datetimeutc"/> <displayDescription>Add a date time UTC field</displayDescription> </property>`
duration	`<property type="duration" required="false" wildcardAllowed="false" multipleFilters="true" minlength="1"> <complexName displayName="Duration field" name="Duration"/> <displayDescription>Add a duration field</displayDescription> </property>`
fileSize	`<property type="fileSize" required="false" wildcardAllowed="false" multipleFilters="true" minlength="1"> <complexName displayName="File size field" name="filesize"/> <displayDescription>Add a file size field</displayDescription> </property>`

Tableau 79. Propriétés supplémentaires pouvant être utilisées pour la définition d'événements personnalisés. (suite)

Type de propriété	A ajouter dans le fichier d'événements XML comme illustré
nonnegativeinteger	<property type="nonnegativeinteger" required="false" wildcardAllowed="false" multipleFilters="true" minlength="1"> <complexName displayName="Non negativeinteger field" name="nonnegativeinteger"/> <displayDescription>Add a non negativeinteger field</displayDescription> </property>
numeric	<property type="numeric" required="false" wildcardAllowed="false" multipleFilters="true" minlength="1"> <property type="numeric" required="false" wildcardAllowed="false" multipleFilters="true" minlength="1"> <displayDescription>Add a numeric field</displayDescription> </property>
percentage	<property type="percentage" required="false" wildcardAllowed="false" multipleFilters="true" minlength="1"> <complexName displayName="Percentage field" name="percentage"/> <displayDescription>Add a percentage field</displayDescription> </property>
string	<property type="string" required="true" wildcardAllowed="true" multipleFilters="true" minlength="1"> <complexName displayName="String with wildcards" name="StringWithWildcards"/> <displayDescription>Add a string with wildcards</displayDescription> </property> or <property type="string" required="true" wildcardAllowed="false" multipleFilters="true" minlength="1"> <complexName displayName="String without wildcards" name="StringWithoutWildcards"/> <displayDescription>Add a string without wildcards</displayDescription> </property>
time	<property type="time" required="false" wildcardAllowed="false" multipleFilters="true" minlength="1"> <complexName displayName="Time field" name="Time"/> <displayDescription>Add a time field</displayDescription> </property>

Vous pouvez modifier les valeurs de tous les attributs de propriété, sauf `type`, pour répondre à vos besoins.

Les propriétés ainsi définies sont converties dans les zones d'entrée après le téléchargement de la définition d'événement et l'ouverture de cette dernière dans Dynamic Workload Console.

Vous pouvez également définir plusieurs événements en répétant les sections `<eventPlugin>...</eventPlugin>`. Par exemple :

```
<?xml version="1.0" encoding="UTF-8"?>
<eventDefinitions
xmlns="http://www.ibm.com/xmlns/prod/tws/1.0/event-management/plugins/events"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ibm.com/xmlns/prod/tws/1.0/event-management/
plugins/events/eventDefinitions.xsd" >
<eventPlugin>
<complexName displayName="Evénement personnalisé" name="GenericEventPlugIn" />
<scopes>
<scope name="Art042StockQuantity">
<scopedef text="{Art042pieces}"/>
</scope>
</scopes>
<!--  Generic Event -->
<event baseAliasName="genericEvt" scope="Generic">
<complexName displayName="Stock of article 042 reaches minimum level" name="art042qty"/>
<displayDescription>The event is sent when the number of art.42 items on stock reaches
minimum level.</displayDescription>
<property type="numeric" required="true" wildcardAllowed="false"
multipleFilters="true" minlength="1">
<complexName displayName="Art042 items on stock" name="art042items"/>
```

```
<displayDescription>The number of art042 items left</displayDescription>
</property>
</event>
</eventPlugin>
<eventPlugin>
<complexName displayName="Evénement personnalisé" name="GenericEventPlugIn" />
<scopes>
<scope name="HDAlmostFull">
<scopedef text="{HDNearingFullPercent} on {Workstation}"/>
</scope>
</scopes>
<!-- Generic Event 2-->
<event baseAliasName="Hard drive saturation" scope="Generic">
<complexName displayName="Hard drive saturation" name="HDSatEvent"/>
<displayDescription>displayDescription>The event is sent when the percentage field
reaches the warning level.</displayDescription>
<property type="percentage" required="true" wildcardAllowed="false"
multipleFilters="true" minlength="1>
<complexName displayName="Percentage Full" name="PercentFull"/>
<displayDescription>The percentage of total disk space used</displayDescription>
</property>
<property type="string" required="true" wildcardAllowed="false"
multipleFilters="false" minlength="1>
<complexName displayName="Poste de travail" name="Workstation" />
<displayDescription>The workstation where the hard drive is installed</displayDescription>
</property>
</event>
</eventPlugin>
</eventDefinitions>
```

Exemples

Dans cet exemple, vous pouvez :

1. Télécharger le fichier XML du fournisseur d'événements génériques sous la forme `c:\custom\myevents.xml`

   ```
   evtdef dumpdef c:\custom\myevents.xml
   ```

2. Editer le fichier pour y ajouter des type d'événements personnalisés.
3. Une fois l'opération terminée, téléchargez le fichier XML du fournisseur d'événements génériques à partir du fichier `c:\custom\myevents.xml`.

   ```
   evtdef loaddef c:\custom\myevents.xml
   ```

evtsize

Définit la taille des fichiers message de Tivoli Workload Scheduler. Cette commande est utilisée par l'administrateur Tivoli Workload Scheduler pour augmenter la taille d'un fichier de messages après la réception du message “End of file on events file” (fin du fichier des événements atteinte)ou pour contrôler la taille de la file d'attente de messages contenue dans le fichier de messages. Vous devez être un utilisateur **maestro** ou **root** sous UNIX, ou **Administrateur** sous Windows pour exécuter **evtsize**. Arrêtez le moteur IBM Tivoli Workload Scheduler avant d'exécuter cette commande.

Syntaxe

evtsize -V | -U

evtsize *file_name taille*

evtsize -compact *file_name* [*taille*]

evtsize -info *file_name*

evtsize -show *file_name*

evtsize -info | -show pobox

Arguments

-V Affiche la version de la commande et quitte l'application.

-U Affiche des informations sur la syntaxe de la commande et quitte l'application.

-compact *file_name* **[***taille***]**
Réduit la taille du fichier de message indiqué à la taille occupée par les messages présents lors de l'exécution de la commande. Facultativement, vous pouvez utiliser ce mot clé pour indiquer également une nouvelle taille de fichier.

-info *nom_de_fichier*
Affiche le pourcentage d'utilisation de la file d'attente de messages contenue dans le fichier de message.

-show *file_name*
Affiche la taille de la file d'attente de messages contenue dans le fichier de messages.

file_name
Nom du fichier des événements. Indiquez l'un des arguments suivants :

```
Courier.msg
Intercom.msg
Mailbox.msg
Planbox.msg
pobox/poste_de_travail.msg
mirrorbox.msg
```

taille Taille maximale, en octets, du fichier des événements. Elle ne doit pas être inférieure à 1048576 octets (1 Mo).

Lors de la création du fichier par Tivoli Workload Scheduler, la taille maximale est définie sur 10 Mo.

Remarque : La taille du fichier de messages est supérieure ou égale à la taille réelle de la file d'attente de messages qu'il contient et elle augmente progressivement jusqu'à ce que la file d'attente de messages soit vide ; au moment où cela se produit, le fichier de messages est vidé.

-info | -show pobox
Affiche le nom du fichier de message, dans le répertoire `pobox`, avec la taille de file d'attente la plus grande calculée en pourcentage de la taille de fichier totale. Le nom du fichier et le pourcentage utilisé sont tous deux renvoyés. **-info** et **-show** renvoient les mêmes résultats.

Exemples

Pour définir la taille maximale du fichier `Intercom.msg` à 20 Mo, exécutez la commande suivante :

```
evtsize Intercom.msg 20000000
```

Pour définir la taille maximale du fichier pobox du poste de travail chicago à 15 Mo, exécutez la commande suivante :

```
evtsize pobox\chicago.msg 15000000
```

La commande suivante :

```
evtsize -show Intercom.msg
```

renvoie le résultat suivant :

```
Tivoli Workload Scheduler (UNIX)/EVTSIZE 8.3 (1.2.2.4) Eléments sous licence -
Propriété d'IBM(R)
5698-WSH
(C) Copyright IBM Corp 1998, 2006 All rights reserved.
US Government User Restricted Rights
Use, duplication or disclosure restricted by
GSA ADP Schedule Contract with IBM Corp.
IBM est une marque d'International Business Machines
Corporation aux Etats-Unis et/ou dans certains autres pays.
AWSDEK703I Taille de la file d'attente courant 240,
10000000 octets maximum (48 en lecture, 288 en écriture)
```

où :

240 représente la taille de la file d'attente actuelle du fichier Intercom.msg

10000000
représente la taille maximale du fichier Intercom.msg

48 en lecture
Indique la position du pointeur pour lire les enregistrements

288 en écriture
Indique la position du pointeur pour écrire les enregistrements

Si la commande suivante :

```
evtsize -info Mailbox.msg
```

renvoie :

```
25
```

cela signifie que 25 pour cent du fichier a été utilisé.

jobinfo

Utilisée dans un script de travail pour renvoyer des informations relatives au travail. Cette commande n'est pas prise en charge sur les agents dynamiques, les pools, les pools dynamiques et les types de travaux avec options avancées.

Syntaxe

jobinfo -V | -U

jobinfo *option-travail* [...]

Arguments

-V Affiche la version de la commande et quitte l'application.

-U Affiche des informations sur la syntaxe de la commande et quitte l'application.

option-travail

Option de travail. Indiquez un ou plusieurs des éléments suivants :

confirm_job
Renvoie **YES** si le travail doit être confirmé.

is_command
Renvoie **YES** si le travail a été planifié ou soumis à l'aide de la structure **docommand**.

nom_travail
Renvoie le nom du travail sans le nom du poste de travail et le nom du flot de travaux.

job_pri
Renvoie le niveau de priorité du travail.

programmatic_job
Renvoie **YES** si le travail a été soumis à l'aide de la commande **at** ou **batch**. Sous UNIX uniquement.

re_job Renvoie **YES** si le travail est de nouveau exécuté suite à l'exécution d'une commande **conman rerun** ou de l'option de reprise rerun.

re_type
Renvoie l'option de reprise du travail (**stop**, **continue** ou **rerun**).

rstrt_flag
Renvoie **YES** si le travail est exécuté en tant que travail de reprise.

rstrt_retcode
Si le travail en cours est un travail de reprise, renvoie le code retour du travail parent.

schedule
Renvoie le nom du flot de travaux dans lequel le travail s'exécute.

schedule_ia
Renvoie l'heure et la date où le lancement du flot de travaux est planifié.

schedule_id
Renvoie l'ID *ID_flot_travaux* du flot de travaux dans lequel le travail est soumis.

time_started
Renvoie l'heure à laquelle l'exécution du travail a commencé.

Commentaires

Les valeurs d'option de travail sont renvoyées, à raison d'une par ligne, dans l'ordre dans lequel elles ont été demandées.

Exemples

1. Le fichier script `/jcl/backup` est référencé à deux reprises et les noms de travaux **partback** et **fullback** lui sont attribués. Si le travail est exécuté sous la forme **partback**, il effectue une sauvegarde partielle. S'il est exécuté sous la forme **fullback**, il effectue une sauvegarde complète. Cette distinction est déterminée dans le script à l'aide de commandes similaires aux commandes suivantes :

```
#Choix entre partiel (1) ou intégral (2) :
if [ "`\`maestro\`/bin/jobinfo job_name`" = "PARTBACK" ]
then
bkup=1
```

```
else
bkup=2
fi
...
```

2. Pour afficher le code retour du travail parent, exécutez la commande suivante si le travail courant est un travail de reprise :

   ```
   $ jobinfo rstrt_retcode
   ```

   Le premier travail (travail parent) a été défini dans le script `recovery.sh` alors que le second travail (travail de reprise) est activé uniquement si le premier se termine de manière anormale.

   Si vous associez un code retour à la commande **jobinfo rstrt_retcode**, vous pouvez diriger le travail de reprise pour réaliser différentes actions selon le code retour du travail parent. Dans l'exemple suivant, nous avons illustré un travail de reprise :

   ```
   $JOBS
   MASTER#DBSELOAD DOCOMMAND "/usr/local/tws/maestro/scripts/populate.sh"
   STREAMLOGON "^TWSUSER^"
   DESCRIPTION "alimentation manuelle de la base de données"
   RECOVERY RERUN AFTER MASTER#RECOVERY
   RCCONDSUCC "(RC = 0) OR ((RC > 4) AND (RC < 11))"
   ```

   Remarque : 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.

   Dans l'exemple suivant, le travail de reprise proprement dit est défini :

   ```
   $ JOBS
   MASTER#RECOVERY DOCOMMAND "^TWSHOME^/scripts/recovery.sh"
   STREAMLOGON "^TWSUSER^"
   DESCRIPTION "reprise manuelle de la base de données"
   RECOVERY STOP
   ```

jobstdl

Renvoie les noms des listes standard. Cette commande doit être exécutée par l'utilisateur pour lequel Tivoli Workload Scheduler a été installé. Si vous utilisez la commande sans paramètres, vous devez être connecté en tant qu'utilisateur Tivoli Workload Scheduler.

Syntaxe

jobstdl -V | -U

jobstdl
[**-day** *num*]
[{**-first** | **-last** | **-num** *n* | **-all**}]
[**-twslog**]
[{**-name** [*"nom_flot_travaux* [(*hhmm date*),(*ID_flot_travaux*)].]*nom_travail*"
| *num_travail* | **-schedid** *ID_flot_travaux.nom_travail*}]

Arguments

-V Affiche la version de la commande et quitte l'application.

-U Affiche des informations sur la syntaxe de la commande et quitte l'application.

-day *num*
Renvoie les noms de liste standard qui datent du nombre de jours indiqué (1 pour hier, 2 pour avant-hier, etc.). Valeur par défaut : zéro (aujourd'hui).

-first Renvoie le nom de la première liste standard éligible.

-last Renvoie le nom de la dernière liste standard éligible.

-num *n*
Renvoie le nom de la liste standard correspondant à l'exécution indiquée d'un travail.

-all Renvoie le nom de toutes les listes standard éligibles.

-twslog
Renvoie le chemin du fichier *stdlist* du jour courant.

-name **["***nom_flot_travaux***[(***hhmm date***), (***ID_flot_travaux***)].]***nom_travail***"** | *num_travail*
Indique l'instance du flot de travaux et le nom du travail pour lesquels des noms de fichier de liste standard sont renvoyés.

numéro_travail
Indique le numéro du travail pour lequel des noms de liste standard sont renvoyés.

-schedid *ID_flot_travaux.nom_travail*
Indique l'ID de flot de travaux et le nom du travail pour lequel des noms de fichier de liste standard sont retournés.

Commentaires

Les noms de fichier sont renvoyés dans un format approprié pour être entrés dans d'autres commandes. Si le programme renvoie plusieurs noms, il les sépare par un espace.

Lorsque vous utilisez la syntaxe complète de l'argument **-name**, les crochets dans l'expression `[(hhmm date), (ID_flot_travaux)]` font partie de la commande et ne sont pas des indicateurs de syntaxe. De plus, toute la chaîne d'identification de travail doit être placée entre guillemets si la partie désignant l'instance de flot de travaux contient des blancs. Par exemple, parce que *schedtime*, représenté par *hhmm date*, contient un blanc, vous devez placer toute l'identification de travail entre guillemets.

Vous pouvez également exécuter des versions abrégées de l'argument **-name** à l'aide d'une syntaxe plus simple. Si vous souhaitez obtenir moins de sorties spécifiques de la commande, vous pouvez indiquer uniquement *schedtime* (la *date* n'est pas requise si c'est pour le même jour) ou l'*ID_flot_travaux* conjointement avec le *nom_travail*. Tant qu'il n'y a pas d'espaces dans les arguments, vous pouvez omettre les guillemets. Vous pouvez également omettre les crochets droits si vous n'indiquez pas à la fois les *schedtime* et *ID_flot_travaux*.

Les exemples suivants présentent la syntaxe que vous devez utiliser avec le même argument **-name** pour les différents types d'informations que vous attendez en retour, allant du plus spécifique au plus général. Dans l'exemple, `job_stream1` est le nom du flot de travaux, `0600 04/05/06` est le temps planifié, `0AAAAAAAAAAAAAB5` est l'ID de flot de travaux, et `job1` est le nom du travail. Le numéro de travail de `job1` est `310`. Vous pouvez exécuter **jobstdl** pour `job1` comme suit :

```
jobstdl -name "job_stream1[(0600 04/05/10),(0AAAAAAAAAAAAAB5)].job1"
```

Renvoie le nom de fichier de liste standard de `job1` pour l'instance spécifique de `job_stream1` avec les *schedtime* et *ID_flot_travaux* spécifiés.

```
jobstdl -name job_stream1(0AAAAAAAAAAAAAB5).job1
```

Renvoie le nom de fichier de liste standard de job1 pour l'instance de job_stream1 avec l'ID 0AAAAAAAAAAAAAB5.

```
jobstdl -name "job_stream1(0600 04/05/10).job1"
```

Renvoie les noms de fichier de liste standard de job1 pour toutes les instances possibles de job_stream1 planifiées pour s'éxécuter à 0600 le 04/05/10.

```
jobstdl -name job_stream1(0600).job1
```

Renvoie les noms de fichier de liste standard de job1 pour toutes les instances possibles de job_stream1 planifiées pour s'exécuter à 0600 le jour courant.

```
jobstdl -name 310
```

Renvoie les noms de fichier de liste standard de job1 pour toutes les instances pour lesquelles il avait le numéro de travail 310.

Exemples

Pour renvoyer les noms de chemin de tous les fichiers de liste standard du jour courant, exécutez la commande suivante :

```
jobstdl
```

Pour renvoyer le nom de chemin de la liste standard pour la première exécution du travail MY_CPU#ELI[(1824 03/09/06),(0AAAAAAAAAAAAAEE)].DIR du jour courant, exécutez la commande suivante :

```
jobstdl -first -name "MY_CPU#ELI[(1824 03/09/06),(0AAAAAAAAAAAAAEE)].DIR"
```

Pour retourner le chemin d'accès de la liste standard pour la première exécution du travail 0AAAAAAAAAAAAAEE.DIR dans la journée, exécutez la commande suivante :

```
jobstdl -first -schedid 0AAAAAAAAAAAAAEE.DIR
```

Pour renvoyer le nom de chemin de la liste standard pour la deuxième exécution du travail MY_CPU#ELI[(1824 03/09/06),(0AAAAAAAAAAAAAEE)].DIR du jour courant, exécutez la commande suivante :

```
jobstdl -num 2 -name "MY_CPU#ELI[(1824 03/09/06),(0AAAAAAAAAAAAAEE)].DIR"
```

Pour renvoyer les noms de chemin des fichiers de la liste standard pour toutes les exécutions du travail MY_CPU#ELI[(1824 03/09/06),(0AAAAAAAAAAAAAEE)].DIR remontant à trois jours, exécutez la commande suivante :

```
jobstdl -day 3 -name "MY_CPU#ELI[(1824 03/09/06),(0AAAAAAAAAAAAAEE)].DIR"
```

Pour renvoyer le nom de chemin de la liste standard pour la dernière exécution du travail MY_CPU#ELI[(1824 03/09/06),(0AAAAAAAAAAAAAEE)].DIR remontant à quatre jours, exécutez la commande suivante :

```
jobstdl -day 4 -last -name "MY_CPU#ELI[(1824 03/09/06),(0AAAAAAAAAAAAAEE)].DIR"
```

Pour renvoyer le nom de chemin de la liste standard du travail **455**, exécutez la commande suivante :

```
jobstdl 455
```

Pour imprimer le contenu du fichier de liste standard du travail **455**, exécutez la commande suivante :

```
cd `maestro`/bin
lp -p 6 `jobstdl 455`
```

maestro

Renvoie le nom de chemin du répertoire principal Tivoli Workload Scheduler, désigné par *racine_TWS*.

Syntaxe

maestro [**-V** | **-U**]

Arguments

-V Affiche la version de la commande et quitte l'application.

-U Affiche des informations sur la syntaxe de la commande et quitte l'application.

Exemples

Pour afficher le répertoire principal Tivoli Workload Scheduler, exécutez la commande suivante :

```
$ maestro
/usr/lib/maestro
```

Pour modifier le répertoire principal Tivoli Workload Scheduler, exécutez la commande suivante :

```
$ cd `maestro`
```

makecal

Crée un agenda personnalisé. Sous UNIX, vous devez utiliser le shell Korn pour exécuter la commande.

Syntaxe

makecal [**-V** | **-U**]

makecal
[**-c** *name*]
-d *n*
| **-e**
| {**-f 1** | **2** | **3 -s** *date*}
| **-l**
| **-m**
| **-p** *n*
| {**-r** *n* **-s** *date*}
| **-w** *n*
[**-i** *n*]
[**-x** | **-z**]
[**-freedays** *nom_agenda* [**-sa**] [**-su**]]

Arguments

-V Affiche la version de la commande et quitte l'application.

-U Affiche des informations sur la syntaxe de la commande et quitte l'application.

-c *nom* Indique un nom pour l'agenda. Les mots clés Tivoli Workload Scheduler (tels que *Freedays* ou *Schedul*e) ne peuvent pas être utilisés comme noms

d'agenda. Ce nom peut comporter jusqu'à huit caractères alphanumériques et doit commencer par une lettre. N'utilisez pas les noms des jours de la semaine pour nommer un agenda. Le nom par défaut est le suivant : **C***hhmm*, où *hhmm* est l'heure courante exprimée en heures et en minutes.

-d *n*
Indique le *énième* jour de chaque mois.

-e
Indique le dernier jour de chaque mois.

-f 1 | 2 | 3
Crée un agenda de fin de mois fiscal contenant le dernier jour du mois fiscal. Indiquez un des formats suivants :

1 Format de semaine 4-4-5

2 Format de semaine 4-5-4

3 Format de semaine 5-4-4

Cet argument requiert l'argument **-s**.

-i *n*
Indique que *n* dates doivent être placées dans l'agenda.

-l
Indique le dernier jour ouvré de chaque mois. Cet argument ne fonctionne correctement que si le plan de production (fichier `Symphony`) et l'agenda **holidays** existent déjà.

Remarque : L'utilisation de cet argument a pour résultat d'inclure également dans le nouvel agenda le dernier jour ouvré du mois précédant la date de création de l'agenda.

-m
Indique le premier et le quinzième jour de chaque mois.

-p *n*
Indique le jour ouvré précédant le *énième* jour de chaque mois. Cet argument ne fonctionne correctement que si le plan de production (fichier `Symphony`) et l'agenda **holidays** existent déjà.

-r *n*
Indique chaque *énième* jour. Cet argument requiert l'argument **-s**.

-s *date*
Indique la date de début pour les arguments **-f** et **-r**. Cette date doit être valide et non ambiguë et placée entre apostrophes ; par exemple, utilisez **JAN 10 2005** et non **1/10/05**. Pour plus d'informations sur les formats de date, voir *base-date* pour **datecalc** à la page «base-date», à la page 547.

-w *n*
Indique le jour ouvré suivant le *énième* jour du mois. Cet argument ne fonctionne correctement que si le plan de production (fichier Symphony) et l'agenda **holidays** existent déjà.

-x
Envoie la sortie de l'agenda à **stdout** au lieu de l'ajouter dans la base de données.

-z
Ajoute l'agenda dans la base de données et compile le plan de production (fichier`Symphony`).

Remarque : Cet argument soumet à nouveau des travaux et des flots de travaux à partir du plan de production du jour. Il est peut-être nécessaire d'annuler des flots de travaux et des travaux.

-freedays
Indique le nom d'un agenda de jours chômés *nom_agenda* devant remplacer les **holidays** dans l'évaluation des *jours ouvrés*.

Dans le cas présent, les *jours ouvrés* sont considérés comme étant *tous les jours de la semaine* sauf *samedi*, *dimanche* et toutes les dates répertoriées dans *Nom_agenda*.

Par défaut, le *samedi* et le *dimanche* ne sont pas considérés comme des *jours ouvrés*, sauf si vous indiquez explicitement le contraire en ajoutant **-sa** et/ou **-su** après *nom_agenda*.

Vous pouvez également affecter le nom **holidays** à l'agenda des jours chômés.

Ce mot clé affecte le traitement de **makecal** avec les options **-l**, **-p** et **-w**.

Exemples

Pour créer un agenda de deux ans où le dernier jour de chaque mois est sélectionné, exécutez la commande suivante :

```
makecal -e -i 24
```

Pour créer un agenda de 30 jours débutant le 30 mai 2005 et où un jour sur trois est sélectionné, exécutez la commande suivante :

```
makecal -r 3 -s "30 MAY 2005" -i 30
```

metronome

La commande metronome est remplacée par **tws_inst_pull_info**. Voir *IBM Tivoli Workload Scheduler Guide d'identification et de résolution des problèmes* pour des informations sur cette commande.

morestdl

Affiche le contenu des listes standard. Cette commande doit être exécutée par l'utilisateur pour lequel Tivoli Workload Scheduler a été installé. Si vous utilisez la commande sans paramètres, vous devez être connecté en tant qu'utilisateur Tivoli Workload Scheduler. Cette commande est prise en charge pour les agents tolérants aux pannes et les agents standard.

Syntaxe

morestdl -V | -U

morestdl
[**-day** *num*]
[**-first** | **-last** | **-num** *n* | **-all**]
[**-twslog**]
[{**-name** ["*nom_flot_travaux* [(*hhmm date*),(*ID_flot_travaux*)].]*nom_travail*"
| *num_travail* | **-schedid** *ID_flot_travaux.nom_travail*}]

Arguments

-V Affiche la version de la commande et quitte l'application.

-U Affiche des informations sur la syntaxe de la commande et quitte l'application.

-day *num*
Affiche les listes standard qui datent du nombre de jours indiqué (1 pour hier, 2 pour avant-hier, etc.). Valeur par défaut : zéro (aujourd'hui).

-first Affiche la première liste standard éligible.

-last Affiche la dernière liste standard éligible.

-num *n*
Affiche la liste standard correspondant à l'exécution indiquée d'un travail.

-all Affiche toutes les listes standard éligibles.

-twslog
Affiche le contenu du fichier *stdlist* du jour courant.

-name ["*nom_flot_travaux* **[(***hhmm date***),(***ID_flot_travaux***)].]***nom_travail***"|** *num_travail*
Indique l'instance du flot de travaux et le nom du travail pour lesquels le fichier de la liste standard est affiché.

numéro_travail
Indique le numéro du travail pour lequel la liste standard est affichée.

-schedid *ID_flot_travaux***.***nom_travail*
Indique l'ID de flot de travaux et le nom du travail pour lequel des noms de fichier de liste standard sont retournés.

Commentaires

Les crochets droits dans l'expression `[(hhmm date), (ID_flot_travaux)]` font partie de la commande et ne sont pas des indicateurs de syntaxe. Vous pouvez donc définir l'un ou l'autre des éléments suivants pour l'argument **-name** :

```
morestdl -name ["nom_flot_travaux[(hhmm date),(ID_flot_travaux)].nom_travail"
morestdl -name num_travail
```

Toute la chaîne d'identification de travail doit être placée entre guillemets si la partie désignant l'instance de flot de travaux contient des blancs. Par exemple, parce que *schedtime*, représenté par `hhmm date`, contient un blanc, vous devez placer toute l'identification de travail entre guillemets.

Si vous voulez simplement identifier un nom de travail, vous n'avez pas besoin de le placer entre guillemets.

Voici un exemple de la syntaxe à utiliser lors de l'identification d'un travail avec et sans son flot de travaux. Dans l'exemple, `job_stream1` est le nom du flot de travaux, `0600 04/05/06` est le temps planifié, `0AAAAAAAAAAAAAB5` est l'ID de flot de travaux, et `job1` est le nom du travail. Vous pouvez exécuter la commande **morestdl** sur `job1` en utilisant l'un des deux formats ci-dessous :

```
morestdl -name "job_stream1[(0600 04/05/06),(0AAAAAAAAAAAAAB5)].job1"
morestdl -name job1
```

Exemples

Pour afficher le fichier de liste standard pour la première exécution du travail `MY_CPU#ELI[(1824 03/09/06),(0AAAAAAAAAAAAAEE)].DIR` du jour courant, exécutez la commande suivante :

```
morestdl -first -name "MY_CPU#ELI[(1824 03/09/06),(0AAAAAAAAAAAAAEE)].DIR"
```

Pour afficher le fichier de liste standard pour la première exécution du travail `0AAAAAAAAAAAAAEE.DIR` du jour courant, exécutez la commande suivante :

```
morestdl -first -schedid 0AAAAAAAAAAAAAEE.DIR
```

Pour afficher le fichier de liste standard pour la deuxième exécution du travail `MY_CPU#ELI[(1824 03/09/06),(0AAAAAAAAAAAAAEE)].DIR` du jour courant, exécutez la commande suivante :

```
morestdl -num 2 -name "MY_CPU#ELI[(1824 03/09/06),(0AAAAAAAAAAAAAEE)].DIR"
```

Pour afficher les fichiers de liste standard pour toutes les exécutions de travail MY_CPU#ELI[(1824 03/09/06),(0AAAAAAAAAAAAAEE)].DIR remontant à trois jours, exécutez la commande suivante :

```
morestdl -day 3 -name "MY_CPU#ELI[(1824 03/09/06),(0AAAAAAAAAAAAAEE)].DIR"
```

Pour afficher le fichier de liste standard pour la dernière exécution de travail MY_CPU#ELI[(1824 03/09/06),(0AAAAAAAAAAAAAEE)].DIR remontant à quatre jours, exécutez la commande suivante :

```
morestdl -day 4 -last -name "MY_CPU#ELI[(1824 03/09/06),(0AAAAAAAAAAAAAEE)].DIR"
```

Pour imprimer le fichier de liste standard du travail 455, exécutez la commande suivante :

```
morestdl 455 | lp -p 6
```

parms

Gère les paramètres définis en local sur les postes de travail. Les paramètres gérés par **parms** peuvent uniquement être utilisés dans une définition de travail ou de flot de travaux avec les mots clés **scriptname** ou **opens** ou dans un fichier de script de travail.

Ces paramètres sont résolus lors de la soumission sur le poste de travail sur lequel le travail ou le flot de travaux est soumis. S'il n'existe aucune correspondance entre le nom *nom_paramètre* défini et le nom des paramètres définis dans la base de données locale sur le poste de travail, une valeur *null* est renvoyée.

Autorisation

Vous devez avoir un accès ***display*** à la base de données des paramètres définis au niveau local. De plus, vous devez disposer du droit d'accès suivant :

build sur un fichier objet
Si vous utilisez l'option **-b** pour créer ou reconstruire la base de données des paramètres locaux.

delete Si vous utilisez l'option **-d** pour supprimer les définitions de paramètres locaux.

modify sur un fichier objet
Si vous utilisez l'option **-replace** pour ajouter ou modifier des définitions de paramètres.

Syntaxe

parms {[**-V** | **-u**] | **-build**}

parms {-replace | **-extract}** *file_name*

parms [**-d**]*nom_paramètre*

parms -c *valeur_nom_paramètre*

Arguments

-V Affiche la version de la commande et quitte l'application.

-u Affiche des informations sur la syntaxe de la commande et quitte l'application.

-build Crée la base de données des paramètres sur le poste de travail si elle n'existe pas. Reconstruit la base de données de paramètres si elle existe déjà en supprimant les enregistrements inutilisés et en évitant la fragmentation provenant de nombreuses additions et suppressions.

-extract
Extrait toutes les définitions de paramètres de la base de données locale et les stocke dans le fichier avec le nom *file_name*. Utilisez cette option pour exporter les définitions de paramètres locaux afin de les importer en tant que définitions de paramètres globaux dans la base de données des objets de planification à l'aide des commandes «add», à la page 316 ou «replace», à la page 356.

-replace
Ajoute dans la base de données locale de nouvelles définitions de paramètres stockés dans un fichier nommé *file_name* ou remplace ceux déjà existants. Utilisez cette option pour importer en tant que définition de paramètres locaux, les définitions de paramètres globaux contenues dans le fichier nommé *nom_fichier* et extraites de la base de données des objets de planification à l'aide de la commande «extract», à la page 330.

-d Supprime les paramètres avec le nom *parametername* de la base de données locale sur le poste de travail.

nom_paramètre
Indique le nom du paramètre dont la valeur est affichée. Indique le nom du paramètre dont la valeur est affichée. Lorsqu'il est utilisé avec l'argument **-d** il représente le nom du paramètre à supprimer.

-c *nom valeur*
Indique le nom et la valeur d'un paramètre. Ce nom peut comporter jusqu'à 16 caractères alphanumériques, incluant les tirets (-) et les traits de soulignement (_), et doit commencer par une lettre. La valeur peut comporter jusqu'à 72 caractères. La valeur doit être délimitée par des guillemets si elle contient des caractères spéciaux. Si le paramètre n'existe pas, il est ajouté à la base de données. S'il existe déjà, sa valeur est modifiée.

Commentaires

Lorsque vous exécutez la commande **parms** sur la ligne de commande sans aucun argument, vous êtes invité à entrer des noms et valeurs de paramètres.

L'utilisation de **parms** dans les définitions de travail et les fichiers de script de travail nécessite que le paramètre existe déjà au niveau local dans la base de données des paramètres sur le poste de travail.

Voici un exemple d'un paramètre local, `MYFILE`, dans une clause de dépendance de fichier :

```
schedule test_js
on everyday
opens "/usr/home/tws_99/'/usr/home/tws_99/bin/parms MYFILE'"
:
test_job
end
```

L'exemple suivant décrit la façon dont la variable ***var***, entourée par des accents circonflexes (^), est remplacée lorsque le travail est en cours. Si le travail est

soumis comme travail ad hoc, le paramètre ***var*** est étendu, à savoir remplacé par la valeur attribuée à ***var*** dans la base de données locale lors de la soumission, et non lors du lancement du travail.

Exemple de définition de travail UNIX :

```
DATA#UX_P_TEST DOCOMMAND "ls ^var^"
STREAMLOGON "mae82"
DESCRIPTION "Test parms in job definition on UNIX."
RECOVERY STOP
```

Exemple de définition de travail Windows :

```
BORG#WIN_P_TEST DOCOMMAND "dir ^var^"
STREAMLOGON "mae82"
DESCRIPTION "Test parms in job definition on Windows."
RECOVERY STOP
```

Lorsque le paramètre est utilisé dans un fichier de script de travail, il n'est pas étendu avant le lancement du script. Il n'est pas étendu lorsque le flot de travaux contenant le travail est traité par **JnextPlan**. Vous trouverez ci-dessous des exemples sur l'utilisation du paramètre ***var*** dans les fichiers de script de travail.

Exemple de script UNIX :

```
#!/bin/sh
TWS_HOME="/opt./tws/mae82/maestro"
export TWS_HOME
MDIR='$TWS_HOME/bin/parms var'
export MDIR
ls -l $MDIR
```

Exemple de script Windows :

```
set TWS_HOME=d:\win32app\TWS\mae82\maestro
echo %TWS_HOME%
FOR /F "Tokens=*" %%a in (%TWS_HOME%\bin\parms var) do set MDIR=%%a
echo %MDIR%
dir %MDIR%
```

Exemples

Pour renvoyer la valeur `monparm`, exécutez la commande suivante :

```
parms monparm
```

Pour modifier la valeur `monparm`, exécutez la commande suivante :

```
parms -c monparm "item 123"
```

Pour créer un nouveau paramètre appelé `sonparm`, exécutez la commande suivante :

```
parms -c sonparm
"item 789"
```

Pour modifier la valeur `monparm` et ajouter le paramètre `sonparm`, exécutez la commande suivante :

```
parms
Nom du paramètre ? monparm < Retour>
Valeur du paramètre ? "item 456" < Retour>
Nom du paramètre ? sonparm < Retour>
Valeur du paramètre ? "item 123" < Retour>
Nom du paramètre ? < Retour>
```

Pour plus d'informations, voir Chapitre 6, «Personnalisation de votre charge de travail à l'aide des tables de variables», à la page 121.

release

Libère les travaux et les flots de travaux à partir des dépendances **needs** sur une ressource. Cette commande doit être émise uniquement à partir d'un fichier de script de travail.

Syntaxe

release -V | -U

release
[**-s**]
[*poste_de_travail***#**]
nom_ressource
[*comptage*]

Arguments

-V Affiche la version de la commande et quitte l'application.

-U Affiche des informations sur la syntaxe de la commande et quitte l'application.

-s Libère la dépendance **needs** à partir de la ressource spécifiée uniquement au niveau du flot de travaux.

Si **-s** n'est pas utilisé, la dépendance **needs** de la ressource spécifiée est libérée au niveau du travail ou au niveau du flot de travaux **needs** est introuvable au niveau du travail.

*poste_de_travail***#**
Indique le nom du poste de travail ou de la classe de postes de travail sur laquelle la ressource est définie. Par défaut, il s'agit du poste de travail local.

nom_ressource
Indique le nom de la ressource impliquée dans la dépendance **needs**.

nombre Indique le nombre d'unités de la ressource à libérer.

Commentaires

Les unités d'une ressource sont acquises par un travail ou un flot de travaux lorsque celui-ci est lancé, et libérées automatiquement à la fin du travail ou du flot de travaux. La commande **release** peut être utilisée dans un script de travail pour libérer des ressources avant la fin du travail ou du flot de travaux ou pour libérer manuellement des travaux et des flots de travaux à partir des dépendances needs dans des situations d'urgence.

Exemples

Dans le flot de travaux suivant, deux unités de la ressource `dbase` sont requises par le flot de travaux `sked5` :

```
schedule ux1#sked5 on tu
nécessite 2 dbase :
job1
jobrel follows job1
job2 follows jobrel
end
```

Pour libérer la ressource `dbase` avant le début du travail `job2`, le fichier script de `jobrel` contient la commande suivante :

Sur les systèmes d'exploitation UNIX :

```
`maestro`/bin/release -s dbase
```

Sur les systèmes d'exploitation Windows :

```
<rép_base_TWS>\bin\release -s dbase
```

Remarque : L'argument **-s** peut être omis, dans la mesure où aucune ressource n'a été bloquée au niveau du travail.

rmstdlist

Supprime ou affiche des listes standard en fonction de l'ancienneté du fichier. Cette fonctionnalité doit être utilisée par l'administrateur Tivoli Workload Scheduler pour gérer l'environnement de planification.

Syntaxe

rmstdlist -V | -U

rmstdlist [**-p**] [*age*]

Arguments

-V Affiche la version de la commande et quitte l'application.

-U Affiche des informations sur la syntaxe de la commande et quitte l'application.

-p Affiche les noms des répertoires des listes standard éligibles. Aucun répertoire ou fichier n'est supprimé. Si **-p** n'est pas indiqué, les listes standard éligibles sont supprimées.

age Age minimum, en jours, des répertoires de listes standard à afficher ou à supprimer. La valeur par défaut est 10 jours.

Remarque : Du fait que la liste des répertoires et des fichiers affichés ou supprimés à l'aide de **rmstdlist** est générée en fonction de leur date de dernier accès, les dates indiquées dans la liste des répertoires peuvent être différentes des dates affichées dans la liste des fichiers.

Syntaxe

A titre de règle, vous devez régulièrement supprimer les fichiers de la liste standard, tous les 10 à 20 jours environ. Des retards plus importants peuvent être plus difficiles à gérer et si le nombre de fichiers devient trop important vous pouvez avoir à effacer certains d'entre eux manuellement avant de pouvoir utiliser à nouveau **rmstdlist**.

Ce problème peut se produire sur les systèmes AIX particulièrement à cause d'une limitation actuellement non résolue avec la commande `rm -rf`. Si **rmstdlist** échoue

à cause de cette limitation, la commande n'affiche pas d'erreur autre que le code de sortie 126. Si vous souhaitez que l'erreur `rm-rf` soit affichée, vous pouvez modifier le script `rmstdlist` de la façon suivante :

1. Localisez le script dans le répertoire *TWS_home*/bin
2. Trouvez la ligne :
   ```
   rm -rf `cat /tmp/rm$$` 2> /dev/null
   ```
3. Supprimez la redirection /dev/null de façon à ce que la ligne devienne :
   ```
   rm -rf `cat /tmp/rm$$`
   ```

Exemples

Pour afficher le nom des répertoires de fichiers de liste standard qui ont plus de 14 jours, exécutez la commande suivante :

```
rmstdlist -p 14
```

Pour supprimer tous les fichiers de liste standard (et leurs répertoires) qui ont plus de sept jours, exécutez la commande suivante :

```
rmstdlist 7
```

sendevent

La commande envoie un les événements personnalisés définis par la commande evtdef vers le serveur de processeur d'événements actuellement actif dans le plan de production. Lors de la réception des événements par le processeur d'événements, ils déclenchent les règles d'événement dans lesquelles ils ont été spécifiés.

Les utilisateurs peuvent ignorer le serveur de destination par défaut (défini par les options globales) en spécifiant l'hôte et le port d'un nouveau serveur.

Syntaxe

sendevent -V | ? | -help | -u | -usage

sendevent [**-hostname** *nom_hôte*]
[{**-port | -sslport**} *port*]
type_événement
source
[[*attribute=valeur*]...]

Arguments

-V Affiche la version de la commande et quitte l'application.

? | -help | -u | -usage
Affiche des informations sur la syntaxe de la commande et quitte l'application.

-hostname *nom_hôte*
Indique le nom d'hôte d'un serveur de processeur d'événement autre que celui actuellement actif. Ce paramètre est obligatoire si la commande est lancée à partir d'un client de ligne de commande.

-port | -sslport} *port*
Indique le numéro de port d'un serveur de processeur d'événements autre que celui actuellement actif. **-sslport** définit le port utilisé pour détecter

les connections SSL entrantes. Ce paramètre est obligatoire si la commande est lancée à partir d'un client de ligne de commande.

type_événement
L'un des types d'événements personnalisés défini par la commande evtdef dans le fournisseur d'événements génériques et spécifié en tant qu'événement déclencheur dans une définition de règle d'événement.

source Nom du fournisseur d'événements que vous avez personnalisé avec evtdef. Il s'agit également du nom que vous devez indiquer comme argument du mot clé `eventProvider` dans la définition des règles d'événement déclenchées par ces événements personnalisés.

Le nom par défaut est `GenericEventPlugIn`.

attribut=*valeur*
L'un ou plusieurs des attributs qualifiant le type d'événement personnalisé qui sont spécifiés en tant qu'attributs d'événement déclencheur pour la règle d'événement.

Commentaires

Cette commande peut également être exécutée sur les systèmes hébergeant uniquement le client de ligne de commande Tivoli Workload Scheduler.

Exemples

Dans cet exemple, une application envoie le type d'événement personnalisé `BusProcCompleted` vers un serveur de traitement d'événements alternatif nommé `master3`. L'événement est que le traitement du fichier `calcweek` est terminé.

```
sendevent -hostname master3 -port 4294 BusProcCompleted
GenericEventPlugIn TransacName=calcweek Workstation=ab5supp
```

Le nom de fichier et du poste de travail associé sont les deux attributs d'événement `BusProcCompleted` qui ont été spécifiés en tant qu'attributs d'événement déclencheur dans une règle d'événement associée.

showexec

Affiche le statut des travaux qui sont en cours d'exécution. Cette commande s'applique uniquement à UNIX. Elle est destinée aux agents standard. Pour les gestionnaires de domaine et les agents tolérants aux pannes, utilisez plutôt la commande **conman showjobs**.

Syntaxe

showexec [**-V** | **-U** | **INFO**]

Arguments

-V Affiche la version de la commande et quitte l'application.

-U Affiche des informations sur la syntaxe de la commande et quitte l'application.

INFO Affiche le nom du fichier de travail au lieu de l'utilisateur, de la date et de l'heure.

Résultats

La sortie de la commande est disponible dans deux formats : **standard** et **INFO**.

Exemples

Pour afficher les travaux en cours d'exécution au format **standard**, exécutez la commande suivante :

```
showexec
```

Pour afficher les travaux en cours d'exécution au format **INFO**, exécutez la commande suivante :

```
showexec INFO
```

Format standard

CPU Poste de travail sur lequel le travail s'exécute.

Schedule
Nom du flot de travaux dans lequel le travail est exécuté.

Job Nom du travail.

Job# Numéro du travail.

User Nom d'utilisateur du travail.

Start Date
Date de début d'exécution du travail.

Start Time
Heure de début d'exécution du travail.

(Est) Elapse
Durée d'exécution estimée du travail (en minutes).

Format info

CPU Poste de travail sur lequel le travail s'exécute.

Schedule
Nom du flot de travaux dans lequel le travail est exécuté.

Job Nom du travail.

Job# Numéro du travail.

JCL Nom de fichier du travail.

shutdown

Arrête les processus Tivoli Workload Scheduler et, éventuellement, WebSphere Application Server. Seuls les postes de travail Windows sont concernés. Vous devez avoir un accès ***shutdown*** au poste de travail.

Syntaxe

shutdown [-V | -U] [-appsrv]

Arguments

-V Affiche la version de la commande et quitte l'application.

-U Affiche des informations sur la syntaxe de la commande et quitte l'application.

-appsrv
Arrête également WebSphere Application Server.

Commentaires

Vérifiez que le *TWS_user* que vous utilisez appartient au groupe Administrators défini sur le poste de travail Windows.

Exemples

Pour afficher le nom et la version de la commande, exécutez la commande suivante :

```
shutdown -V
```

Pour arrêter les processus Tivoli Workload Scheduler et WebSphere Application Server, exécutez la commande suivante :

```
shutdown -appsrv
```

ShutDownLwa

Arrête l'agent. Aucun accès spécifique au poste de travail n'est requis. Exécutez cette commande en local sur l'agent que vous souhaitez arrêter.

Syntaxe

ShutDownLwa

Arguments

Aucun argument n'est nécessaire.

Exemples

Pour arrêter l'agent, exécutez la commande suivante :

```
ShutDownLwa
```

StartUp

Démarre le processus **netman** de gestion du réseau Tivoli Workload Scheduler.

Vous devez avoir un accès ***start*** au poste de travail.

Syntaxe

StartUp [-V | -U]

Arguments

-V Affiche la version de la commande et quitte l'application.

-U Affiche des informations sur la syntaxe de la commande et quitte l'application.

Commentaires

Sous Windows, le service **netman** est lancé automatiquement au redémarrage de l'ordinateur. **StartUp** peut être utilisé pour redémarrer le service si celui-ci est arrêté pour une raison quelconque.

Sous UNIX, la commande **StartUp** peut être exécutée automatiquement en l'appelant à partir du fichier `/etc/inittab`, de façon à ce que l'infrastructure WebSphere Application Server et **netman** soient démarrés à chaque fois qu'un ordinateur est redémarré. **StartUp** peut être utilisé pour redémarrer le processus **netman** si celui-ci est arrêté pour une quelconque raison.

Le reste de l'arborescence des processus peut être redémarré avec les commandes

```
conman start
conman startmon
```

. Pour plus d'informations, voir «start», à la page 479.

Exemples

Pour afficher le nom et la version de la commande, exécutez la commande suivante :

```
StartUp -V
```

Pour lancer le processus **netman**, exécutez la commande suivante :

```
StartUp
```

StartUpLwa

Démarre l'agent.

Aucun accès spécifique au poste de travail n'est requis. Exécutez cette commande en local sur l'agent que vous souhaitez arrêter.

Syntaxe

StartUpLwa

Arguments

Aucun argument n'est nécessaire.

Exemples

Pour démarrer l'agent, exécutez la commande suivante :

```
StartUpLwa
```

tws_inst_pull_info

Il s'agit d'un script qui génère des informations sur votre environnement Tivoli Workload Scheduler et votre poste de travail local et peut prendre des clichés des données DB2 et WebSphere Application Server sur le gestionnaire de domaine maître, en les enregistrant en tant que module daté.

Il peut également générer un rapport contenant non seulement les résultats de la capture d'écran mais également de nombreux paramètres de configuration et

d'environnement. Cet outil s'avère très utile pour décrire un problème au service de support logiciel IBM. Pour de meilleurs résultats, il doit être exécuté dès que le problème est identifié.

Commentaires

Pour plus d'informations sur cette commande, voir *IBM Tivoli Workload Scheduler Guide d'identification et de résolution des problèmes*.

version

Affiche des informations sur la version de Tivoli Workload Scheduler actuellement installée sur le système. Cette commande s'applique uniquement à UNIX. Les informations sont extraites d'un fichier de version.

Syntaxe

version -V | -u | -h

version [**-a**] [**-f** *fichierv*] [*fichier* [...]]

Arguments

-V Affiche la version de la commande et quitte l'application.

-u Affiche des informations sur la syntaxe de la commande et quitte l'application.

-h Affiche des informations d'aide sur la commande et quitte l'application.

-a Affiche des informations relatives aux fichiers produit. Par défaut, seules les informations relatives aux fichiers indiqués s'affichent.

-f *fichierv*
Indique le chemin d'accès et le nom du fichier de version s'il est différent de la valeur par défaut. Il s'agit par défaut d'un fichier intitulé **version.info** dans le répertoire de travail en cours.

fichier Indique les noms des fichiers produit, séparés par des espaces, pour lesquels des informations de version sont affichées. Par défaut, aucune information relative aux fichiers ne s'affiche ou, si **-a** est utilisé, toutes les informations relatives aux fichiers s'affichent.

Résultats

L'en-tête de la sortie contient le nom, la version, le système d'exploitation, le niveau de modification et la date d'installation du produit. Le reste de l'écran répertorie des informations relatives au(x) fichier(s) indiqué(s). Les fichiers sont répertoriés au format suivant :

Fichier
Nom du fichier.

Révision
Numéro de révision du fichier.

Modification
Niveau de modification du fichier, s'il y a lieu.

Taille (octets)
Taille du fichier, en octets.

Total de contrôle
Total de contrôle du fichier. Le total de contrôle est calculé via la commande UNIX **sum**. Sous AIX, la commande **sum** est associée à l'argument **-o**.

Commentaires

Les informations relatives au fichier Tivoli Workload Scheduler se trouvent dans le fichier `version.info`. Ce dernier est placé dans le répertoire *racine_TWS*`version` lors de l'installation. Le fichier `version.info` est dans un format spécifique et n'est pas modifié.

Vous pouvez transférer le fichier `version.info` vers un autre répertoire. Toutefois, vous devez ensuite inclure l'argument **-f** pour rechercher le fichier.

Exemples

Pour afficher les informations sur la version de Tivoli Workload Scheduler installée, exécutez la commande suivante :

```
./version
```

Voici un exemple de sortie de cette commande :

```
IBM Tivoli Workload Scheduler/VERSION 9.21 (C) Copyright IBM Corp 1998, 2013

IBM Tivoli Workload Scheduler 9.2 UNIX
```

Pour afficher les informations relatives à tous les fichiers, exécutez la commande suivante :

```
version/version -a -f version/version.info
```

Pour afficher les informations relatives au fichier `customize`, exécutez la commande suivante :

```
cd version
./version customize
```

Pour afficher les informations relatives au fichier `customize` (si `version.info` se trouve dans le répertoire `/apps/maestro`), exécutez la commande suivante :

```
cd version
./version -f /apps/maestro/version.info customize
```

Commandes non prises en charge

Les commandes non prises en charge suivantes proposent des fonctions sous Windows, qui sont similaires aux commandes UNIX **ps** et **kill**. Ils peuvent être utilisés si des utilitaires Windows similaires ne sont pas disponibles.

Syntaxe

listproc

killproc *pid*

Commentaires

listproc
Affiche la liste des processus du système sous forme d'une table.

killproc
Supprime le processus dont l'ID est *pid*.

Remarque : Exécutée par l'administrateur, la commande **killproc** permet de supprimer les processus système.

Chapitre 14. Utilisation des commandes d'utilitaire dans l'environnement dynamique

Le présent chapitre décrit les commandes d'utilitaire Tivoli Workload Scheduler pour l'environnement dynamique. Alors que certaines commandes s'exécutent sur le gestionnaire de domaine maître ou sur un gestionnaire de domaine dynamique, d'autres s'exécutent sur les agents. Elles sont installées sur le chemin *racine_TWA*/TDWB/bin et s'exécutent avec les systèmes d'exploitation UNIX et Windows. Les commandes d'utilitaire sont exécutées à partir de l'invite de commande du système d'exploitation, sauf l'utilitaire **jobprop** qui s'utilise uniquement dans une définition de travail (voir la section «Transmission d'un jeu de variables d'un travail à un autre dans la même instance de flot de travaux à l'aide de l'utilitaire **jobprop**», à la page 530).

Le tableau 80 contient la liste des commandes d'utilitaire avec, pour chacune d'entre elles, la description et le type de poste de travail où vous pouvez l'exécuter.

Tableau 80. Liste des commandes d'utilitaire pour les postes de travail dynamiques

Commande	Description	Type de poste de travail
exportserverdata	Télécharge la liste des instances de courtier de charge de travail à partir de la base de données Tivoli Workload Scheduler et modifie un numéro de port ou un nom d'hôte.	gestionnaire de domaine maître ou gestionnaire de domaine dynamique
importserverdata	Télécharge la liste des instances de Dynamic Workload Broker dans la base de données Tivoli Workload Scheduler après l'édition du fichier temporaire pour changer un numéro de port ou un nom d'hôte.	gestionnaire de domaine maître ou gestionnaire de domaine dynamique
jobprop	Définit les valeurs de variables dans un travail à transmettre au travail successif dans la même instance de flot de travaux. Pour plus d'informations sur l'utilisation de cet utilitaire dans une définition de travail, voir «Transmission d'un jeu de variables d'un travail à un autre dans la même instance de flot de travaux à l'aide de l'utilitaire **jobprop**», à la page 530. Cet utilitaire est installé dans le répertoire <REP_INSTALLATION_TWS>/TWS/bin et s'exécute sur les systèmes d'exploitation UNIX et Windows.	agents
movehistorydata	Déplace les données présentes dans la base de données Tivoli Workload Scheduler vers les tables d'archivage	gestionnaire de domaine maître ou gestionnaire de domaine dynamique
param	Crée, affiche et supprime des variables et des mots de passe utilisateur sur des agents dynamiques.	agents

Tableau 80. Liste des commandes d'utilitaire pour les postes de travail dynamiques (suite)

Commande	Description	Type de poste de travail
resource	Crée, modifie, associe, interroge ou définit des ressources en ligne ou hors ligne.	gestionnaire de domaine maître, gestionnaire de domaine dynamique ou agents
sendevent	Envoie des événements génériques au serveur de processeur d'événement actif.	gestionnaire de domaine dynamique et agents
twstrace	Lors de l'exécution, modifie les paramètres pour la fonction de trace sur les agents	agents

Remarque : Pour supprimer les fichiers des journaux de travaux pour les postes de travail des agents dynamiques, définissez la valeur de la propriété `MaxAge` dans `JobManager.ini`. Pour plus de détails, voir les manuels d'*IBM Tivoli Workload Scheduler : Guide d'administration - Configuration de l'agent - Configuration des propriétés communes aux lanceurs [Lanceurs]*.

Fichier de configuration de ligne de commande

Le fichier `CLIConfig.properties` contient les informations de configuration utilisées lors de la saisie de commandes. Par défaut, les arguments requis lors de la saisie des commandes sont extraits de ce fichier, à moins qu'ils ne soient explicitement spécifiés dans la syntaxe des commandes.

Le fichier `CLIConfig.properties` est créé au moment de l'installation et situé sur le gestionnaire de domaine maître dans le chemin d'accès suivant :
TWA_home`/TDWB/config`

Le fichier `CLIConfig.properties` contient l'ensemble de paramètres suivants :

Propriétés par défaut Dynamic Workload Broker

ITDWBServerHost
Spécifie l'adresse IP de Dynamic Workload Broker.

ITDWBServerPort
Définit le numéro de port du Dynamic Workload Broker. La valeur par défaut est **9550**.

ITDWBServerSecurePort
Définit le numéro de port du Dynamic Workload Broker lorsque la sécurité est activée. La valeur par défaut est **9551**.

use_secure_connection
Indique si une connexion sécurisée doit être utilisée. La valeur par défaut est **false**.

Nom de fichier et chemin d'accès du fichier de clés et de clés certifiées

Fichier de clés
Indique le nom et le chemin d'accès du fichier de clés contenant les clés privées. Un fichier de clés contient des clés publiques et des clés privées. Les clés publiques sont stockées comme des certificats de signataire tandis que les clés privées sont stockées dans des certificats personnels. La valeur par défaut est /Certs/TDWBClientKeyFile.jks.

Fichier de clés certifiées
Indique le nom et le chemin d'accès du fichier de clés certifiées contenant les clés publiques. Un fichier de clés certifiées est un fichier de la base de données de clés qui contient des clés publiques. La clé publique est stockée comme un certificat de signataire. Les clés sont utilisées à des fins diverses, y compris pour l'authentification et l'intégrité des données. La valeur par défaut est /Certs/TDWBClientTrustFile.jks.

Mots de passe des fichiers de clés et des fichiers de clés certifiées

keyStorepwd
Définit le mot de passe du fichier de clés.

trustStorepwd
Définit le mot de passe du fichier de clés certifiées.

Types de fichier pour les fichiers de clés et les fichiers de clés certifiées

keyStoreType
Définit le type de fichier pour le fichier de clés certifiées. La valeur par défaut est JKS.

trustStoreType
Définit le type de fichier pour le fichier de clés certifiées. La valeur par défaut est JKS.

ID utilisateur et mot de passe pour Dynamic Workload Broker

tdwb_user
Désigne le nom d'un utilisateur autorisé à effectuer des opérations sur Dynamic Workload Broker lorsque la sécurité est activée. La valeur par défaut est **ibmschedcli**. Ce mot de passe doit être défini auparavant sur IBM WebSphere. Pour plus d'informations sur la sécurité, voir *Tivoli Workload Scheduler - Guide d'administration*, *SC11-6396*.

tdwb_pwd
Indique le mot de passe d'un utilisateur autorisé à effectuer des opérations sur Dynamic Workload Broker lorsque la sécurité est activée. Ce mot de passe doit être défini auparavant sur IBM WebSphere. Pour plus d'informations sur la sécurité, voir *Tivoli Workload Scheduler - Guide d'administration*.

Niveau de détail du journal de ligne de commande et des informations de trace

logger.Level
Indique le niveau de détail des fichiers de trace et des fichiers journaux de ligne de commande. Les fichiers de trace et les fichiers journaux de ligne de commande sont créés aux emplacements suivants :

fichier journal
TWA_home/TDWB/logs/Msg_cli.log.log

Fichier de trace
TWA_home/TDWB/logs/Trace_cli.log

La valeur par défaut est INFO.

logger.consoleLevel
Indique le niveau de détail pour les informations de journal et de trace à diriger vers une sortie standard. La valeur par défaut est

FINE. Les valeurs prises en charge pour les paramètres **consoleLevel** et **loggerLevel** sont les suivantes :

TOUS Indique que tous les messages sont consignés.

SEVERE
Indique que les messages d'erreur grave sont consignés.

AVERTISSEMENT
Indique que les messages d'avertissement sont consignés.

INFO Indique que les messages d'information sont consignés.

CONFIG
Indique que les messages de configuration statique sont consignés.

FINE Indique que les informations de la fonction de trace sont consignées.

FINER
Indique que les informations détaillées de la fonction de trace sont consignées.

FINEST
Indique que les informations très détaillées de la fonction de trace sont consignées.

OFF Indique que la fonction de consignation est désactivée.

logger.limit
Indique la taille maximale, en octets, d'un fichier journal. La valeur par défaut est 400000. Lorsque la taille maximale est atteinte, un nouveau fichier est créé jusqu'à ce que le nombre maximal de fichiers soit atteint. Lorsque tous les fichiers atteignent la taille maximale et que le nombre maximal de fichier est dépassé, le fichier le plus ancien est réécrit.

logger.count
Indique le nombre maximal de fichiers journaux. La valeur par défaut est 6. Lorsque le nombre maximal est atteint, un nouveau fichier est créé jusqu'à de que le nombre maximal de fichiers soit atteint. Lorsque tous les fichiers atteignent la taille maximale et que le nombre maximal de fichier est dépassé, le fichier le plus ancien est réécrit. Lorsqu'un fichier est créé, le suffixe 0 est ajouté après l'extension de fichier. Le fichier qui comporte le suffixe 0 est toujours le fichier actuel. Tous les fichiers plus anciens sont renumérotés.

java.util.logging.FileHandler.pattern
Indique que les informations de trace pour la machine virtuelle Java sont consignées dans le fichier Trace_cli.log. La valeur par défaut est INFO.

java.util.logging.FileHandler.limit
Indique la taille maximale, en octets, d'un fichier de trace. La valeur par défaut est 400000. Lorsque la taille maximale est atteinte, un nouveau fichier est créé jusqu'à ce que le nombre maximal de fichiers soit atteint. Lorsque tous les fichiers atteignent la taille maximale et que le nombre maximal de fichier est dépassé, le fichier le plus ancien est réécrit.

java.util.logging.FileHandler.count
Indique le nombre maximal de fichiers de trace. La valeur par défaut est 6. Lorsque le nombre maximal est atteint, un nouveau fichier est créé jusqu'à de que le nombre maximal de fichiers soit atteint. Lorsque tous les fichiers atteignent la taille maximale et que le nombre maximal de fichier est dépassé, le fichier le plus ancien est réécrit. Lorsqu'un fichier est créé, le suffixe 0 est ajouté après l'extension de fichier. Le fichier qui comporte le suffixe 0 est toujours le fichier actuel. Tous les fichiers plus anciens sont renumérotés.

java.util.logging.FileHandler.formatter
Indique le composant de formatage à utiliser pour le fichier Trace_cli.log. La valeur par défaut est `com.ibm.logging.icl.jsr47.CBEFormatter`.

Configuration DAO commune
Cette section définit les paramètres du système SGBDR pour les commandes **exportserverdata**, **importserverdata** et **movehistorydata**. Ces commandes utilisent le système SGBDR installé sur Dynamic Workload Broker. Les valeurs de ces paramètres sont renseignées lors de l'installation et ne doivent pas être modifiées, à l'exception de `com.ibm.tdwb.dao.rdbms.useSSLConnections` comme indiqué ci-après.

com.ibm.tdwb.dao.rdbms.rdbmsName
Indique le nom du système SGBDR.

com.ibm.tdwb.dao.rdbms.useDataSource
Indique la source de données à utiliser.

com.ibm.tdwb.dao.rdbms.jdbcPath
Indique le chemin d'accès pour le pilote JDBC.

com.ibm.tdwb.dao.rdbms.jdbcDriver
Indique le pilote JDBC.

com.ibm.tdwb.dao.rdbms.userName
Indique le nom de l'utilisateur du système SGBDR.

com.ibm.tdwb.dao.rdbms.password
Indique le mot de passe de l'utilisateur du système SGBDR.

com.ibm.tdwb.dao.rdbms.useSSLConnections
Indique que l'accès à la base de données DB2 Tivoli Workload Scheduler par certaines des commandes CLI 'effectue sur SSL. Paramétré sur `FALSE` par défaut. Vous devez le définir sur `TRUE` si la base de données est DB2 et que vous utilisez la sécurité FIPS pour exécuter les commandes suivantes :

- **`exportserverdata`**
- **`importserverdata`**
- **`movehistorydata`**

exportserverdata

Utiliser la commande **exportserverdata** pour télécharger (exporter) la liste des instance Dynamic Workload Broker à partir de la base de données Tivoli Workload Scheduler et modifier un numéro de port ou un nom d'hôte.

Syntaxe

exportserverdata ?

exportserverdata -dbUsr *db_user_name* **-dbPwd** *db_user_password* **-exportFile** *filename*

Description

Cette commande extrait une liste des identificateurs URI (Uniform Resource Identifier) de tous les serveurs instance Dynamic Workload Broker à partir de la base de données Tivoli Workload Scheduler et les copie dans un fichier temporaire de sorte que, si le nom d'hôte ou le numéro de port de n'importe quelle instance répertoriée est modifié, l'administrateur peut enregistrer ces informations dans le fichier et les replacer dans la base de données avec la commande `importserverdata`. Par défaut, la liste des identificateurs URI est sauvegardée dans le fichier server.properties, situé dans le répertoire en cours.

Cette action est nécessaire car la liste des serveurs instance Dynamic Workload Broker doit être tenue à jour et disponible à tout moment, puisque les agents Resource Advisor se connectent périodiquement à l'instance active pour envoyer leurs données sur les ressources détectées sur chaque ordinateur. Ils peuvent passer automatiquement d'une instance de cette liste à l'autre et trouver celle qui est active pour copier ces données dans son référentiel de ressources. Etant donné que le gestionnaire de domaine maître et que chaque maître de sauvegarde sont installés avec un instance Dynamic Workload Broker, le instance Dynamic Workload Broker actif s'exécute dans le gestionnaire de domaine maître pendant qu'une instance de veille réside sur chaque maître de sauvegarde.

L'URI pointant vers chaque instance Dynamic Workload Broker est le suivant :
`https://`*`nom_hôte`*`:`*`numéro_port`*`/JobManagerRESTWeb/JobScheduler`

Vous ne pouvez modifier que le nom d'hôte et le numéro de port.

Important : La liste est ordonnée. Vous pouvez modifier l'ordre des instances tel qu'elles apparaissent dans cette liste et les agents suivront cet ordre. Si vous disposez de plusieurs maîtres de sauvegarde et que vous décidez de suivre un ordre de basculement spécifique en cas d'échec d'un maître, vous pouvez demander aux agents de permuter sur l'instance correcte à l'aide de cette liste ordonnée, ce qui accélère le temps de transition.

Si votre base de données Tivoli Workload Scheduler est DB2 et que vous utilisez la sécurité FIPS, vous devez définir l'option **com.ibm.tdwb.dao.rdbms.useSSLConnections** sur `TRUE` dans le fichier `CLIConfig.properties` pour exécuter cette commande avec succès.

Options

? Affiche des informations d'aide.

-dbUsr *db_user_name*
Nom d'utilisateur nécessaire pour accéder au serveur de base de données Tivoli Workload Scheduler.

-dbPwd *db_user_password*
Mot de passe utilisateur nécessaire pour accéder au serveur de base de données Tivoli Workload Scheduler.

-exportFile *filename*
Nom du fichier temporaire dans lequel les identificateurs URI extraits à partir de la base de données sont copiés pour édition. Ce fichier texte est créé lorsque vous exécutez la commande et vous pouvez l'ouvrir avec n'importe quel éditeur pour modifier le nom d'hôte ou le numéro de port. Si vous n'indiquez pas de chemin, le fichier est créé dans le répertoire où figure la commande :

```
<TWA_home>/TDWB/bin
```

Dans la pratique, si vous spécifiez un chemin d'accès différent, assurez-vous que le chemin existe avant d'exécuter cette commande.

Exemple

Pour télécharger la liste actualisée de tous les serveurs instance Dynamic Workload Broker (serveurs actifs et serveurs de sauvegarde) et les copier dans un fichier intitulé `c:\myservers\uris160709`, exécutez :

```
exportserverdata -dbUsr twsadm -dbPwd fprefect -exportFile c:\myservers\uris160709
```

La commande renvoie le fichier `uris160709` qui se présente comme suit :

```
https://accrec015:42127/JobManagerRESTWeb/JobScheduler
https://prodop099:52529/JobManagerRESTWeb/JobScheduler
https://prodop111:31116/JobManagerRESTWeb/JobScheduler
```

`prodop099` est le serveur instance Dynamic Workload Broker actif car il est hébergé par le gestionnaire de domaine maître actuellement actif, tandis que `accrec015` et `prodop111` sont en veille car ils sont hébergés par des maîtres de sauvegarde.

Vous pouvez éditer ce fichier pour appliquer vos modifications avant d'utiliser la commande `importserverdata` pour envoyer à nouveau les identificateurs URI à la base de données.

Voir aussi

«importserverdata»

importserverdata

Utiliser la commande **importserverdata** pour importer la liste des serveurs instance Dynamic Workload Broker dans la base de données Tivoli Workload Scheduler après l'édition du fichier temporaire pour modifier un numéro de port ou un nom d'hôte.

Syntaxe

importserverdata ?

importserverdata -dbUsr *db_user_name* **-dbPwd** *db_user_password* **-importFile** *filename*

Description

Cette commande replace la liste des serveurs instance Dynamic Workload Broker dans la base de données Tivoli Workload Scheduler à partir du fichier temporaire dans lequel ils ont été téléchargés à l'aide de la commande `exportserverdata`.

Utilisez les commandes `exportserverdata` et `importserverdata` si vous devez enregistrer des modifications de nom d'hôte ou de numéro de port dans les identificateurs URI des instances. Il est nécessaire de disposer de la liste actualisée des serveurs instance Dynamic Workload Broker à tout moment, car les agents Resource Advisor se connectent périodiquement à l'instance active pour envoyer leurs données sur les ressources identifiées sur chaque ordinateur. Ils peuvent passer automatiquement d'une instance de cette liste à l'autre et trouver celle qui est active pour copier ces données dans son référentiel de ressources. Etant donné que le gestionnaire de domaine maître et que chaque maître de sauvegarde sont installés avec un instance Dynamic Workload Broker, le instance Dynamic Workload Broker actif s'exécute dans le gestionnaire de domaine maître pendant qu'une instance de veille réside sur chaque maître de sauvegarde.

Important : La liste est ordonnée. Vous pouvez modifier l'ordre des instances tel qu'elles apparaissent dans cette liste et les agents suivront cet ordre. Si vous disposez de plusieurs maîtres de sauvegarde et que vous décidez de suivre un ordre de basculement spécifique en cas d'échec d'un maître, vous pouvez demander aux agents de permuter sur l'instance correcte à l'aide de cette liste ordonnée, ce qui accélère le temps de transition.

Si votre base de données Tivoli Workload Scheduler est DB2 et que vous utilisez la sécurité FIPS, vous devez définir l'option **com.ibm.tdwb.dao.rdbms.useSSLConnections** sur `TRUE` dans le fichier `CLIConfig.properties` pour exécuter cette commande avec succès.

Options

? Affiche des informations d'aide.

-dbUsr *db_user_name*
Nom d'utilisateur nécessaire pour accéder au serveur de base de données Tivoli Workload Scheduler.

-dbPwd *db_user_password*
Mot de passe utilisateur nécessaire pour accéder au serveur de base de données Tivoli Workload Scheduler.

-importFile f*filename*
Nom du fichier temporaire que vous spécifiez avec le mot-clé `-exportFile` dans la commande `exportserverdata`.

Exemple

Pour importer la liste modifiée des identificateurs URI instance Dynamic Workload Broker à partir du fichier `c:\myservers\uris160709` vers la base de données Tivoli Workload Scheduler :

```
importserverdata -dbUsr twsadm -dbPwd fprefect -importFile c:\myservers\uris160709
```

Voir aussi

«exportserverdata», à la page 585

jobprop

Utilisez la commande **jobprop** sur une définition de travail afin de définir des variables localement pour un travail sur des agents dynamiques et agents Tivoli Workload Scheduler for z/OS.

Vous pouvez utiliser cette commande sur un travail natif ou exécutable afin de définir la valeur de variable pouvant être transférée dans un travail successif du même flot de travaux. Les valeurs sont définies lors de l'exécution.

Syntaxe

jobprop *variable valeur*

Arguments

Variable
Nom de variable.

valeur Valeur de *variable*.

Commentaires

Les noms de variable sont sensibles à la casse.

Exemples

Sur les systèmes d'exploitation UNIX, l'utilitaire **jobprop** définit les variables suivantes dans le travail exécutable NC125133#JOBA :

- Variable *VAR1* définie sur la valeur value1.
- Variable *VAR2* définie sur la valeur value2.
- Variable *VAR3* définie sur la valeur value3.
- Variable *VAR4* définie sur la valeur value4.

```
NC125133#JOBA
 TASK
    <?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl" xmlns:
jsdle="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdle">
  <jsdl:application name="executable">
    <jsdle:executable interactive="false">
    <jsdle:script>#!/bin/sh
. /home/ITAuser/TWA/TWS/tws_env.sh
jobprop VAR1 value1
jobprop VAR2 value2
jobprop VAR3 value3
jobprop VAR4 value4
</jsdle:script>
    </jsdle:executable>
  </jsdl:application>
</jsdl:jobDefinition>
 DESCRIPTION "Sample Job Definition"
 RCCONDSUCC "RC>=0"
 RECOVERY STOP
```

Sur les systèmes d'exploitation Windows, l'utilitaire **jobprop** définit les variables suivantes dans le travail exécutable WIN1#JOB1 :

- Variable *var1* définie sur la valeur value1.
- Variable *var2* définie sur la valeur value2.
- Variable *var3* définie sur la valeur value3.
- Variable *var4* définie sur la valeur value4.

```
WIN1#JOB1
 TASK
    <?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl" xmlns:
```

```
jsdle="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdle">
  <jsdl:application name="executable">
    <jsdle:executable interactive="false">
    <jsdle:script>
call C:\Progra~1\IBM\TWA\TWS\tws_env.cmd
jobprop var1 value1
jobprop var2 value2
jobprop var3 value3
jobprop var4 value4
</jsdle:script>
    </jsdle:executable>
  </jsdl:application>
</jsdl:jobDefinition>
 DESCRIPTION "Sample Job Definition"
 RCCONDSUCC "RC>=0"
 RECOVERY STOP
```

Remarque : Avant d'exécuter l'utilitaire **jobprop** dans la définition de travail, veillez à exécuter la commande **tws_env.cmd** avec la syntaxe correcte `call <REP_INST_TWS>\TWS\tws_env.cmd` où *<REP_INST_TWS>* désigne le répertoire d'installation de Tivoli Workload Scheduler.

movehistorydata

Utilisez la commande **movehistorydata** lorsque l'accès à la base de données devient trop lent. Cette commande déplace les données du référentiel de travaux vers les tables d'archive

La lenteur de l'accès à la base de données peut être due à un grand nombre d'enregistrements dans celle-ci, par exemple lorsque des soumissions de travaux groupées sont réalisées.

Lorsque vous exécutez cette commande, les travaux sont déplacés dans les tables suivantes de la base de données :

JOA_JOB_ARCHIVES
: Contient les instances de travail archivées

JRA_JOB_RESOURCE_ARCHIVES
: Contient des informations de ressource liées aux travaux

MEA_METRIC_ARCHIVES
: Contient des mesures collectées pour les travaux

Pour plus d'informations sur les tables d'historique, voir *Tivoli Workload Scheduler - Guide d'administration*, *SC11-6396*.

Remarque : Selon le nombre de travaux et d'accès à la base de données, une opération de nettoyage peut provoquer des pics d'utilisation de la mémoire ou de l'UC.

Si votre base de données Tivoli Workload Scheduler est DB2 et que vous utilisez la sécurité FIPS, vous devez définir l'option **com.ibm.tdwb.dao.rdbms.useSSLConnections** sur `TRUE` dans le fichier `CLIConfig.properties` pour exécuter cette commande avec succès.

Syntaxe

movehistorydata ?

movehistorydata -dbUsr *db_user_name***-dbPwd** *db_user_password*
[**-successfulJobsMaxAge** *successfulJobsMaxAge* [**-notSuccessfulJobsMaxAge** *notSuccessfulJobsMaxAge*][**-archivedJobsMaxAge** *archivedJobsMaxAge*]]

Description

Cette commande effectue un nettoyage dans la base de données du référentiel de travaux. En fonction des valeurs que vous indiquez, les informations sur les travaux soumis sont déplacées vers la base de données d'archive, puis les informations de cette base de données sont supprimées.

Utilisez cette commande pour substituer temporairement les paramètres définis dans le fichier **JobDispatcherConfig.properties** lorsque des événements inattendus nécessitent un nettoyage de base de données immédiat. Les paramètres contenus dans le fichier **JobDispatcherConfig.properties** restent inchangés. Pour plus d'informations sur le fichier **JobDispatcherConfig.properties**, voir *Tivoli Workload Scheduler - Guide d'administration*.

Options

? Affiche des informations d'aide.

-dbUsr *db_user_name*
Indique le nom d'utilisateur d'un utilisateur autorisé à effectuer des opérations sur la ligne de commande.

-dbPwd *db_user_password*
Indique le mot de passe d'un utilisateur autorisé à effectuer des opérations sur la ligne de commande.

-successfulJobsMaxAge *successfulJobsMaxAge*
Indique le nombre d'heures de conservation des travaux terminés avec succès ou annulés dans la base de données du référentiel de travaux avant archivage. La valeur est de 240 heures, soit 10 jours.

-notSuccessfulJobsMaxAge *notSuccessfulJobsMaxAge*
Indique le nombre d'heures de conservation des travaux en échec ou dont le statut est inconnu dans la base de données du référentiel de travaux avant archivage. La valeur est de 720 heures, soit 30 jours.

-archivedJobsMaxAge *archivedJobsMaxAge*
Indique le nombre d'heures de conservation des travaux dans la base de données d'archive avant archivage. La valeur est de 720 heures, soit 30 jours.

Valeurs de retour

La commande **movehistorydata** renvoie l'une des valeurs suivantes :
0 Indique que la commande **movehistorydata** s'est terminée avec succès.
< > 0 Indique que la commande **movehistorydata** a échoué.

Exemples

1. Pour déplacer dans la base de données d'archive tous les travaux réussis dans les 40 dernières heures, entrez la commande suivante :

   ```
   movehistorydata -dbUsr halmst -dbPwd dgordon -successfulJobsMaxAge 40
   ```

2. Pour déplacer dans la base de données d'archive tous les travaux, quel que soit leur statut, et retirer de la base de données d'archive tous les travaux de plus de 700 heures, entrez la commande suivante :

```
movehistorydata -dbUsr halmst -dbPwd dgordon -successfulJobsMaxAge 0
                -notSuccessfulJobsMaxAge 0 -archivedJobsMaxAge 700
```

param

Utilisez la commande **param** pour définir et gérer des mots de passe et des variables utilisateur en local sur des agents dynamiques et sur des agents Tivoli Workload Scheduler for z/OS.

Vous pouvez utiliser cette commande sur les types de travaux avec options avancées. Les valeurs sont résolues lors de la soumission sur l'agent sur lequel le travail est soumis.

Remarque : Sous Windows 2012, la commande n'est pas prise en charge sur Windows PowerShell.

Autorisation

Pour créer, supprimer ou afficher des variables et des mots de passe, vous devez disposer des droits administrateur ou superutilisateur sur le poste de travail qui exécute l'agent ou des droits *TWS_user* sur l'agent.

Syntaxe

param -u | -V |
{-c | -ec} [*fichier.section.*|*fichier..*|*section.*] *variable* [*valeur*] |
[*fichier.section.*|*fichier..*|*section.*] *variable* |
{-d | -fd} [*fichier.section.*|*fichier..*|*section.*] *variable*

Arguments

-u Affiche des informations sur la syntaxe de la commande et quitte l'application.

-V Affiche la version de la commande et quitte l'application.

-c | -ec
Crée la variable ou le mot de passe *variable* et définit sa valeur *valeur*. La variable ou le mot de passe est placé dans un espace de nom *fichier* que vous pouvez organiser en une ou plusieurs sections nommées *section*.

Si vous ne pouvez pas fournir de nom de fichier *fichier*, la variable ou le mot de passe est placé dans le fichier par défaut `jm_variables` dans le chemin d'accès *agent_installation_path*`\TWA\TWS\ITA\cpa\config\jm_variables_files` (`/TWA/TWS/ITA/cpa/config/jm_variables_files`) sur l'agent dynamique.

Si vous ne pouvez pas fournir de nom de section *section*, la variable ou le mot de passe est placé dans le corps du fichier.

Important : Si vous définissez un mot de passe, vous devez spécifier une section nommée `password` pour *variable*. Cela indique que *variable* est un mot de passe.

Si vous créez une variable, *variable* est le nom de variable et *valeur* représente sa valeur. Si vous créez un mot de passe, *variable* représente le nom d'utilisateur et *valeur* est son mot de passe. Si vous n'entrez pas de *valeur* dans les arguments, la commande demande de manière interactive la saisie d'une valeur.

Argument **-c** crée la variable en clair. Argument **-ec** crée la variable dans un format chiffré. Les mots de passe sont chiffrés par défaut également si vous utilisez l'argument **-c**.

-d | -fd
Supprime (**-d**) ou force la suppression (**-fd**) d'un fichier, d'une section ou d'une variable (mot de passe). Vous pouvez utiliser les caractères génériques suivants :

***** Remplace un ou plusieurs caractères alphanumériques.

? Remplace un caractère alphanumérique.

Avec **-d**, la commande demande une confirmation avant la suppression. Avec **-fd**, la commande effectue la suppression sans demander de confirmation.

Lorsque vous supprimez toutes les variables d'une section, cette dernière est supprimée du fichier. Lorsque vous supprimez toutes les sections et toutes les variables d'un fichier, celui-ci est supprimé.

fichier Nom du fichier utilisé comme espace de nom pour *variable*. Si vous n'indiquez pas de *fichier*, la commande utilise le fichier par défaut `jm_variables` dans le chemin d'accès *agent_installation_path*`\TWA\TWS\ITA\cpa\config\jm_variables_files` (`/TWA/TWS/ITA/cpa/config/jm_variables_files`).

Tous les espaces de nom de variables sont dirigés dans le chemin d'accès *agent_installation_path*`\TWA\TWS\ITA\cpa\config\jm_variables_files` (`/TWA/TWS/ITA/cpa/config/jm_variables_files`).

section Nom de la section à l'intérieur de *fichier* où *variable* est défini. Lorsque *variable* est utilisé pour un mot de passe, est placé dans une section nommée `password`. Aucun nom de section n'est requis pour stocker des variables.

valeur Valeur de *variable*.

Variable
Il peut s'agir d'un nom de variable ou d'une identification utilisateur. S'il est utilisé pour l'identification, il doit être placé dans une section nommée `password` dans le fichier d'espaces de nom.

Commentaires

Pour afficher une variable ou un mot de passe, un fichier d'espaces de noms ou une section, utilisez la commande comme suit :

param [*fichier.section.*|*fichier..*|*section.*] *variable*

Où vous pouvez utilisez les caractères génériques * et ? décrits pour la commande de suppression.

Les fichiers d'espaces de nom, y compris le fichier par défaut `jm_variables`, n'ont pas d'extension.

Les noms de variable sont sensibles à la casse.

Sur les systèmes IBM i, si vous utilisez les interpréteurs de commandes QP2TERM et QSH, les mots de passe sont visibles pendant le processus de création avec `param` et apparaissent clairement dans les journaux de l'interpréteur de commandes. Pour garantir l'obscurcissement des mots de passe, vous devez utiliser

les interpréteurs de commandes AIXTERM ou XTERM.

Exemples

La commande :

```
param -c compassets.hardware.platform1 unix
```

définit la variable `platform1` avec la valeur `unix` dans la section `hardware` du nouveau fichier ou du fichier existant nommé `compassets`. La valeur n'est pas chiffrée.

La commande :

```
param -c compassets..platform1 unix
```

définit la variable `platform1` avec la valeur `unix` dans le nouveau fichier ou le fichier existant nommé `compassets`. La valeur n'est pas chiffrée.

La commande :

```
param -ec hardware.platform1 unix
```

définit la variable `platform1` avec la valeur `unix` dans la section `hardware` du fichier par défaut *agent_installation_path*`\TWA\TWS\ITA\cpa\config\jm_variables_files\jm_variables`. La valeur est chiffrée.

La commande :

```
param -c compassets.password.jladams san07rew
```

définit la variable `jladams` avec la valeur `san07rew` dans la section `password` du nouveau fichier ou du fichier existant nommé `compassets`. Puisque `jladams` est défini dans la section `password`, il est interprété comme un nom d'utilisateur. La valeur `san07rew` est chiffrée par défaut puisqu'elle est interprétée comme un mot de passe.

La commande :

```
param *.*.platform1
```

répertorie la variable `platform1` dans tous ses emplacements définis. C'est-à-dire :

```
...\TWA\TWS\ITA\cpa\config\jm_variables_files\compassets.hardware.platform1=unix
...\TWA\TWS\ITA\cpa\config\jm_variables_files\compassets..platform1=unix
...\TWA\TWS\ITA\cpa\config\jm_variables_files\jm_variables.hardware.platform1=***
```

La commande :

```
param password.*adam*
```

répertorie toutes les variables y compris la chaîne`adam` contenue dans la section `password` de tous les fichiers. Dans ce cas :

```
...\TWA\TWS\ITA\cpa\config\jm_variables_files\compassets.password.jladams********
```

La commande :

```
param -d compassets.password.jladams
```

supprime la variable `jladams`.

La commande :

```
param -d compassets.password.*
```

supprime toutes les variables trouvées dans la section `password` et supprime donc la section du fichier `compassets`.

La commande :

```
param -d compassets.*.*
```

supprime tous les contenus (variables et sections contenant des variables) trouvés dans le fichier `compassets` et supprime donc le fichier.

Ressource

La commande **resource** permet de créer, modifier, associer ou définir des ressources en ligne ou hors ligne.

En configurant correctement le fichier **`CLIConfig.properties`** sur l'agent, vous pouvez également exécuter cette commande à partir de n'importe quel agent Tivoli Workload Scheduler connecté. Pour plus d'informations, voir «Utilisation de la commande resource à partir d'un agent», à la page 603.

Syntaxe

resource ?

resource [**-usr** *nom_utilisateur* **-pwd** *mot_de_passe*]
{
[**-create**{ **-logical** *nom* **-type** *type*[**-quantity** *quantité*][**-offline**] |
-group *nom*[**-offline**]}]
|
[**-delete**{**-logical** *nom* |
-group *nom* }]
|
[**-update**{**-computer** *nom*{[**-setOnline** | **-setOffline**]} |
-logical *nom*
[**-setName** *nom*]
[**-setType** *type*]
[**-setQuantity** *quantité*]
[**-setOnline** | **-setOffline**]
[**-addComputer** *nom* |
-addComputerByID *ID* |
-removeComputer *nom* |
-removeComputerByID *ID*]
|
-group *nom*
[**-setName** *nom*]
[**-setOnline** | **-setOffline**]
[**-addComputer** *nom* |
-addComputerByID *ID* |
-removeComputer *nom* |
-removeComputerByID *ID* |
-addLogical *nom* |
-removeLogical *nom*]}]
|
[**-query**{**-computer** *nom* [*-v*] |
-logical *nom* [*-v*] |

-group *nom* [*-v*]}
[**-configFile** *fichier_configuration*]
}

Description

Utilisez cette commande pour gérer des ordinateurs, des ressources logiques et des groupes de ressources. En particulier, il est possible de :

- Créer, mettre à jour, répertorier et supprimer des ressources logiques ou des groupes
- Créer des ressources logiques, les associer à des ordinateurs, définir des groupes de ressources logiques ou d'ordinateurs, et les définir en ligne ou hors ligne
- Récupérer et mettre à jour des propriétés de ressource à l'aide des options de requête et de mise à jour
- Etablir la liste d'ordinateurs associés à une ressource logique en effectuant une requête détaillée sur celle-ci
- Modifier l'association entre des ordinateurs et des ressources logiques
- Définir des ressources en ligne ou hors ligne et récupérer des propriétés d'ordinateur

Options

? Affiche des informations d'aide.

-usr *nom_utilisateur*
Indique le nom d'un utilisateur autorisé à effectuer des opérations sur la ligne de commande. Cette option est requise lorsque la sécurité est activée et que le nom d'utilisateur n'est pas défini dans le fichier de configuration `CLIConfig.properties` (avec le mot-clé `tdwb_user`).

-pwd *mot_passe*
Indique le mot de passe d'un utilisateur autorisé à effectuer des opérations sur la ligne de commande. Cette option est requise lorsque la sécurité est activée et que le mot de passe n'est pas défini dans le fichier de configuration `CLIConfig.properties` (avec le mot-clé `tdwb_pwd`).

-create -logical *nom* **-type** *type*
Crée la ressource logique avec le nom et le type spécifiés. Il est également possible de définir une quantité spécifique ou de définir la ressource hors connexion à l'aide de paramètres facultatifs de la façon suivante :

-create -logical *nom* **-type** *type***-quantity** *quantité* **-offline**

-create -group *nom*
Crée le groupe de ressources du nom spécifié. Il est également possible de le définir hors connexion à l'aide du paramètre facultatif `-offline` de la façon suivante :

-create -group *nom* **-offline**

-delete -logical *nom*
Supprime la ressource logique du nom spécifié.

-delete -group *nom*
Supprime le groupe de ressources du nom spécifié.

-update -computer *nom*
Met à jour le système informatique du nom spécifié. Vous pouvez mettre l'ordinateur en ligne ou hors ligne comme suit :

-update -computer *nom* **-setOnline**
Définit l'ordinateur spécifié en ligne.

-update -computer *nom* **-setOffline**
Définit l'ordinateur spécifié hors ligne.

-update -logical *nom*
Met à jour la ressource logique spécifiée. Vous pouvez mettre à jour les propriétés et le statut d'une ressource comme suit :

-update -logical *nom* **-setName** *nom*
Met à jour le nom de la ressource logique spécifiée.

-update -logical *nom* **-setType** *type*
Met à jour le type de la ressource logique spécifiée.

-update -logical *nom* **-setQuantity** *quantité*
Met à jour la quantité de la ressource logique spécifiée.

-update -logical *nom* **-setOnline**
Définit en ligne la ressource logique spécifiée.

-update -logical *nom* **-setOffline**
Définit hors ligne la ressource logique spécifiée.

Vous pouvez modifier l'association entre une ressource logique et un ordinateur comme suit :

-update -logical *nom* **-addComputer** *nom*
Associe la ressource logique spécifiée à l'ordinateur dont le nom est spécifié.

-update -logical *nom* **-addComputerByID** *ID*
Associe la ressource logique spécifiée à l'ordinateur dont l'ID est spécifié.

-update -logical *nom* **-removeComputer** *nom*
Supprime l'association entre la ressource logique spécifiée et l'ordinateur dont le nom est spécifié.

-update -logical *nom* **-removeComputerByID** *ID*
Supprime l'association entre la ressource logique spécifiée et l'ordinateur dont l'ID est spécifié.

-update -group *nom*
Met à jour le groupe de ressources spécifié. Vous pouvez mettre à jour les propriétés et le statut d'un groupe de ressources comme suit :

-update -group *nom* **-setName** *nom*
Met à jour le nom du groupe de ressources spécifié.

-update -group *nom* **-setOnline**
Définit en ligne le groupe de ressources spécifié.

-update -group *nom* **-setOffline**
Définit hors ligne le groupe de ressources spécifié.

Vous pouvez ajouter ou supprimer des ressources logiques ou des ordinateurs dans un groupe de ressources comme suit :

-update -group *nom* **-addLogical** *nom*
Ajoute la ressource logique avec le nom spécifié au groupe de ressources.

-update -group *nom* **-removeLogical** *nom*
Supprime la ressource logique avec le nom spécifié du groupe de ressources.

-update -group *nom* **-addComputer** *nom*
Ajoute l'ordinateur avec le nom spécifié au groupe de ressources.

-update -group *nom* **-addComputerByID** *ID*
Ajoute l'ordinateur avec l'ID spécifié au groupe de ressources.

-update -group *nom* **-removeComputer** *nom*
Supprime l'ordinateur avec le nom spécifié du groupe de ressources.

-update -groupe *nom* **-removeComputerByID** *ID*
Supprime l'ordinateur avec l'ID spécifié du groupe de ressources.

-query -computer *nom*
Récupère les propriétés suivantes de l'ordinateur spécifié :

- Nom
- ID ordinateur
- Nom du système d'exploitation
- Type du système d'exploitation
- Version du système d'exploitation
- Statut
- Statut de disponibilité

Récupère les propriétés supplémentaires suivantes si vous ajoutez l'option **-v** :

- Mémoire physique
- Mémoire virtuelle
- Utilisation de l'UC
- Mémoire physique disponible
- Mémoire virtuelle disponible
- Espace de permutation disponible
- Mémoire physique attribuée
- Mémoire virtuelle attribuée
- Espace de permutation distribué
- Nombre de processeurs
- Nombre de processeurs attribué
- Type de processeur
- Vitesse de processeur
- Fabricant
- Modèle
- Numéro de série
- Interfaces réseau
- Systèmes de fichiers

Vous pouvez utiliser un astérisque (*) en guise de caractère générique comme suit :

En tant que paramètre unique
Vous devez l'indiquer entre guillemets, par exemple :

```
C:\IBM\TWA\TDWB\bin>resource –query –computer "*"
```

Cette commande renvoie la liste de tous les ordinateurs existants.

Pour compléter un nom d'ordinateur

Vous devez le nom entier entre guillemets, par exemple :

```
C:\IBM\TWA\TDWB\bin> resource –query –computer "lab123*"
```

Cette commande renvoie la liste de tous les ordinateurs existants dont le nom commence par `lab123`.

-query -logical *nom*

Récupère le nom et le type de la ressource logique spécifiée. Récupère les propriétés supplémentaires suivantes si vous ajoutez l'option `-v` :

- Statut
- Quantité
- Allocation en cours
- Liste d'ordinateurs

Vous pouvez utiliser un astérisque (*) en guise de caractère générique comme suit :

En tant que paramètre unique

Vous devez l'indiquer entre guillemets, par exemple :

```
C:\IBM\TWA\TDWB\bin>resource –query –logical "*"
```

Cette commande renvoie la liste de toutes les ressources logiques existantes.

Pour compléter un nom de ressource

Vous devez le nom entier entre guillemets, par exemple :

```
C:\IBM\TWA\TDWB\bin> resource –query –logical "maRes*"
```

Cette commande renvoie la liste de toutes les ressources logiques existantes dont le nom commence par `maRes`.

-query -group *nom*

Récupère le nom et le statut du groupe de ressources spécifié. Récupère la liste d'ordinateurs et de ressources logiques contenue dans le groupe de ressources si vous utilisez l'option `–v`.

Vous pouvez utiliser un astérisque (*) en guise de caractère générique comme suit :

En tant que paramètre unique

Vous devez l'indiquer entre guillemets, par exemple :

```
C:\IBM\TWA\TDWB\bin>resource –query –group "*"
```

Cette commande renvoie la liste de tous les groupes de ressources existants.

Pour compléter un nom de groupe de ressources

Vous devez le nom entier entre guillemets, par exemple :

```
C:\IBM\TWA\TDWB\bin> resource –query –group "myResGrou*"
```

Cette commande renvoie la liste de tous les groupes de ressources existants dont le nom commence par `myResGrou`.

-configFile *fichier_configuration*

Indique le nom et le chemin d'accès d'un fichier de configuration personnalisé. Ce mot clé est facultatif. Si vous ne l'indiquez pas, le fichier de configuration par défaut est utilisé. Pour plus d'informations sur le fichier de configuration, consultez la section relative à **CLIConfig.properties**.

Autorisation

Le nom d'utilisateur et le mot de passe de la commande sont définis dans le fichier `CLIConfig.properties`. Pour substituer les paramètres définis dans ce fichier, vous pouvez entrer le nom d'utilisateur et le mot de passe dans la commande. Pour plus d'informations sur le fichier de configuration `CLIConfig.properties`, consultez la section qui lui est consacrée.

Valeurs de retour

La commande **resource** renvoie une des valeurs suivantes :

0 Indique que la commande s'est exécutée correctement.

< > 0 Indique que la commande a échoué.

Exemples

- Pour créer une ressource logique nommée `myApplication` (de type Applications), entrez la commande suivante :

  ```
  resource.bat -usr john -pwd BXVFDCGS -create -logical myApplication
  -type Applications
  ```

  Le résultat suivant apparaît :

  ```
  AWKCLI153I
  Ressource logique "myApplication" créée.
  ```

- Pour mettre à jour la quantité de la ressource logique `myApplication`, entrez la commande suivante :

  ```
  resource.bat -update -logical myApplication -setQuantity 5
  -usr john -pwd BXVFDCGS
  ```

  Le résultat suivant apparaît :

  ```
  AWKCLI165I Ressource logique "myApplication" mise à jour.
  ```

- Pour ajouter la relation entre une ressource logique et un ordinateur, entrez la commande suivante :

  ```
  resource.bat -update -logical myApplication -addComputer myComputer
  -usr john -pwd BXVFDCGS
  ```

  Le résultat suivant apparaît :

  ```
  AWKCLI165I Ressource logique "myApplication" mise à jour.
  ```

- Pour récupérer les détails d'une ressource logique `myApplication`, entrez la commande suivante :

  ```
  resource.bat -usr john -pwd BXVFDCGS -query -logical myApplication –v
  ```

  Le résultat suivant apparaît :

  ```
  AWKCLI171I Appel du référentiel de ressources en cours pour effectuer
  une requête sur les ressources.

  AWKCLI172I "1" ressources logiques ont été trouvées pour votre requête.
  Les détails sont les suivants :

  Resource Name:myApplication
  Resource Type:Applications
  Resource Status:Online
  Resource Quantity:5
  Resource Current Allocation:0
  Computers List:
          Computer Name:myComputer
  ```

```
                                Computer ID:D656470E8D76409F9F4FDEB9D764FF59
                                Computer Status:Online
                                Computer Availability Status:Unavailable
```

- Pour définir la ressource logique myApplication hors ligne, entrez la commande suivante :

  ```
  resource.bat -usr john -pwd BXVFDCGS -update -logical myApplication
  -setOffline
  ```

  Le résultat suivant apparaît :

  ```
  AWKCLI165I Ressource logique "myApplication" mise à jour.
  ```

- Pour définir l'ordinateur myApplication hors ligne, entrez la commande suivante :

  ```
  resource.bat -usr john -pwd BXVFDCGS -update -computer myComputer
  -setOffline
  ```

  Le résultat suivant apparaît :

  ```
  AWKCLI165I
  Ordinateur "myComputer" mis à jour.
  ```

- Pour récupérer les propriétés de base de l'ordinateur myComputer, entrez la commande suivante :

  ```
  resource.bat -usr john -pwd BXVFDCGS -query -computer myComputer
  ```

  Le résultat suivant apparaît :

```
AWKCLI171I Appel du référentiel de ressources en cours pour effectuer
une requête sur les ressources.
AWKCLI174I "1" ordinateurs ont été trouvés pour votre requête.
Les détails sont les suivants :


Computer Name:myComputer
Computer ID:D656470E8D76409F9F4FDEB9D764FF59
Computer OS Name: Microsoft Windows XP Professional English (United States) version
Computer OS Type:Windows XP
Computer OS Version:5
Computer Status:Offline
Computer Availability Status:Unavailable
```

- Pour récupérer les propriétés détaillées de l'ordinateur myComputer, entrez la commande suivante :

  ```
  resource.bat -usr john -pwd BXVFDCGS -query -computer myComputer -v
  ```

  Le résultat suivant apparaît :

```
AWKCLI171I Appel du référentiel de ressources en cours pour effectuer
une requête sur les ressources.
AWKCLI174I "1" ordinateurs ont été trouvés pour votre requête.
Les détails sont les suivants :


Computer Name:myComputer
Computer ID:D656470E8D76409F9F4FDEB9D764FF59
Computer OS Name:Microsoft Windows XP Professional English (United States) version
Computer OS Type:Windows XP
Computer OS Version:5
Computer Status:Offline
Computer Availability Status:Unavailable
Computer details:
         Physic memory = 2095536.0
         Virtual memory = 3513788.0
         Cpu utilization = 16.0
         Free physic memory = 947972.0
         Free virtual memory = 2333484.0
```

```
Free swap space = 52.0
Allocated physic memory = 0.0
Allocated virtual memory = 0.0
Allocated swap space = 0.0
Processors number = 1.0
Allocated processors number = 0.0
Processor type = x86
Processor speed = 1995.00
Manufacturer = IBM
Model = 2668F8G
Serial number = L3WZYNC
```

- Pour récupérer les propriétés détaillées de la ressource logique `geneva`, y compris la liste des ordinateurs associés, entrez la commande suivante :

  ```
  resource.bat -usr john -pwd BXVFDCGS -query -logical geneva -v
  ```

  Le résultat suivant apparaît :

```
Définition des variables d'environnement de l'interface CLI....
AWKCLI171I Appel du référentiel de ressources en cours pour effectuer
une requête sur les ressources.
AWKCLI172I "1" ressources logiques ont été trouvées pour votre requête.
Les détails sont les suivants :


Resource Name:geneva
Resource Type:prod_wks
Resource Status:Online
Resource Quantity:1
Resource Current Allocation:0
Computers List:
    Computer Name:bd_ff139_1
    Computer ID:666AADE61CBA11E0ACBECD0E6F3527DE
    Computer Status:Online
    Computer Availability Status:Available
    AWKCLI171I Appel du référentiel de ressources en cours pour effectuer
 une requête sur les ressources.
```

- Pour créer un groupe de ressources nommé `myGroup`, entrez la commande suivante :

  ```
  resource.bat -usr john -pwd BXVFDCGS -create -group myGroup
  ```

  Le résultat suivant apparaît :

  ```
  AWKCLI153I
  Groupe de ressources "myGroup" créé.
  ```

- Pour récupérer les propriétés de base du groupe de ressources `myGroup`, entrez la commande suivante :

  ```
  resource.bat -query -group myGroup
  ```

  Le résultat suivant apparaît :

```
Définition des variables d'environnement de l'interface CLI....
  AWKCLI171I Appel du référentiel de ressources en cours pour effectuer
  une requête sur les ressources.
  AWKCLI173I "1" groupes ont été trouvés pour votre requête.
  Les détails sont les suivants :


  Group Name:myGroup
  Group Status:Online
```

- Pour ajouter l'ordinateur `myComputer` à un groupe de ressources nommé `myGroup`, entrez la commande suivante :

  ```
  resource.bat -update -group myGroup -addComputer myComputer
  ```

Le résultat suivant apparaît :

```
Définition des variables d'environnement de l'interface CLI....
  AWKCLI165I Groupe de ressources "myGroup" mis à jour.
```

- Pour récupérer les détails d'un groupe de ressources nommé myGroup, entrez la commande suivante :

```
resource.bat -query -group myGroup -v
```

Le résultat suivant apparaît :

```
Définition des variables d'environnement de l'interface CLI....
  AWKCLI171I Appel du référentiel de ressources en cours pour effectuer
  une requête sur les ressources.
  AWKCLI173I "1" groupes ont été trouvés pour votre requête.
  Les détails sont les suivants :


  Group Name:myGroup
  Group Status:Online
  Computers List:
   Computer Name:myComputer
           Computer ID:D656470E8D76409F9F4FDEB9D764FF59
           Computer Status:Online
   Computer Availability Status:Unavailable

  Resources List:
```

Utilisation de la commande resource à partir d'un agent

Vous pouvez créer et gérer des ressources et des groupes de ressources à partir d'agents Tivoli Workload Scheduler autres que le gestionnaire de domaine maître.

Activation de la commande resource

Pour activer cette fonction, vous devez :

1. Ajouter l'environnement d'exécution pour les travaux Java lors de l'installation de l'agent. Pour plus d'informations sur la procédure d'installation de l'agent, voir le manuel *Planification et installation*.
2. Configurez le fichier **CLIConfig.properties**. Voir la section sur le fichier **CLIConfig.properties**.
3. Exécutez la commande **resource**. Voir «Exécution de la commande resource», à la page 604.

A cet effet, une instance supplémentaire du fichier **CLIConfig.properties** est installée sur chaque agent. Si vous envisagez d'exécuter la commande **resource** à partir d'un agent, vous devez configurer le fichier **CLIConfig.properties** localement.

Configuration du fichier CLIConfig.properties local

Lorsque vous installez l'agent, une copie locale de **CLIConfig.properties** est installée automatiquement et configurée partiellement sur votre agent dans le chemin d'accès suivant :

rép_principal_TWA/TWS/TDWB_CLI/config

Pour exécuter la commande **resource.bat** ou **resource.sh** à partir de l'agent, personnalisez les mots-clés suivants du fichier **CLIConfig.properties** local :

ITDWBServerHost
Indiquez l'adresse IP ou le nom d'hôte du gestionnaire de domaine maître.

ITDWBServerPort
Indiquez le numéro de port HTTP de WebSphere Application Server.

ITDWBServerSecurePort
Indiquez le numéro de port HTTPS de WebSphere Application Server.

tdwb_user
Indiquez le nom d'utilisateur d'un utilisateur autorisé à effectuer des opérations sur IBM Tivoli Workload Scheduler lorsque la sécurité est activée. Cet utilisateur doit être défini auparavant sur IBM WebSphere. Pour plus d'informations sur la sécurité, voir *Tivoli Workload Scheduler - Guide d'administration*, *SC11-6396*.

tdwb_pwd
Indiquez le mot de passe d'un utilisateur autorisé à effectuer des opérations sur IBM Tivoli Workload Scheduler lorsque la sécurité est activée. Ce mot de passe doit être défini auparavant sur IBM WebSphere. Pour plus d'informations sur la sécurité, voir *Tivoli Workload Scheduler - Guide d'administration*.

Exécution de la commande resource

En fonction du système d'exploitation utilisé, entrez ce qui suit pour exécuter la commande :

Sous Windows
```
resource.bat
```
Sous UNIX
```
resource.sh
```

Permutation des gestionnaires

Vous pouvez définir dans le fichier `CLIConfig.properties` les serveurs de courtier de sauvegarde que l'interface CLI de ressource doit contacter si le serveur de courtier en cours ne répond pas. Pour configurer l'interface CLI de ressource pour contacter les serveurs de sauvegarde en cas d'indisponibilité du serveur actuel, vous devez spécifier dans le fichier `CLIConfig.properties` les propriétés de connexion de chacun des serveurs de courtier de sauvegarde. Les propriétés à spécifier sont les mêmes que celles du serveur de courtier exécuté sur le gestionnaire de domaine maître principal.

Définissez les propriétés de connexion suivantes :

```
ITDWBServerHost
ITDWBServerPort
ITDWBServerSecurePort
use_secure_connection
tdwb_user
tdwb_pwd
```

Pour les serveurs de sauvegarde, le même nombre ordinal doit être ajouté à chaque nom de propriété associé au même serveur de sauvegarde.

Dans l'exemple suivant, le fichier `CLIConfig.properties` spécifie le serveur de courtier exécuté sur le gestionnaire de domaine maître principal et deux serveurs de courtier de sauvegarde :

```
# Propriétés du serveur de courtier exécuté sur le gestionnaire de domaine maître principal
ITDWBServerHost = BrokerServer.mycompany.com
ITDWBServerPort = 51117
ITDWBServerSecurePort = 51118
use_secure_connection = true
```

```
tdwb_user = tdwbUser
tdwb_pwd = xxxx

# Propriétés du premier serveur de courtier de sauvegarde (_1)
ITDWBServerHost_1 = FirstBackupBrokerServer.mycompany.com
ITDWBServerPort_1 = 41117
ITDWBServerSecurePort_1 = 41118
use_secure_connection_1 = false
tdwb_user_1 = backup1TdwbUser
tdwb_pwd_1 = yyyy

# Propriétés du second serveur de courtier de sauvegarde
ITDWBServerHost_2 = SecondBackupBrokerServer.mycompany.com
ITDWBServerPort_2 = 61117
ITDWBServerSecurePort_2 = 61118
use_secure_connection_2 = false
tdwb_user_2 = backup2TdwbUser
tdwb_pwd_2 = zzzz
```

Vous pouvez définir jusqu'à 10 serveurs de courtier.

Pour que l'interface CLI de ressource ne contacte pas des serveurs indisponibles, le nom du serveur de courtier connecté est enregistré dans la propriété `ITDWBLastGoodServerHost` du fichier `CLIConfig.properties`.

sendevent

La commande envoie à partir d'un agent dynamique ou d'un gestionnaire de domaine les événements personnalisés définis par la commande evtdef vers le serveur de processeur d'événements actuellement actif dans le plan de production. Lors de la réception des événements par le processeur d'événements, ils déclenchent les règles d'événement dans lesquelles ils ont été spécifiés.

Remarque : Sous Windows 2012, la commande n'est pas prise en charge sur Windows PowerShell.

Les utilisateurs peuvent ignorer le serveur de destination par défaut (défini par les options globales) en spécifiant le nom d'hôte et le port d'un nouveau serveur.

Syntaxe

sendevent -V | ? | -help | -u | -usage

sendevent [**-hostname** *nom_hôte*]
[**-port** *port*]
type_événement
source
[[*attribute=valeur*]...]

Arguments

-V Affiche la version de la commande et quitte l'application.

? | -help | -u | -usage
Affiche des informations sur la syntaxe de la commande et quitte l'application.

-hostname *nom_hôte*
Indique le nom d'hôte d'un serveur de processeur d'événement autre que celui actuellement actif.

-port *port*
Indique le numéro de port d'un serveur de processeur d'événements autre que celui actuellement actif.

type_événement
L'un des types d'événements personnalisés défini par la commande evtdef dans le fournisseur d'événements génériques et spécifié en tant qu'événement déclencheur dans une définition de règle d'événement.

source Nom du fournisseur d'événements que vous avez personnalisé avec evtdef. Il s'agit également du nom que vous devez indiquer comme argument du mot clé `eventProvider` dans la définition des règles d'événement déclenchées par ces événements personnalisés.

Le nom par défaut est `GenericEventPlugIn`.

attribut=*valeur*
L'un ou plusieurs des attributs qualifiant le type d'événement personnalisé qui sont spécifiés en tant qu'attributs d'événement déclencheur pour la règle d'événement.

Commentaires

La commande s'applique à l'environnement dynamique uniquement. Pour envoyer des événements à partir d'agents non dynamiques, voir «sendevent», à la page 573.

Exemples

Dans cet exemple, une application s'exécutant sur un agent dynamique envoie le type d'événement personnalisé `BusProcCompleted` vers le processeur d'événement par défaut. L'événement est que le traitement du fichier `calcweek` est terminé.

```
sendevent BusProcCompleted GenericEventPlugIn TransacName=calcweek
Workstation=acagn002
```

Le nom de fichier et du poste de travail associé sont les deux attributs d'événement `BusProcCompleted` qui ont été spécifiés en tant qu'attributs d'événement déclencheur dans une règle d'événement associée.

twstrace

Change le niveau de trace sur l'agent dynamique sans avoir à arrêter et redémarrer l'agent.

Remarque : Sous Windows 2012, la commande n'est pas prise en charge sur Windows PowerShell.

Autorisation

Vous devez vous connecter avec les données d'identification de l'utilisateur qui a installé l'agent dynamique. Vous pouvez également utiliser toute autorisation supérieure à celle de l'utilisateur qui a installé l'agent dynamique.

Syntaxe

twstrace -u | -V -enable | -disable [**-level** *valeur*] [**-maxFiles***nombre_fichiers*] [**-maxFileBytes** *nombre_octets*] [**-getLogs** [**-zipFile** *nom_fichier_zip*] [**-host***nom_hôte*] [**-protocol** *{http|https}*] [**-port** *numéro_port*] [**-inifile** *nom_fichier_ini*]]

Arguments

enable
Active la trace à son niveau maximum. Le niveau maximum est **3000**. Par défaut, les traces sont désactivées.

disable
Désactive la fonction de trace.

level *valeur*
Niveau de détail des traces :

1000 Les messages d'erreur, d'avertissement et d'information font l'objet d'une trace.

2000 Les messages d'erreur et d'avertissement font l'objet d'une trace.

3000 Les messages d'erreur font l'objet d'une trace.

maxFiles *nombre_fichiers*
Nombre maximal de fichiers de trace que vous souhaitez créer.

maxFileBytes *nombre_octets*
Taille maximale en octets que peut atteindre le fichier de trace. La valeur par défaut est **1024000** octets.

getLogs
Permet de collecter les fichiers de trace, les fichiers messages et les fichiers de configuration dans un fichier compressé.

zipfile *nom_fichier_zip*
Nom du fichier compressé qui contient toutes les informations, à savoir des journaux, des traces et des fichiers de configuration (`ita.ini` et `jobManager.ini`) pour l'agent. La valeur par défaut est `logs.zip`.

host *nom_hôte*
Nom d'hôte ou adresse IP de l'agent pour lequel vous souhaitez collecter les traces. La valeur par défaut est **localhost**.

protocol *http|https*
Protocole de l'agent pour lequel vous collectez la trace. Par défaut, il s'agit du protocole indiqué dans le fichier **.ini** de l'agent.

port *numéro_port*
Port de l'agent. Par défaut, il s'agit du numéro de port de l'agent sur lequel vous exécutez la ligne de commande.

inifile *nom_ficier_ini*
Nom du fichier **.ini** qui contient la configuration SSL de l'agent pour lequel vous souhaitez collecter les traces. Par défaut, il s'agit du fichier **.ini** de l'agent local. Si vous collectez la trace d'un agent distant pour lequel vous avez personnalisé les certificats de sécurité, vous devez importer le certificat sur l'agent local et indiquer le nom du fichier **.ini** qui contient la configuration. Pour cela, procédez comme suit :

1. Extrayez le certificat du fichier de clés de l'agent distant.
2. Importez le certificat sur le fichier de clés d'un agent local. Vous pouvez créer un fichier de clés ad hoc dont le nom doit être **TWSClientKeyStore.kdb**.
3. Créez un fichier **.ini** dans lequel vous précisez :
   - **0** dans la propriété **tcp_port** comme suit :

```
tcp_port=0
```

- Port de l'agent distant dans la propriété **ssl_port** comme suit :

  ```
  ssl_port=<ssl_port>
  ```

- Chemin d'accès au fichier de clés que vous avez créé à l'étape 2 dans la propriété **key_repository_path** comme suit :

  ```
  key_repository_path=<local_agent_keystore_path>
  ```

r Affiche la syntaxe de la commande.

V Affiche la version du produit.

Exemples

Pour définir le niveau de trace afin d'enregistrer les messages d'erreur et d'avertissement, exécutez la commande suivante :

```
twstrace -enable -level 2000
```

Pour extraire les informations sur le niveau de trace, exécutez la commande suivante :

```
twstrace -level -maxFiles -maxFileBytes

AWSITA1761 Les propriétés de trace sont les suivantes : level="1000",
maxFiles="3", file size="1024000"
```

Chapitre 15. Extraction des états et des statistiques

Le présent chapitre détaille les commandes de génération d'états qui permettent de préparer un récapitulatif ou les détails du plan de production précédent ou suivant. Ces commandes sont exécutées à partir de l'invite de commande du système d'exploitation sur le gestionnaire de domaine maître. Il comprend les sections suivantes :

Configuration de l'utilisation des commandes de génération d'états

Pour configurer l'environnement pour utiliser les commandes de génération d'états, définissez les variables *PATH* et *TWS_TISDIR* en exécutant l'un des scripts suivants :

- `. ./`*racine_TWS*`/tws_env.sh` pour les interpréteurs de commandes Bourne et Korn sous UNIX
- `. ./`*racine_TWS*`/tws_env.csh` pour les interpréteurs de commandes C sous UNIX
- *racine_TWS*`\tws_env.cmd` dans Windows

Les commandes de génération d'états doivent être exécutées à partir du répertoire *TWS_home*.

Les variables d'environnement suivantes contrôlent le résultat des commandes de génération d'états :

MAESTROLP
: Indique la destination du résultat d'une commande. La valeur par défaut est **stdout**. Vous pouvez lui attribuer l'une des valeurs suivantes :

  file_name
  : Le résultat est écrit dans un fichier.

  > *file_name*
  : Sous UNIX uniquement. Le résultat est redirigé vers un fichier dont le contenu est remplacé. Si le fichier n'existe pas, il est créé.

  >> *file_name*
  : Sous UNIX uniquement. Le résultat est redirigé vers un fichier et ajouté à la fin de ce dernier. Si le fichier n'existe pas, il est créé.

  | *commande*
  : Sous UNIX uniquement. La sortie est dirigée vers une commande système ou un processus. La commande système est toujours exécutée.

| | *commande*
Sous UNIX uniquement. La sortie est dirigée vers une commande système ou un processus. La commande système n'est pas exécutée en l'absence de résultat.

MAESTRO_OUTPUT_STYLE
Indique le style du résultat pour les noms d'objet longs. Avec la valeur **LONG**, des zones longues sont utilisées pour les noms d'objet.

Si la valeur affectée à cette variable est différente de **LONG**, les noms longs sont tronqués à huit caractères suivis du signe plus. Par exemple : **A1234567+**.

Pour obtenir le format de rapport d'état correct, vous devez utiliser une taille de police fixe.

Modification du format de date

Dans Tivoli Workload Scheduler, le format de date affecte toutes les commandes qui acceptent une date comme option d'entrée (à l'exception de la commande **datecalc**) ainsi que les en-têtes de tous les états. Le format de date par défaut est *mm/jj/aa*. Pour sélectionner un format différent, éditez l'option locale ***format de date*** dans le fichier `localopts`. Les valeurs possibles sont les suivantes :

Tableau 81. Formats de date

Valeur du *format de date*	**Résultat de format de date correspondant**
0	jj/mm/aa
1	jj/mm/aa
2	jj/mm/aa
3	Les variables NLS.

Voir *IBM Tivoli Workload Scheduler - Guide d'administration* pour plus d'informations sur la modification des variables locales du fichier `localopts`.

Description des commandes

Les commandes de génération d'états de Tivoli Workload Scheduler sont répertoriées dans le tableau 82 :

Tableau 82. Liste des commandes de génération d'états

Commande	**Description**
rep1	Etat 01 - Liste des caractéristiques des travaux
rep2	Etat 02 - Liste des invites
rep3	Etat 03 - Liste des agendas des utilisateurs
rep4a	Etat 04A - Liste des paramètres des utilisateurs
rep4b	Etat 04B - Liste des ressources TWS
rep7	Etat 07 - Liste des historiques des travaux
rep8	Etat 08 - Histogramme des travaux
rep11	Etat 11 - Calendrier de production planifiée

Tableau 82. Liste des commandes de génération d'états (suite)

Commande	Description
reptr	Etat 09A - Récapitulatif de la production planifiée Etat 09B - Détail de la production planifiée Etat 09D - Détail de la production planifiée (format étendu) Etat 10A - Récapitulatif de la production réelle Etat 10B - Détails de la production réelle
xref	Etat 12 - Etat de références croisées

rep1 à rep4b

Ces commandes permettent d'imprimer les états suivants :

Etat 01
Liste des caractéristiques des travaux

Etat 02
Liste des invites

Etat 03
Liste des agendas

Etat 04A
Liste des paramètres

Etat 04B
Liste des ressources

Syntaxe

rep[*x*] [**-V**|**-U**]

Exécutez la commande à partir du répertoire *TWS_home*.

Pour `rep3`, exécutez la commande à partir d'un répertoire sur lequel vous avez l'accès en écriture `write`.

Lors de l'impression des rapports pour les types de travaux avec options avancées, la zone du fichier JCL renvoie le nom de l'application.

Arguments

x Numéro de l'état. Les numéros possibles sont les suivants : **1**, **2**, **3**, **4a** ou **4b**.

-U Affiche des informations sur la syntaxe de la commande et quitte l'application.

-V Affiche la version de la commande et quitte l'application.

Commentaires

La Liste des caractéristiques des travaux (rapport 01) ne peut pas contenir de travaux ayant été soumis via un nom d'alias.

Le temps écoulé affiché pour un travail reflet est le temps écoulé du travail distant lié.

Exemples

Imprimez l'Etat 03 "Liste des agendas des utilisateurs" :

```
rep3
```

Affichez les informations d'utilisation pour la commande **rep2** :

```
rep2 -U
```

Sous UNIX, imprimez deux exemplaires de l'Etat 04A "Liste des paramètres des utilisateurs" sur l'imprimante lp2 :

```
MAESTROLP="| lp -dlp2 -n2"
export MAESTROLP
rep4a
```

Voici un exemple de rapport pour le travail WAGES2_1 :

```
Travail : WAGES2_1   #FTP                Description :
Fichier JCL : filetransfer
Connexion :                  Créateur : tws86
Travail de reprise :
Type de reprise : STOP
Invite de reprise :
Autodoc composer : Oui
Nombre total d'exécutions : 0 - 0 réussies, 0 abandonnées

                 Ecoulé(mins) CPU(secs)
Total       0           0
Normal      0
Dernière exécution     0           0 (De    0 à   0)
Maximum      0           0 (De    0)
Minimum      0           0 (De    0)>
```

rep7

Cette commande permet d'imprimer l'Etat 07 - Liste des historiques des travaux.

Syntaxe

rep7 -V|-U

rep7
[**-c** *poste_travail*]
[**-s** *nom_flot_données*]
[**-j** *travail*]
[**-f** *date*]
[**-t** *date*]
[**-l**]

Exécutez la commande à partir du répertoire *TWS_home*.

Arguments

-U Affiche des informations sur la syntaxe de la commande et quitte l'application.

-V Affiche la version de la commande et quitte l'application.

-c *poste_travail*
Indique le nom du poste de travail sur lequel les travaux s'exécutent. Par défaut, tous les postes de travail sont pris en compte.

-s *nom_flot_travaux*
Indique le nom du flot de travaux dans lequel les travaux s'exécutent. Par défaut, tous les flots de travaux sont pris en compte.

-j *travail*
Indique le nom du travail. Par défaut, tous les travaux sont pris en compte.

-f *date* Indique que l'historique des travaux est imprimé à partir de cette date. Entrez la date au format *aaaammjj*. Par défaut, la date disponible la plus ancienne est prise en compte.

-t *date* Indique que l'historique des travaux est imprimé jusqu'à cette date. Entrez la date au format *aaaammjj*. Par défaut, la date la plus récente est prise en compte.

-l Limite les informations de ligne de résumé aux travaux appartenant à la plage de dates définie par les options -f ou -t. L'utilisation de cette option entraîne l'inversion du résultat : la ligne de résumé est imprimée après les lignes d'exécution individuelles. Cette option est valable uniquement si vous précisez aussi au moins l'une des options -f ou -t.

Commentaires

Le temps écoulé affiché pour un travail reflet est le temps écoulé du travail distant lié.

Chaque fois que vous exécutez **rep7**, la sortie de la commande contient les informations enregistrées jusqu'à la dernière exécution de **JnextPlan**. Les informations concernant l'exécution du plan de production courant seront contenues dans la sortie **rep7** lors de la prochaine exécution de **JnextPlan**. Pour cette raison, si vous exécutez **rep7** après la première génération du plan de production ou après une commande **ResetPlan**, la sortie de la commande ne contient aucune information statistique.

Exemples

Imprimez l'historique de tous les travaux pour le poste de travail `ux3` :

```
rep7 -c ux3
```

Imprimez l'historique de tous les travaux contenus dans le flot de travaux `sked25` :

```
rep7 -s sked25
```

Imprimez l'historique de tous les travaux contenus dans le flot de travaux `mysked` sur le poste de travail `x15` entre le 21 janvier 2005 et le 25 janvier 2005 :

```
rep7 -c x15 -s mysked -f 20050121 -t 20050125
```

rep8

Cette commande permet d'imprimer l'état 08 - Histogramme des travaux.

Syntaxe

rep8 -V|-U

rep8
[**-f** *date* **-b** *heure* **-t** *date* **-e** *heure*]
[**-i** *fichier*]
[**-p**]

rep8
[**-b** *heure* **-e** *heure*]
[**-i** *fichier*]
[**-p**]

Exécutez la commande à partir du répertoire *TWS_home*.

Arguments

-U Affiche des informations sur la syntaxe de la commande et quitte l'application.

-V Affiche la version de la commande et quitte l'application.

-f *date* Indique que l'historique des travaux est imprimé à partir de cette date. Entrez la date au format *aaaammjj*. La valeur par défaut est la date du jour.

-b *heure*
Indique que l'historique des travaux est imprimé à partir de cette heure. Entrez l'heure au format *hhmm*. La valeur par défaut correspond au paramètre *startOfDay* de Tivoli Workload Scheduler.

-t *date* Indique que l'historique des travaux est imprimé jusqu'à cette date. Entrez la date au format *aaaammjj*. Par défaut, la date la plus récente est prise en compte.

-e *heure*
Indique que l'historique des travaux est imprimé jusqu'à cette heure. Entrez l'heure au format *hhmm*. La valeur par défaut est l'heure de début de la journée de Tivoli Workload Scheduler.

-i *fichier*
Indique le nom du fichier journal duquel est extrait l'historique des travaux. Notez que les fichiers journaux sont stockés dans le répertoire `schedlog`. Par défaut, il s'agit du fichier `Symphony` courant.

Remarque : Assurez-vous que la plage horaire indiquée par les arguments [**-f** *date* **-b** *heure* **-t** *date* **-e** *heure*] est comprise dans la plage horaire et la date définies dans le nom du fichier journal **-i** *fichier*.

-p Indique qu'un saut de page est inséré après chaque date d'exécution.

Commentaires

Chaque fois que vous exécutez **rep8**, la sortie de la commande contient les informations enregistrées jusqu'à la dernière exécution de **JnextPlan**. Les informations concernant l'exécution du plan de production courant seront contenues dans la sortie **rep8** lors de la prochaine exécution de **JnextPlan**. Pour cette raison, si vous exécutez **rep8** après la première génération du plan de production ou après une commande **ResetPlan**, la sortie de la commande ne contient aucune information statistique.

Exemples

Imprimez un histogramme des travaux qui inclut toutes les informations du plan courant (fichier `Symphony`) :

```
rep8
```

Imprimez un histogramme des travaux qui commence à 6 heures du matin le 25 janvier 2005 et qui se termine à 5 heures 59 du matin le 26 janvier 2005.

```
rep8 -f 20050125 -b 0600 -t 20050126 -e 0559 -i schedlog/M199801260601
```

Imprimez un histogramme des travaux, à partir du plan courant (fichier Symphony), qui commence à 6 heures du matin et se termine à 22 heures :

```
rep8 -b 0600 -e 2200
```

rep11

Cette commande permet d'imprimer l'état 11 "Calendrier de production planifiée".

Syntaxe

rep11 -V|-U

rep11
[**-m** *mm*[*aa*] [...]]
[**-c** *stat_travail* [...]]
[**-s** *nom_flot_données*]
[**-o** *sortie*]

Exécutez la commande à partir du répertoire *TWS_home*.

Arguments

-U Affiche des informations sur la syntaxe de la commande et quitte l'application.

-V Affiche la version de la commande et quitte l'application.

-m *mm*[*aa*]
Indique les mois à prendre en compte. Entrez le numéro du mois au format *mm*. Par défaut, le mois courant est pris en compte.

Vous pouvez également indiquer une année en utilisant le format *aa*. Par défaut, l'année courante est prise en compte ou l'année suivante si vous indiquez un mois qui précède le mois courant.

-c *poste_travail*
Indique les postes de travail à prendre en compte dans les états. Par défaut, tous les postes de travail sont pris en compte.

-s *nom_flot_travaux*
Indique le nom du flot de travaux dans lequel les travaux s'exécutent. Par défaut, tous les flots de travaux sont pris en compte.

-o *sortie*
Indique le fichier de sortie. Il s'agit, par défaut, du fichier défini par la variable ***MAESTROLP***. Si la variable ***MAESTROLP*** n'est pas définie, le fichier par défaut est **stdout**.

Exemples

Générez un état pour les mois de juin, juillet et août 2004 pour les postes de travail `principal`, `site1` et `sagent1` :

```
rep11 -m 0604 0704 0804 -c main site1 sagent1
```

Générez un état pour les mois de juin, juillet et août de l'année en cours pour tous les postes de travail et dirigez la sortie vers le fichier `r11out` :

```
rep11 -m 06 07 08 -o r11out
```

Générez un état pour le mois et l'année en cours pour le poste de travail `site2` :

```
repll -c site2
```

reptr

Cette commande permet d'imprimer les états suivants :

Etat 09A
Récapitulatif de la production planifiée

Etat 09B
Détail de la production planifiée

Etat 10A
Récapitulatif de la production réelle

Etat 10B
Détail de la production réelle

Les états 09A et 09B se rapportent à un traitement de production futur alors que les états 10A et 10B affichent les résultats et le statut du traitement de chaque travail de la production déjà traitée.

Syntaxe

reptr [-V|-U]

reptr -pre
[-{**summary** | **detail**}]
[*fichier_sym*]

reptr -post
[-{**summary** | **detail**}]
[*fichier_journal*]

Exécutez la commande à partir d'un répertoire auquel vous avez accès `write`.

Arguments

-U Affiche des informations sur la syntaxe de la commande et quitte l'application.

-V Affiche la version de la commande et quitte l'application.

-pre Permet de demander l'impression des états de préproduction (09A et 09B).

-post Permet de demander l'impression des états de postproduction (10A et 10B).

-summary
Permet de demander l'impression des états récapitulatifs (09A et 10A). Si les arguments **-summary** et **-detail** sont omis, les deux types d'état sont imprimés.

-detail Permet de demander l'impression des états détaillés (09B et 10B). Si les arguments **-summary** et **-detail** sont omis, les deux types d'état sont imprimés.

fichier_sym
Indique le nom du fichier de plan de production à partir duquel les états seront imprimés. Le fichier par défaut est `Symnew` dans le répertoire courant. Si le fichier ne se trouve pas dans le répertoire de travail en cours, vous devez ajouter le chemin absolu au nom du fichier.

fichier_journal
Spécifie le nom complet du fichier journal à partir duquel les rapports seront imprimés. Notez que les fichiers journaux du plan de production sont stockés dans le répertoire `schedlog`. Le fichier par défaut est celui du plan courant (fichier `Symphony`).

Si la commande est exécutée sans options, les deux états **pre** (09A et 09B) sont imprimés et les informations sont extraites du fichier `Symphony`.

Exemples

Imprimez l'état détaillé de préproduction à partir du fichier `Symnew` :

```
reptr -pre -detail
```

Imprimez l'état récapitulatif de préproduction à partir du fichier `mysym` :

```
reptr -pre -summary mysym
```

Imprimez l'état récapitulatif de postproduction à partir du fichier journal `M199903170935` :

```
reptr -post -summary schedlog/M199903170935
```

Imprimez les états de préproduction à partir du fichier Symphony.

```
reptr
```

Lorsque les arguments sont spécifiés, les états de préproduction s'appuient sur les informations contenues dans le fichier `Symnew` alors que les états de postproduction s'appuient sur les informations contenues dans le fichier `Symphony`.

xref

Cette commande permet d'imprimer l'Etat 12 "Etat de références croisées".

Syntaxe

xref [**-V** | **-U**]

xref
[**-cpu** *poste_de_travail*]
[**-depends** | **-files** | **-jobs** | **-prompts** | **-resource** | **-schedules** | **-when**[...]]

Exécutez la commande à partir du répertoire *TWS_home*.

Arguments

-U Affiche des informations sur la syntaxe de la commande et quitte l'application.

-V Affiche la version de la commande et quitte l'application.

-cpu *poste_de_travail*
Permet d'imprimer l'état du poste de travail désigné. Le caractère générique @ étant autorisé, les informations de tous les postes de travail qualifiés sont prises en compte. Par défaut, tous les postes de travail sont pris en compte.

-depends
Permet d'imprimer un état détaillant les flots de travaux et les travaux avec leurs successeurs.

-files Permet d'imprimer un état détaillant les flots de travaux et les travaux dépendants de chaque fichier.

-jobs Permet d'imprimer un état détaillant les flots de travaux dans lesquels chaque travail est exécuté.

-prompts
Permet d'imprimer un état détaillant les flots de travaux et les travaux dépendants de chaque invite.

-resource
Permet d'imprimer un état détaillant les flots de travaux et les travaux dépendants de chaque ressource.

-schedules
Permet d'imprimer un état détaillant les flots de travaux et les travaux remplaçants de chaque flot de travaux.

-when Permet d'imprimer un état détaillant les dates `Include` et `Exclude` des flots de travaux.

Si la commande est exécutée sans options, tous les postes de travail et toutes les options sont sélectionnés.

Exemples

Imprimez un état pour tous les postes de travail, qui détaille toutes les informations de référence croisée :

```
xref
```

Imprimez un état pour tous les postes de travail. Incluez les informations de référence croisée relatives à toutes les dépendances remplaçantes :

```
xref -cpu @ -depends -schedules
```

Exemples de sorties d'état

Etat 01 - Liste des caractéristiques des travaux :

```
TWS for UNIX (AIX)/REPORT1 8.3 (1.7)     ibm              Page  1
Etat 01              Liste détails travaux         03/06/06

Travail     : FTAWIN8+    #SCHEDDDD
Description :
Fichier JCL   : dir
Connexion    : maestro_adm
Créateur     : root
Travail de reprise :
Type de reprise : STOP
Invite de reprise :
Autodoc composer : Yes
Nombre total d'exécutions : 0 - 0 réussies, 0 abandonnées

                 Ecoulé(mins) CPU(secs)
Total 00:00:00 0
Normale 00:00:00
Dernière exécution 00:00:00 0 (Sur à 00:00)
Maximum 00:00:00 0 (Sur )
Minimum 00:00:00 0 (Sur )


Travail       : MASTER8+    #JnextPlan
Description   : AJOUTE PAR LE PROGRAMME composer POUR LE CALENDRIER MASTER821#FINAL.
```

```
Fichier JCL     : /test/maestro_adm/tws/JnextPlan
Connexion    : maestro_adm
Créateur     : maestro_adm
Travail de reprise :
Type de reprise : STOP
Invite de reprise :
Autodoc composer : Yes
Nombre total d'exécutions : 11 - 11 réussies, 0 abandonnées

                  Ecoulé(mins) CPU(secs)
Total       00:00:14       44
Normale      00:00:01
Dernière exécution     00:00:01       4 (Le 03/05/06 à 23:16)
Maximum      00:00:02       4 (Le 03/04/06)
Minimum      00:00:01       4 (le 03/04/06)


Travail     : MASTER8+    #JOB1
Description   : AJOUTE PAR LE PROGRAMME composer
Fichier JCL   : pwd
Connexion    : ^ACCLOGIN^
Créateur     : root
Travail de reprise :
Type de reprise : STOP
Invite de reprise :
Autodoc composer : Yes
Nombre total d'exécutions : 1 - 1 réussies, 0 abandonnées

                  Ecoulé(mins) CPU(secs)
Total       00:00:01       0
Normale      00:00:01
Dernière exécution      00:00:01       0 (Le 03/05/06 à 22:22)
Maximum      00:00:01       0 (Le 03/05/06)
Minimum      00:00:01       0 (Le 03/05/06)

                        * * * *   Fin de l'état  * * * *
```

Dans la sortie, les valeurs définies dans la section «Travail», à la page 772 sont indiquées comme suit :

Autodoc composer
Indique si l'instruction du travail a été décrite dans la définition du flot de travaux à l'aide de l'interface de ligne de commande.

CPU (secs)
Représente la durée réelle, exprimée en secondes, pendant laquelle le travail a utilisé l'UC pour s'exécuter.

Total Représente le temps UC total enregistré pour le 'Nombre total d'exécutions'.

Normal
Représente la valeur moyenne de temps UC enregistrée au cours du 'Nombre total d'exécutions'.

Dernière exécution
Représente le temps UC enregistré au cours de la dernière exécution du travail.

Maximum
Représente la valeur maximale parmi les valeurs collectées pour le temps UC au cours du Nombre total d'exécutions' (calculé uniquement pour les travaux s'étant achevés correctement).

Minimum
Représente la valeur minimale parmi les valeurs collectées pour le temps UC au cours du Nombre total d'exécutions' (calculé uniquement pour les travaux s'étant achevés correctement).

Créateur
Représente le nom de l'utilisateur qui a créé la définition du travail.

Description
Correspond à la description textuelle du travail définie dans la zone **description** de l'instruction de définition du travail.

Ecoulé
Représente le temps écoulé, exprimé en minutes, qui inclut non seulement le temps pendant lequel le travail a utilisé l'UC, mais aussi le temps pendant lequel il a dû attendre que d'autres processus la libèrent.

Total Représente le temps total écoulé enregistré pour le 'Nombre total d'exécutions'.

Normal
Représente la valeur moyenne de temps écoulé enregistrée au cours du 'Nombre total d'exécutions'.

Dernière exécution
Représente le temps écoulé enregistré au cours de la dernière exécution du travail.

Maximum
Représente la valeur maximale parmi les valeurs collectées pour le temps écoulé au cours du 'Nombre total d'exécutions' (calculé uniquement pour les travaux s'étant achevés correctement).

Minimum
Représente la valeur minimale parmi les valeurs collectées pour le temps écoulé au cours du Nombre total d'exécutions' (calculé uniquement pour les travaux s'étant achevés correctement).

Remarque : Le temps écoulé affiché pour un travail reflet est le temps écoulé du travail distant lié.

Fichier JCL
Représente le nom du fichier défini dans la zone **scriptname** qui contient le script à exécuter, ou la commande spécifiée dans la zone **docommand** indiquant la commande à appeler lors de l'exécution du travail.

Travail
Identifiant du travail, [*poste_de_travail***#**]*nom_travail*.

Connexion
Correspond au nom d'utilisateur spécifié dans la zone **streamlogon** et sous lequel le travail est exécuté.

Travail de reprise
Travail, spécifié par *after [poste_de_travail#]nom_travail*, exécuté en cas d'abandon du travail parent.

Invite de reprise
Représente le texte de l'invite, spécifié dans la zone **abendprompt**, qui s'affiche en cas d'abandon de ce travail.

Type de reprise
Représente l'option de reprise définie dans la définition du travail. Elle peut avoir pour valeur **stop**, **continue** ou **rerun**.

Etat 02 - Liste des invites :

```
TWS for UNIX (AIX)/REPORT2 8.3 (1.7)     ibm            Page  1
Etat 02              Liste des invites        03/06/06


Invite   Message

PROMPT1   Répondez OUI dès que vous êtes prêt à exécuter les processus acc103 et acc104.
PROMPT2   Tous les utilisateurs se sont-ils déconnectés ?
CALLNO   555-0911
CALLOPER  Appelez ^PERSON2CALL^ au ^CALLNO^ pour vérifier que toutes les cartes de présence ont été traitées.
PERSON2CALL Louis Dubois

Nombre total d'invites sur le fichier :    5

                        * * * *   Fin de l'état  * * * *
```

La sortie de l'Etat 02 répertorie le nom et le texte des invites définies dans l'environnement.

Etat 03 - Liste des agendas :

```
TWS for UNIX (AIX)/REPORT3 8.3 (1.7)     ibm              Page  1
Etat 03              Liste des agendas des utilisateurs          03/06/06


Type d'agenda : MONTHEND
Description : fin du mois jusqu'à fin 2006.


          Jan 2006           Fév 2006           Mars 2006
Dim Lun Mar Mer Jeu Ven Sam  Dim Lun Mar Mer Jeu Ven Sam  Dim Lun Mar Mer Jeu Ven Sam
                        .             .   .   .   .   .             .   .   .   .   .
 .   .   .   .   .   .   .    .   .   .   .   .   .   .    .   .   .   .   .   .   .
 .   .   .   .   .   .   .    .   .   .   .   .   .   .    .   .   .   .   .   .   .
 .   .   .   .   .   .   .    .   .   .   .   .   .   .    .   .   .   .   .   .   .
 .   .   .   .   .   .   .    .  28             .   .   .   .  31
 .  31

          Avril 2006           Mai 2006           Juin 2006
Dim Lun Mar Mer Jeu Ven Sam  Dim Lun Mar Mer Jeu Ven Sam  Dim Lun Mar Mer Jeu Ven Sam
                      .   .    .   .   .   .   .   .   .                 .   .   .   .
 .   .   .   .   .   .   .    .   .   .   .   .   .   .    .   .   .   .   .   .   .
 .   .   .   .   .   .   .    .   .   .   .   .   .   .    .   .   .   .   .   .   .
 .   .   .   .   .   .   .    .   .   .   .   .   .   .    .   .   .   .   .   .   .
 .   .   .   .   .   .  30   .   .  31           .   .   .   .  30


          Juil 2006           Août 2006           Sept 2006
Dim Lun Mar Mer Jeu Ven Sam  Dim Lun Mar Mer Jeu Ven Sam  Dim Lun Mar Mer Jeu Ven Sam
                      .   .        .   .   .   .   .   .                     .   .   .
 .   .   .   .   .   .   .    .   .   .   .   .   .   .    .   .   .   .   .   .   .
 .   .   .   .   .   .   .    .   .   .   .   .   .   .    .   .   .   .   .   .   .
 .   .   .   .   .   .   .    .   .   .   .   .   .   .    .   .   .   .   .   .   .
 .   .   .   .   .   .   .    .   .   .  31         .   .   .   .   .  30
31

          Oct 2006           Nov 2006           Déc 2006
Dim Lun Mar Mer Jeu Ven Sam  Dim Lun Mar Mer Jeu Ven Sam  Dim Lun Mar Mer Jeu Ven Sam
                        .             .   .   .   .   .                     .   .   .
 .   .   .   .   .   .   .    .   .   .   .   .   .   .    .   .   .   .   .   .   .
```

```
 .   .   .   .   .   .   .    .   .   .   .   .   .   .    .   .   .   .   .   .   .
 .   .   .   .   .   .   .    .   .   .   .   .   .   .    .   .   .   .   .   .   .
 .   .   .   .   .   .   .    .   .   . 30       .   .   .   .   .   .  31
 .  31
```

```
                        * * * *   Fin de l'état  * * * *
```

Dans la sortie, les jours de fin de mois sélectionnés dans le calendrier **MONTHEND** sont mis en évidence.

Etat 04A - Liste des paramètres :

```
TWS for UNIX (AIX)/REPORT4A 8.3 (1.7)     ibm             Page  1
Etat 4A               Liste des paramètres des utilisateurs        03/06/06


Nom du Paramètre     Contenu

ACCHOME          /usr/local/Tivoli/maestro_adm
ACCLOGIN         maestro_adm
BADEXIT 99
GOODEXIT 0
SCRPATH          /usr/local/Tivoli/maestro_adm/scripts

Nombre de paramètres pour le fichier :   5

                        * * * *   Fin de l'état  * * * *
```

La sortie de l'Etat 04A répertorie le nom et le contenu des paramètres définis dans l'environnement.

Etat 04B - Liste des ressources :

```
TWS for UNIX (AIX)/REPORT4B 8.3 (1.7)     ibm             Page  1
Etat 4B               Liste des ressources TWS        03/06/06


               Ressource    Numéro
Poste de travail  Nom       Disponible    Description

FTAHP       #DATTAPES        1      Unités de bande DAT

FTAWIN8+     #QUKTAPES        2      Unités de bande rapides

MASTER8+     #TAPES         2      Unités de bande

MASTER8+     #JOBSLOTS      1024      Emplacements de travaux


Nombre de ressources pour le fichier :       4

                        * * * *   Fin de l'état  * * * *
```

La sortie de l'Etat 04B répertorie le nom, le nombre des ressources disponibles définies dans l'environnement et leur description.

Etat 07 - Liste des historiques des travaux :

```
TWS for UNIX (AIX)/REPORT7 8.3  (1.13)     ibm             Page  1
Etat 07               Liste des historiques des travaux         03/08/06

Date    Heure   Nom du flot de travaux   Ecoulé    UC   Etat

Travail : MASTER8+#MyJS Exécutions : Abandonnées 0 Réussies 11 Temps écoulé : Normal 1 Min 1 Max 2
```

```
03/03/06  01:46   MASTER8+#JS1      1     4    SU
03/03/06  19:08   MASTER8+#JS2      1     4    SU
03/03/06  19:33   MASTER8+#JS3      1     4    SU
03/03/06  19:37   MASTER8+#JS4      1     4    SU
03/03/06  23:08   MASTER8+#JS5      2     4    SU
03/03/06  05:59   MASTER8+#JS_A      1     4    SU
03/05/06  05:59   MASTER8+#JS_G      1     4    SU
03/06/06  05:59   MASTER8+#JS_H      1     4    SU
03/06/06  21:57   MASTER8+#TIMEJ     2     4    SU
03/06/06  23:16   MASTER8+#SLEEPJ     1     4    SU

Travail :MASTER8+#JOB1 Exécutions : Abandonnées 0 Réussies 1 Temps écoulé : Normal 1 Min 1 Max 1

03/06/06  22:22   MASTER8+#JOBS      1     0    SU


                        * * * *   Fin de l'état  * * * *
```

L'Etat 7 lit les informations relatives à l'exécution du travail enregistrées dans la base de données et les affiche. Les différents états possibles sont les suivants :

AB pour les travaux qui ont échoué

SU pour les travaux qui se sont correctement terminés

DN pour les travaux soumis dont l'état est inconnu car ni un message de succès, ni un message d'échec n'ont encore été reçus.

Etat 08 - Histogramme des travaux :

```
TWS for UNIX (AIX)/REPORT8 8.3 (1.7)     ibm              Page  1
Etat 08         Histogramme des travaux 03/05/06 14:05 - 03/06/06 14:04   03/06/05

Intervalle par col. : 15 minutes

                        1  1  1  1  2  2  2  0  0  0  0  0  0  0  1  1  1
                        4  5  7  8  0  1  3  0  2  3  5  6  8  9  1  2  4
                        0  3  0  3  0  3  0  3  0  3  0  3  0  3  0  3  0
                        5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  4

Nom du travail   Etat
03/06/06

CF05066+.JnextPlan  SU                      .*.


                        * * * *   Fin de l'état  * * * *
```

La sortie de l'Etat 8 indique les périodes au cours desquelles les travaux s'exécutent. Les chiffres fournis dans la partie supérieure de l'histogramme des travaux correspondent à des heures, écrites verticalement de haut en bas ; par exemple, la première colonne se lit 1405, ce qui signifie 14 heures 05. Les périodes pendant lesquelles le travail s'exécute sont marquées par des points et des astérisques lorsque la position du marqueur est alignée avec une heure écrite de haut en bas.

Etat 9B - Détail de la production planifiée :

```
TWS for UNIX (AIX) REPORTER 8.3 (1.7)    ibm               Page  1
Etat 09B  Symnew    Détail de la production planifiée pour le 03/06/06         03/06/06

                      Temps UC
                Nom du travail estimé  Prio Heure début Jusqu'à     Fréquence Limite Dépendances
```

```
Schedule NETAG #EXTERNAL      0
                E0000000      0
              Total    00:00
              Total    00:00

Schedule MYFTA #IWDSKE      10                       NETAG#EXTERNAL.E0000000
                JOBIWD      10         23:00(03/06/06) 01:00
              Total    00:00

Schedule MYMST #TESTSKE  00:29 10
                TESTCRO+  00:01 10
                NEWTEST  00:29 10 08:30(03/06/06)               TESTCROME
              Total    00:29


Calendrier MYMST #FINAL   00:00 10 05:59(03/07/06)
                JnextPlan  00:01 10
              Total    00:01
              Total    00:34

                        * * * *   Fin de l'état  * * * *
```

La sortie de l'Etat 9B montre ce qui doit être exécuté à la date sélectionnée dans l'environnement Tivoli Workload Scheduler. Les informations affichées proviennent des définitions stockées dans la base de données Tivoli Workload Scheduler. La sortie indique les flots de travaux dont l'exécution est prévue le 6 mars 2006, avec leur description, la liste des travaux qu'ils contiennent, les dépendances horaires, la fréquence de répétition, le nombre maximal de travaux (s'il est défini) et la dépendance par rapport à d'autres travaux ou flots de travaux. Par exemple, le flot de travaux nommé `iwdske` dont l'exécution est prévue sur `MYFTA` possède une dépendance `follows` sur le travail `NETAG#EXTERNAL.E0000000` dont l'exécution est prévue sur l'agent réseau nommé `NETAG`.

La zone **Heure de début** dans la sortie des états générés par la commande **reptr** affiche :

Une restriction de temps définie dans la définition de flot de travaux à l'aide du mot clé at.
Si la date est comprise entre des parenthèses (), par exemple :

```
Heure de début
06:00(03/20/06)
```

L'heure d'exécution planifiée du flot de travaux dans la définition de flot de travaux à l'aide du mot clé schedtime.
Si la date est comprise entre des accolades {}, par exemple :

```
Heure de début
06:00{03/20/06}
```

L'heure de début réelle d'exécution du flot de travaux.
Si la date n'est ni entre parenthèses, ni entre accolades, par exemple :

```
Heure de début
06:00 03/20/06
```

Etat 10B - Détail de la production réelle :

```
TWS for UNIX (AIX) REPORTER 8.3 (1.7)          ibm                            Page    1
Report 10B   Symphony           Actual Production Detail For 03/06/06          03/07/06

                      Estimated                          Actual    CPU    Job
             Job Name  Run Time Priority Start Time    Run Time Seconds Number  Status

Schedule NETAG #EXTERNAL             0                                          EXTRN
                E0000000             0                                          ERROR
```

```
                Total      00:00                          00:00    0

Schedule MYMST #MONTHSKE    00:02    10     06:01(03/06/06)  00:03                 SUCC
               GETLOGS      00:02    10     06:01(03/06/06)  00:03         #J11612 SUCC
                Total       00:02                          00:03    0

Schedule MYFTA #IWSKE                10                                          HOLD
               JOBIWD                10                                          HOLD
                Total       00:00                          00:00    0

Schedule MYMST #TESTSKE     00:29    10     06:01(03/06/06)  00:02                 STUCK
               TESTCRO+     00:01    10     06:01(03/06/06)  00:02         #J11613 ABEND
               NEWTEST      00:29    10                                          HOLD
                Total       00:30                          00:02    0

Schedule MYMST #FINAL       00:01    10     05:59(03/07/06)                       HOLD
               JnextPlan     00:01    10                                          HOLD
                Total       00:01                          00:00    0


                Total       01:38                          00:09    0

                          * * * *   Fin de l'état  * * * *
```

La sortie de l'Etat 10B indique les états des activités de planification en cours d'exécution sur le réseau Tivoli Workload Scheduler. Les informations affichées proviennent d'une copie du fichier Symphony qui est actuellement utilisé et mis à jour dans l'ensemble de l'environnement de planification. Ainsi, chaque fois que cette commande de génération d'état est exécutée au cours du traitement, les informations affichées reflètent le statut réel de l'activité planifiée.

Si vous comparez cette sortie à la sortie de l'Etat 9B, vous constatez que le flot de travaux MONTHSKE s'est exécuté pendant la journée de production en cours, le 6 mars, mais que son exécution n'est pas prévue le lendemain, 7 mars. Le flot de travaux EXTERNAL a échoué sur l'agent de réseau NETAG et, par conséquent, le flot de travaux IWSKE qui possède une dépendance de prédécesseur/successeur du flot de travaux EXTERNAL, reste à l'état HOLD.

Le flot de travaux TESTSKE se trouve à l'état STUCK, ce qui signifie que l'intervention de l'opérateur est requise du fait que, pendant le temps d'exécution du flot de travaux, le travail TESTCROME, après avoir démarré avec l'ID de travail J11613, a échoué à l'état ABEND et a entraîné le passage à l'état HOLD du travail dépendant NEWTEST.

Etat 11 - Calendrier de production planifiée :

```
TWS for UNIX (AIX)/REPORT11 8.3 (1.7)                      Page  1
Etat 11         Calendrier de production planifiée pour FEV 2006        03/08/065

Poste de travail : FTAWIN8+

         Nomb. Temps UC 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
Cal   travaux estimé Ma Me Je Ve Sa Di Lu Ma Me Je Ve Sa Di Lu Ma Me Je Ve Sa Di Lu Ma Me Je Ve Sa Di Lu
SCHED1  1    1                *

Un * entre le nom du calendrier et Nomb. travaux indique que le calendrier possède des travaux en
cours d'exécution sur d'autres postes de travail.
-------------------------------------------------------------------------------------------------
Temps UC estimé par jour en secondes
     Lun  Mar   Mer   Jeu   Ven   Sam   Dim

              1    2    3    4    5    6
              0    0    0    0    0    0
```

```
 7    8    9   10   11   12   13
 0    0    0    0    1    0    0

14   15   16   17   18   19   20
 0    0    0    0    0    0    0

21   22   23   24   25   26   27
 0    0    0    0    0    0    0

28
 0


TWS for UNIX (AIX)/REPORT11 8.3 (1.7)                      Page  2
Etat 11            Calendrier de production planifiée pour FEV 2006    03/08/06

Poste de travail : MASTER8+

         Nomb. Temps UC 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
Cal   travaux estimé Ma Me Je Ve Sa Di Lu Ma Me Je Ve Sa Di Lu Ma Me Je Ve Sa Di Lu Ma Me Je Ve Sa Di Lu
FINAL   1    4  * * * * * * * * * * * * * * * * * * * * * * * * * * * *

Un * entre le nom du calendrier et Nomb. travaux indique que le calendrier possède des travaux en
cours d'exécution sur d'autres postes de travail.
-------------------------------------------------------------------------------------------------------
Temps UC estimé par jour en secondes
     Lun  Mar   Mer   Jeu   Ven   Sam   Dim

                1    2    3    4    5    6
                4    4    4    4    4    4

 7    8    9   10   11   12   13
 4    4    4    4    4    4    4

14   15   16   17   18   19   20
 4    4    4    4    4    4    4

21   22   23   24   25   26   27
 4    4    4    4    4    4    4

28
 4


                    * * * *   Fin de l'état  * * * *
```

La sortie de l'Etat 11 indique le moment où l'exécution des flots de travaux est planifiée au cours du mois sélectionné. La première ligne contient le nombre de travaux que le flot de travaux contient, le temps UC estimé utilisé par le flot de travaux pour s'exécuter et le moment où l'exécution du flot de travaux est prévue. Le tableau indique pour chaque jour du mois sélectionné le temps UC estimé utilisé pour l'exécution de ce flot de travaux.

Etat 12 - Etat de références croisées :

La sortie de l'Etat 12 indique des informations différentes en fonction de l'indicateur utilisé lors de l'émission de la commande `xref`. Dans cette section, vous trouverez quelques exemples de sortie. Pour chacun de ces exemples, l'indicateur correspondant utilisé avec la commande xref est mis en évidence.

xref -when

```
TWS for UNIX (AIX)/CROSSREF 8.3 (1.7)         ibm                              Page    1
Etat 12 Etat de références croisées pour les options ON, EXCEPT(*) et FREEDAYS(f). 03/08/06

Poste de travail : FTAHP

WHEN                 Utilisé par les calendriers suivants :
REQUEST              TRFINAL



TWS for UNIX (AIX)/CROSSREF 8.3 (1.7)         ibm                              Page    2
Etat 12 Etat de références croisées pour les options ON, EXCEPT(*) et FREEDAYS(f). 03/08/06

Poste de travail : FTAWIN8+

WHEN                 Utilisé par les calendriers suivants :
MONTHEND             SCHED1
REQUEST              SCHED1  , SCHEDDAA



TWS for UNIX (AIX)/CROSSREF 8.3 (1.7)         ibm                              Page    3
Etat 12 Etat de références croisées pour les options ON, EXCEPT(*) et FREEDAYS(f). 03/08/06

Poste de travail : MASTER8+

WHEN                 Utilisé par les calendriers suivants :
EVERYDAY             FINAL
REQUEST              TMP


                       * * * *   Fin de l'état  * * * *
```

xref -jobs

```
TWS for UNIX (AIX)/CROSSREF 8.3 (1.7)         ibm                              Page    4
Etat 12                    Etat de références croisées pour les noms de travaux.            03/08/06

Poste de travail : FTAWIN8+

Nom du travail       Existe dans les calendriers
SCHEDDDD             SCHED1



TWS for UNIX (AIX)/CROSSREF 8.3 (1.7)         ibm                              Page    5
Etat 12                    Etat de références croisées pour les noms de travaux.            03/08/06

Poste de travail : MASTER8+

Nom du travail       Existe dans les calendriers
JnextPlan             FINAL
JOB1                 TMP


                       * * * *   Fin de l'état  * * * *
```

xref -resource

```
TWS for UNIX (AIX)/CROSSREF 8.3 (1.7)         ibm                              Page    8
Etat 12                   Etat de références croisées pour les utilisateurs de ressources.       03/08/06

Poste de travail : FTAWIN8+

Ressource            Utilisée par :
QUKTAPES(N/F)        SCHED1
```

```
TWS for UNIX (AIX)/CROSSREF 8.3 (1.7)         ibm                              Page    9
Etat 12                    Etat de références croisées pour les utilisateurs de ressources.       03/08/06

Poste de travail : MASTER8+

Ressource             Utilisée par :
TAPES(N/F)            TMP


                        * * * *   Fin de l'état  * * * *
```

xref -prompts

```
TWS for UNIX (AIX)/CROSSREF 8.3 (1.7)         ibm                              Page    6
Etat 12                  Etat de références croisées pour les dépendances d'invite.     03/08/06
Poste de travail : FTAWIN8+

Invite                Utilisée par :

Texte défini          SCHED1
par l'utilisateur



TWS for UNIX (AIX)/CROSSREF 8.3 (1.7)         ibm                              Page    7
Etat 12                  Etat de références croisées pour les dépendances d'invite.     03/08/06

Poste de travail : MASTER8+

Invite                Utilisée par :
BADEXIT               FTAWIN8+#SCHED1
GOODEXIT              FTAWIN8+#SCHED1  , TMP

Texte défini          TMP
par l'utilisateur


                        * * * *   Fin de l'état  * * * *
```

xref -files

```
TWS for UNIX (AIX)/CROSSREF 8.3 (1.7)         ibm                              Page   10
Etat 12                   Etat de références croisées pour les dépendances de fichier.      03/08/06

Poste de travail : MASTER8+

Nom du fichier        Utilisé par :
/root/MY_FILE.sh      FTAWIN8+#SCHED1  , TMP


                        * * * *   Fin de l'état  * * * *
```

Programmes d'extraction d'états

Des programmes d'extraction de données sont utilisés pour générer plusieurs des états Tivoli Workload Scheduler. Les programmes sont répertoriés dans le tableau 83 :

Tableau 83. Programmes d'extraction d'états

Programme d'extraction d'états	Description
jbxtract	Permet de générer l'état 01 - Liste des caractéristiques des travaux et l'état 07 - Liste des historiques des travaux

Tableau 83. Programmes d'extraction d'états (suite)

Programme d'extraction d'états	Description
prxtract	Permet de générer l'état 02 - Liste des invites
caxtract	Permet de générer l'état 03 - Liste des agendas
paxtract	Permet de générer l'état 04A - Liste des paramètres
rextract	Permet de générer l'état 04B - Liste des ressources
r11xtr	Permet de générer l'état 11 - Calendrier de production planifiée
xrxtrct	Permet de générer l'état 12 - Etat de références croisées

La sortie des programmes d'extraction est contrôlée par la variable ***MAESTRO_OUTPUT_STYLE*** qui définit la façon dont les noms d'objet longs sont traités. Pour plus d'informations sur la variable ***MAESTRO_OUTPUT_STYLE***, voir «Description des commandes», à la page 610.

jbxtract

Extrait de la base de données les informations relatives aux travaux.

Syntaxe

jbxtract [**-V** | **-U**]
[**-j** *travail*]
[**-c** *poste_travail*]
[**-o** *sortie*]

Arguments

-V Affiche la version de la commande et quitte l'application.

-U Affiche des informations sur la syntaxe de la commande et quitte l'application.

-j *travail*
Indique le travail pour lequel l'extraction est effectuée. Par défaut, tous les travaux sont pris en compte.

-c *poste_travail*
Indique le poste de travail des travaux pour lesquels l'extraction est effectuée. Par défaut, tous les postes de travail sont pris en compte.

-o *sortie*
Indique le fichier de sortie. La valeur par défaut est **stdout**.

Résultats

La variable ***MAESTRO_OUTPUT_STYLE*** spécifie le style de sortie pour les noms d'objet longs. Avec la valeur **LONG**, des zones longues sont utilisées pour les noms d'objet. Si la valeur affectée à cette variable est différente de **LONG**, les noms longs sont tronqués à huit caractères suivis du signe plus. Par exemple : `A1234567+`.

Chaque enregistrement de travail contient des zones de longueur variable délimitées par des tabulations. Ces zones sont décrites dans le tableau 84, à la page 630.

Tableau 84. Zones de sortie Jbxtract

Zone	Description	Longueur maximale (octets)
1	Nom du poste de travail	16
2	Nom du travail	16
3	Nom du fichier du script de travail	4096
4	Description du travail	65
5	Nom du travail de reprise	16
6	Option de reprise (0=stop, 1=rerun, 2=continue)	5
7	Texte de l'invite de reprise	64
8	Option d'auto-documentation (0=désactivée, 1=activée)	5
9	Nom de connexion de l'utilisateur pour le travail	36
10	ID utilisateur du créateur du travail	36
11	Nombre d'exécutions réussies	5
12	Nombre d'exécutions abandonnées	5
13	Durée totale de toutes les exécutions de travaux	8
14	Temps nécessaire au poste de travail pour toutes les exécutions des travaux	8
15	Temps moyen écoulé	8
16	Date de la dernière exécution (aammjj)	8
17	Heure de la dernière exécution (hhmm)	8
18	Secondes de la dernière exécution	8
19	Dernier temps écoulé	8
20	Durée (en secondes) de sollicitation du poste de travail	8
21	Temps écoulé maximal	8
22	Date d'exécution maximale (aammjj)	8
23	Sollicitation minimale du poste de travail en secondes	8
24	Temps minimal écoulé	8
25	Date d'exécution minimale (aammjj)	8

Remarque : Le temps écoulé affiché pour un travail reflet est le temps écoulé du travail distant lié.

Exemples

Pour extraire les informations sur le travail `mon_travail` du poste de travail `principal` et diriger les résultats dans le fichier `jinfo`, exécutez la commande suivante :

```
jbxtract -j mon_travail -c principal -o jinfo
```

prxtract

Extrait de la base de données les informations relatives aux invites.

Syntaxe

prxtract [**-V** | **-U**] [**-o** *sortie*]

Arguments

-V Affiche la version de la commande et quitte l'application.

-U Affiche des informations sur la syntaxe de la commande et quitte l'application.

-o *sortie*
Indique le fichier de sortie. La valeur par défaut est **stdout**.

Résultats

Chaque enregistrement d'invite contient des zones de longueur variable délimitées par des tabulations. Ces zones sont décrites dans le tableau 85.

Tableau 85. Zones de sortie Prxtract

Zone	Description	Longueur maximale (octets)
1	Nom de l'invite	8
2	Valeur de l'invite	200

Exemples

Pour extraire les informations sur toutes les définitions d'invite et diriger les résultats vers le fichier `prinfo`, exécutez la commande suivante :

```
prxtract -o prinfo
```

caxtract

Extrait de la base de données les informations relatives aux agendas.

Syntaxe

caxtract [**-V** | **-U**] [**-o** *sortie*]

Arguments

-V Affiche la version de la commande et quitte l'application.

-U Affiche des informations sur la syntaxe de la commande et quitte l'application.

-o *sortie*
Indique le fichier de sortie. La valeur par défaut est **stdout**.

Résultats

Chaque enregistrement d'agenda contient des zones de longueur variable, délimitées par des tabulations. Ces zones sont décrites dans le tableau 86, à la page 632.

Tableau 86. Zones de sortie Caxtract

Zone	Description	Longueur maximale (octets)
1	Nom de l'agenda	8
2	Description de l'agenda	64

Exemples

Pour extraire les informations sur toutes les définitions d'agenda et diriger les résultats vers le fichier `cainfo`, exécutez la commande suivante :

```
caxtract -o cainfo
```

paxtract

Permet d'extraire des informations relatives aux paramètres globaux (variables globales) de la base de données.

Syntaxe

paxtract [**-V** | **-U**] [**-o** *sortie*] [**-a**]

Arguments

-V Affiche la version de la commande et quitte l'application.

-U Affiche des informations sur la syntaxe de la commande et quitte l'application.

-o *sortie*
Indique le fichier de sortie. La valeur par défaut est **stdout**.

-a Permet d'afficher toutes les variables définies dans toutes les tables de variables. S'il n'est pas indiqué, seules les variables définies dans la table de variables par défaut s'affichent.

Résultats

Chaque enregistrement de variable contient des zones de longueur variable délimitées par des tabulations. Ces zones sont décrites dans le tableau 87.

Tableau 87. Zones de sortie Paxtract

Zone	Description	Longueur maximale (octets)
1	nom de table	80
2	nom de la variable	16
3	valeur de variable	72

A faire : Si vous ne spécifiez pas l'option **-a** (tout) dans la commande, seules les zones 2 et 3 s'affichent et les variables répertoriées sont celles uniquement contenues dans la table de variables par défaut.

Exemples

Pour extraire les informations relatives à toutes les définitions de variable et diriger les résultats vers le fichier `allvarinfo`, exécutez la commande suivante :

```
paxtract -a -o allvarinfo
```

rextract

Extrait de la base de données les informations relatives aux ressources.

Syntaxe

rextract [**-V** | **-U**] [**-o** *sortie*]

Arguments

-V Affiche la version de la commande et quitte l'application.

-U Affiche des informations sur la syntaxe de la commande et quitte l'application.

-o *sortie*
Indique le fichier de sortie. La valeur par défaut est **stdout**.

Résultats

Chaque enregistrement de ressource contient des zones de longueur variable délimitées par des tabulations. Ces zones sont décrites dans le tableau 88.

Tableau 88. Zones de sortie Rextract

Zone	Description	Longueur maximale (octets)
1	Nom du poste de travail	8/16
2	nom de la ressource	8
3	Unités de ressource totales	4
4	Description de la ressource	72

Exemples

Pour extraire les informations sur toutes les définitions de ressource et diriger les résultats vers le fichier `reinfo`, exécutez la commande suivante :

```
rextract -o reinfo
```

r11xtr

Extrait de la base de données les informations relatives aux flots de travaux.

Syntaxe

r11xtr [**-V** | **-U**]
[**-m** *mm*[*aaaa*]]
[**-c** *poste_travail*]
[**-o** *sortie*]
[**-s** *nom_flot_données*]

Arguments

-V Affiche la version du programme et quitte l'application.

-U Affiche des informations sur la syntaxe de la commande et quitte l'application.

-m *mm*[*aa*]
Indique le mois (*mm*) et éventuellement l'année (*aa*) des flots de travaux. La valeur par défaut est le mois et l'année en cours.

-c *poste_travail*
Indique le poste de travail à prendre en compte dans les états. Par défaut, tous les postes de travail sont pris en compte.

-s *nom_flot_travaux*
Indique le nom du flot de travaux dans lequel les travaux s'exécutent. Par défaut, tous les flots de travaux sont pris en compte.

-o *sortie*
Indique le fichier de sortie. La valeur par défaut est **stdout**.

Résultats

La variable ***MAESTRO_OUTPUT_STYLE*** spécifie le style de sortie pour les noms d'objet longs. Avec la valeur **LONG**, des zones longues sont utilisées pour les noms d'objet. Si la valeur affectée à cette variable est différente de **LONG**, les noms longs sont tronqués à huit caractères suivis du signe plus. Par exemple : `A1234567+`.

Chaque enregistrement de flot de travaux contient des zones de longueur variable délimitées par des tabulations. Ces zones sont décrites dans le tableau 89.

Tableau 89. Zones de sortie R11xtr

Zone	Description	Longueur maximale (octets)
1	Nom du poste de travail	16
2	Nom du flot de travaux	16
3	Date du flot de travaux (aammjj)	6
4	Durée estimée (en secondes) de sollicitation du poste de travail	6
5	Option plusieurs postes de travail (* signifie que certains travaux sont exécutés sur d'autres postes de travail)	1
6	Nombre de travaux	4
7	Jour de la semaine (Di, Lu, Ma, Me, Je, Ve, Sa)	2

Exemples

Pour extraire les informations sur les flots de travaux remontant à juin 2004 sur le poste de travail `principal`, exécutez la commande suivante :

```
r11xtr -m 0604 -c principal
```

Pour extraire les informations sur tous les flots de travaux remontant au mois de juin sur tous les postes de travail et diriger le résultat vers le fichier `r11out`, exécutez la commande suivante :

```
r11xtr -m 06 -o r11out
```

xrxtrct

Extrait de la base de données les informations relatives aux références croisées.

Syntaxe

xrxtrct [**-V** | **-U**]

Arguments

-V Affiche la version de la commande et quitte l'application.

-U Affiche des informations sur la syntaxe de la commande et quitte l'application.

Résultats

La variable *MAESTRO_OUTPUT_STYLE* spécifie le style de sortie pour les noms d'objet longs. Avec la valeur **LONG**, des zones longues sont utilisées pour les noms d'objet. Si la valeur affectée à cette variable est différente de **LONG**, les noms longs sont tronqués à huit caractères suivis du signe plus. Par exemple : `A1234567+`.

Le résultat de la commande est écrit dans huit fichiers, **xdep_job**, **xdep_sched**, **xfile**, **xjob**, **xprompt**, **xresources**, **xsched** et **xwhen**. Ces fichiers sont écrits dans le répertoire de travail en cours. Pour exécuter la commande, vous devez disposer du droit en écriture et des droits d'exécution sur ce répertoire.

Exemples

Pour extraire les informations relatives à toutes les références croisées, exécutez la commande suivante :

```
xrxtrct
```

Fichier xdep_job

Le fichier **xdep_job** contient deux types d'enregistrement. Le premier enregistrement contient les informations sur les travaux et les flots de travaux qui dépendent d'un travail. Chaque enregistrement d'un travail et d'un flot de travaux dépendant contient des zones de longueur fixe sans délimiteur. Ces zones sont décrites dans le tableau 90.

Tableau 90. Zones de sortie Xdep_job

Zone	Description	Longueur (octets)
1	**03**	2
2	Nom du poste de travail	16
3	Nom du travail	40
4	Nom du flot de travaux	16
5	Non utilisée	240
6	Nom du poste de travail du flot de travaux dépendant	16
7	Nom du flot de travaux dépendant	16
8	Nom du poste de travail du travail dépendant	16

Tableau 90. Zones de sortie Xdep_job (suite)

Zone	Description	Longueur (octets)
9	Nom du travail dépendant	40
10	Non utilisée	6
11	Non utilisée	6
12	Non utilisée	8
13	Fin d'enregistrement (null)	1

Le deuxième type d'enregistrement contient des informations sur les travaux et les flots de travaux soumis aux interdépendances de réseaux interconnectés. Chaque enregistrement de travail et de flot de travaux contient des zones de longueur fixe sans délimiteur. Ces zones sont décrites dans le tableau 91.

Tableau 91. Zones de sortie Xdep_job (suite)

Zone	Description	Longueur (octets)
1	**08**	2
2	Nom du poste de travail	16
3	Nom du travail	120
4	Non utilisée	128
5	Nom du poste de travail du flot de travaux dépendant	16
6	Nom du flot de travaux dépendant	16
7	Nom du poste de travail du travail dépendant	16
8	Nom du travail dépendant	40
9	Non utilisée	6
10	Non utilisée	6
11	Non utilisée	8
12	Fin d'enregistrement (null)	1

Fichier xdep_sched

Le fichier **xdep_sched** contient des informations sur les travaux et les flots de travaux qui dépendent d'un flot de travaux. Chaque enregistrement de travail ou de flot de travaux contient des zones de longueur fixe sans délimiteur. Ces zones sont décrites dans le tableau 92.

Tableau 92. Zones de sortie Xdep_sched

Zone	Description	Longueur (octets)
1	**02**	2
2	Nom du poste de travail	16
3	Nom du flot de travaux	16
4	Non utilisée	248
5	Nom du poste de travail du flot de travaux dépendant	16
6	Nom du flot de travaux dépendant	16
7	nom du poste de travail du travail dépendant	16
8	nom du travail dépendant	40
9	Non utilisée	6

Tableau 92. Zones de sortie Xdep_sched (suite)

Zone	Description	Longueur (octets)
10	Non utilisée	6
11	Non utilisée	8
12	Fin d'enregistrement (null)	1

Fichier xfile

Le fichier **xfile** contient des informations sur les travaux et les flots de travaux qui dépendent d'un fichier. Chaque enregistrement contient des zones de longueur fixe sans délimiteur. Ces zones sont décrites dans le tableau 93.

Tableau 93. Zones de sortie Xfile

Zone	Description	Longueur (octets)
1	**07**	2
2	Nom du poste de travail	16
3	nom de fichier	256
4	Nom du poste de travail du flot de travaux dépendant	16
5	Nom du flot de travaux dépendant	16
6	Nom du poste de travail du travail dépendant	16
7	Nom du travail dépendant	40
8	Non utilisée	6
9	Non utilisée	6
10	Non utilisée	8
11	Fin d'enregistrement (null)	1

Fichier xjob

Le fichier **xjob** contient des informations sur les flots de travaux dans lesquels figure chaque travail. Chaque enregistrement de travail contient des zones de longueur fixe sans délimiteur. Ces zones sont décrites dans le tableau 94.

Tableau 94. Zones de sortie Xjob

Zone	Description	Longueur (octets)
1	**04**	2
2	Nom du poste de travail	16
3	Nom du travail	40
4	Non utilisée	248
5	Nom du poste de travail des flots de travaux	16
6	Nom du flot de travaux	16
7	Non utilisée	8
8	Non utilisée	8
9	Non utilisée	6
10	Non utilisée	6
11	Non utilisée	8
12	Fin d'enregistrement (null)	1

Fichier xprompt

Le fichier **xprompt** contient des informations sur les travaux et les flots de travaux qui dépendent d'une invite. Chaque enregistrement d'invite contient des zones de longueur fixe sans délimiteur. Ces zones sont décrites dans le tableau 95.

Tableau 95. Zones de sortie Xprompts

Zone	Description	Longueur (octets)
1	**05**	2
2	Nom du poste de travail	16
3	Nom ou texte de l'invite	20
4	Non utilisée	236
5	Nom du poste de travail du flot de travaux dépendant	16
6	Nom du flot de travaux dépendant	16
7	Nom du poste de travail du travail dépendant	16
8	Nom du travail dépendant	40
9	Non utilisée	6
10	Non utilisée	6
11	Non utilisée	8
12	Fin d'enregistrement (null)	1

Fichier xresource

Le fichier **xresource** contient des informations sur les travaux et les flots de travaux qui dépendent d'une ressource. Chaque enregistrement de ressource contient des zones de longueur fixe sans délimiteur. Ces zones sont décrites dans le tableau 96.

Tableau 96. Zones de sortie Xresource

Zone	Description	Longueur (octets)
1	**06**	2
2	Nom du poste de travail	16
3	Nom de la ressource	8
4	Non utilisée	248
5	Nom du poste de travail du flot de travaux dépendant	16
6	Nom du flot de travaux dépendant	16
7	Nom du poste de travail du travail dépendant	16
8	Nom du travail dépendant	40
9	Unités allouées	6
10	Non utilisée	6
11	Non utilisée	8
12	Fin d'enregistrement (null)	1

Fichier xsched

Le fichier **xsched** contient des informations sur les flots de travaux. Chaque enregistrement de flot de travaux contient des zones de longueur fixe sans délimiteur. Ces zones sont décrites dans le tableau 97, à la page 639.

Tableau 97. Zones de sortie Xsched

Zone	Description	Longueur (octets)
1	**00**	2
2	Nom du poste de travail	16
3	Nom du flot de travaux	16
4	Non utilisée	248
5	Nom du poste de travail (identique à 2 ci-dessus)	16
6	Nom du flot de travaux (identique à 3 ci-dessus)	16
7	Non utilisée	8
8	Non utilisée	8
9	Non utilisée	6
10	Non utilisée	6
11	Non utilisée	8
12	Fin d'enregistrement (null)	1

Fichier xwhen

Le fichier **xwhen** contient des informations sur le moment auquel les flots de travaux seront exécutés. Chaque enregistrement de flot de travaux contient des zones de longueur fixe sans délimiteur. Ces zones sont décrites dans le tableau 98.

Tableau 98. Zones de sortie Xwhen

Zone	Description	Longueur (octets)
1	**01**	2
2	Nom du poste de travail	16
3	Nom ou date ON/EXCEPT	8
4	Option except (*=EXCEPT)	1
5	Non utilisée	128
6	Nom du poste de travail	16
7	Nom du flot de travaux	16
8	Non utilisée	8
9	Non utilisée	8
10	Non utilisée	6
11	Numéro de décalage	6
12	Unité de décalage	8
13	Fin d'enregistrement (null)	1

Exécution de rapports Dynamic Workload Console et de rapports de traitement par lots

Vous pouvez exécuter les rapports suivants à partir de Dynamic Workload Console :

Rapport d'historique d'exécution du travail
Rapport collectant les données historiques relatives à l'exécution des travaux pendant une période donnée. Il s'avère utile d'identifier les travaux qui se sont terminés avec une erreur ou les travaux en retard, ainsi que les

travaux critiques et promus et la dernière heure à laquelle le travail peut démarrer sans que le travail critique ne dépasse son échéance. Il contient également les travaux dont les délais n'ont pas été respectés ou qui ont duré plus longtemps que prévu, ainsi que les indicateurs de réexécution.

Rapport de statistiques d'exécution du travail
Rapport collectant les statistiques d'exécution du travail. Ce rapport permet de détecter les réussites, les taux d'erreur ; la durée minimale, maximale et moyenne ; les retards et les longues durées.

Rapport récapitulatif de la charge de travail du poste de travail
Rapport indiquant le charge de travail sur les postes de travail spécifiés. La charge de travail correspond au nombre de travaux qui se sont exécutés dessus. Ce rapport permet d'ajuster la planification de la capacité (modélisation de la charge de travail et réglage des postes de travail).

Rapport du temps d'exécution de la charge de travail du poste de travail
Rapport indiquant les heures et la durée d'exécution des travaux sur les postes de travail spécifiés. Ce rapport permet d'ajuster la planification de la capacité (modélisation de la charge de travail et réglage des postes de travail).

Rapport des détails de la production planifiée
Rapport basé sur les informations stockées dans un plan d'essai ou un plan prévisionnel. Les informations contenues dans ces plans proviennent de la base de données Tivoli Workload Scheduler. Un rapport détaillé de production planifiée peuvent être exécutés sur des moteurs distribués (gestionnaire de domaine maître et gestionnaire de domaine de sauvegarde). Un rapport de production réelle extrait d'un agent tolérant aux pannes peut contenir des informations différentes par rapport à un plan extrait d'un gestionnaire de domaine maître. Par exemple, pour un même nombre de travaux et de flots de travaux, le statut peut être différent car un travail exécuté correctement dans le domaine maître peut être suspendu ou prêt dans l'agent. Seul le taux de mise à jour du statut est identique sur l'agent en mode statut intégral qui s'exécute dans le domaine maître.

Rapport des détails de la production réelle
Rapport basé sur les informations stockées dans le plan en cours ou un plan archivé. Les informations contenues dans ces plans sont extraites des fichiers Symphony. Les rapports détaillés de production réelle peuvent être exécutés sur des moteurs distribués (gestionnaire de domaine maître, gestionnaire de domaine de sauvegarde, gestionnaire de domaine avec connecteur et agent tolérant aux pannes avec connecteur).

Rapport SQL personnalisé
Ce rapport vous permet de créer des rapports en exécutant vos propres requêtes SQL. Le rapport résultant contient un tableau comportant le nom de colonne spécifié dans la portion SELECT de l'instruction SQL. Les données sur lesquelles porte le rapport sont stockées dans une base de données relationnelle DB2 et résident du côté distribué. Tivoli Workload Scheduler for z/OS se connecte à la base de données au moyen de l'interface Java Database Connectivity (JDBC). Un pilote JDBC de type 4 est utilisé pour la connexion au système distant DB2 for LUW version 8.2, ou suivante.

Pour plus d'informations sur la définition et l'exécution de rapports depuis Dynamic Workload Console, voir Dynamic Workload Console - Guide d'utilisation (section sur la génération de rapports).

Certains de ces rapports sont également disponibles en tant que *rapports de traitement par lots* et peuvent être exécutés à partir d'une ligne de commande. Pour plus d'informations sur l'exécution des rapports de traitement par lots, voir «Exécution des rapports de traitement par lots à partir de l'interface de ligne de commande», à la page 646.

Selon l'interface à partir de laquelle vous exécutez le rapport ou le système d'exploitation du moteur, les formats de sortie suivants sont disponibles :

Tableau 99. Formats de sortie de rapport pris en charge

Nom du rapport	**Formats de sortie pris en charge par Dynamic Workload Console**	**Formats de sortie pris en charge par les rapports de traitement par lots**
Rapport d'historique d'exécution du travail	HTML, CSV Format de tableau uniquement	HTML, CSV, PDF Format de tableau uniquement
Rapport de statistiques d'exécution du travail	HTML, CSV Formats de tableau et de diagramme	HTML, CSV, PDF Formats de tableau et de diagramme
Rapport récapitulatif de la charge de travail du poste de travail	HTML, CSV Formats de tableau et de diagramme	HTML, CSV, PDF Formats de tableau et de diagramme
Rapport du temps d'exécution de la charge de travail du poste de travail	HTML, CSV Formats de tableau et de diagramme	HTML, CSV, PDF Formats de tableau et de diagramme
Rapport des détails de la production planifiée	XML, CSV Format de tableau uniquement	Indisponible
Rapport des détails de la production réelle	XML, CSV Format de tableau uniquement	Indisponible
Rapport SQL personnalisé	HTML, CSV Format de tableau uniquement	HTML, CSV, PDF Format de tableau uniquement
Rapport d'audit général (Pour plus d'informations, voir les informations relatives au suivi des modifications de base de données à l'aide des rapports d'audit dans *Guide d'administration*)	Indisponible	HTML, CSV, PDF Format de tableau uniquement
Rapport d'audit général (Pour plus d'informations, voir les informations relatives au suivi des modifications de base de données à l'aide des rapports d'audit dans *Guide d'administration*)	Indisponible	HTML, CSV, PDF Format de tableau uniquement

Pour pouvoir exécuter ces rapports, vous devez être titulaire des autorisations appropriées dans le fichier de sécurité des objets de rapport (accordées par défaut

à *utilisateur_tws* sur les nouvelles installations). Voir *Guide d'administration* pour obtenir des informations sur le fichier de sécurité.

Voir également le *Guide d'administration* pour apprendre à configurer Dynamic Workload Console afin de visualiser les rapports.

Rapports historiques

Le tableau ci-après récapitule les rapports historiques par :

- Fonctionnalité
- Critères de sélection
- Options du contenu de sortie

Tableau 100. Récapitulatif des rapports historiques

Nom du rapport	Description	Critères de sélection	Options du contenu de sortie
Historique d'exécution du travail	Correspond au **Rapport 07**. Collecte les données historiques relatives à l'exécution des travaux pendant un intervalle de temps donné. Permet d'obtenir les éléments suivants : • Travaux s'étant achevés avec des erreurs • Travaux en retard • Echéances dépassées • Longue durée • Indicateurs de réexécution • Autres informations historiques.	• Nom de travail, nom de flot de travaux, nom de poste de travail (flot de travaux). Chaque zone peut être spécifiée à l'aide d'un caractère générique. • Statut (Succès, Erreur, Inconnu) • Indicateurs de retard • Intervalle d'exécution du travail • Inclure/exclure les réexécutions	Vous pouvez sélectionner l'une des options suivantes : • Heure de début réelle • Durée estimée • Durée réelle • Numéro de travail • Démarré en retard (retard) • Arrêté en retard (retard) • Statut • Dernier démarrage critique • Critique • Promu • Longue durée • Nom de la définition du travail • Consommation CPU (indisponible sur les postes de travail Windows) • Connexion utilisateur • Type de réexécution • Numéro d'itération • Code retour La sortie est contenue dans la vue tabulaire.

Tableau 100. Récapitulatif des rapports historiques (suite)

Nom du rapport	Description	Critères de sélection	Options du contenu de sortie
Statistiques d'exécution du travail	Correspond au **Rapport 01** Collecte les statistiques relatives à l'exécution des travaux. Permet d'obtenir les éléments suivants : • Taux de succès/erreur • Temps UC écoulé maximal et minimal • Durée moyenne • Statistiques d'exécution tardive et de longue durée **Remarque :** Ce rapport ne contient pas les travaux ayant été soumis via un nom d'alias.	• Nom de travail, nom de flot de travaux et nom de connexion utilisateur. Chaque zone peut être spécifiée à l'aide d'un caractère générique. • Pourcentage de travaux ayant le statut Succès, Erreur, Démarré en retard, Arrêté en retard et Longue durée • Nombre total d'exécutions et de réexécutions	Vous pouvez sélectionner l'une des options suivantes : • Détails sur le travail : – Connexion utilisateur – Créateur du travail – Description – Script – Informations de reprise • Statistiques du travail : – Total des exécutions (Exécutions terminées, Erreurs) – Total d'exceptions d'exécution (Démarré en retard, Terminé en retard, Longue durée) – Durées et temps UC minimal, maximal et moyen (pour les exécutions ayant abouti uniquement) – Consommation CPU (indisponible sur les postes de travail Windows) • Format du rapport : – Vue Graphiques – Vue Table – Inclure la table des matières par travail ou par poste de travail

Tableau 100. Récapitulatif des rapports historiques (suite)

Nom du rapport	Description	Critères de sélection	Options du contenu de sortie
Récapitulatif de la charge de travail du poste de travail	Fournit des données sur la charge de travail suivant le nombre de exécutés sur chaque poste de travail. Permet d'ajuster les prévisions de charge (modélisation de la charge de travail et optimisation des postes de travail).	• Noms des postes de travail. Chaque zone peut être spécifiée à l'aide d'un caractère générique. • Plages de dates ou jours spécifiques pour le filtrage de la charge de travail. • Intervalles de temps relatifs (permet de réutiliser la même tâche de rapport afin d'exécuter celui-ci quotidiennement et extraire le rapport correspondant à la production de la veille)	Vous pouvez sélectionner l'une des options suivantes : • Cohérence des informations sur les postes de travail organisées d'après les critères suivants : – Heure – Jour – Jour de production • Options de regroupement d'informations : – Fournit des informations récapitulatives cumulatives et agrégées, relatives à tous les postes de travail ou à un sous-ensemble défini de celles-ci • Format du rapport : – Vue Graphiques – Vue Table – Inclure la table des matières par date ou par poste de travail

Tableau 100. Récapitulatif des rapports historiques (suite)

Nom du rapport	Description	Critères de sélection	Options du contenu de sortie
Temps d'exécution de la charge de travail du poste de travail	Correspond au **Rapport 08**. Fournit des données sur les exécutions de travaux (heure et durée) sur le poste de travail. Permet d'ajuster les prévisions de charge (modélisation de la charge de travail et optimisation des postes de travail).	• Noms de travail et de poste de travail. Chaque zone peut être spécifiée à l'aide d'un caractère générique. • Période d'exécution de la charge de travail • Intervalle de temps quotidien	Vous pouvez sélectionner l'une des options suivantes : • Informations regroupées par : – Poste de travail – Date d'exécution Peut être trié par réexécution • Jour de production • Information du travail – Durée réelle – Statut – Réexécution – Nom de la définition du travail • Format du rapport : – Vue Graphiques – Vue Table
SQL personnalisé	Un assistant vous aide à définir vos requêtes SQL personnalisées (uniquement dans les vues de la base de données pour lesquelles vous disposez d'une autorisation d'accès).	Les critères indiqués dans la requête SQL personnalisée.	Le rapport résultant contient un tableau comportant le nom de colonne spécifié dans la portion SELECT de l'instruction SQL.

Rapports de production

Le tableau ci-après récapitule les rapports de production par :

- Fonctionnalité
- Critères de sélection
- Options du contenu de sortie

Tableau 101. Récapitulatif des rapports de production

Nom du rapport	Description	Critères de sélection	Options du contenu de sortie
Détail de la production réelle	Correspond au **Rapport 10B**. Fournit des données sur les plans en cours et archivés.	• Nom du travail • Nom du poste de travail (travail) • Nom du flot de travaux • Nom du poste de travail (flot de travaux)	Vous pouvez sélectionner l'une des options suivantes : • Format du rapport : – A plat – CSV – Microsoft Project • Inclut : – Prédécesseur de premier niveau – Journal du travail
Détail de la production planifiée	Correspond au **Rapport 9B**. Fournit des données sur les plans d'essai et les plans prévisionnels.	• Nom du travail • Nom du poste de travail (travail) • Nom du flot de travaux • Nom du poste de travail (flot de travaux)	Vous pouvez sélectionner l'une des options suivantes : • Format du rapport : – A plat – CSV – Microsoft Project • Inclut : – Prédécesseur de premier niveau – Journal du travail

Exécution des rapports de traitement par lots à partir de l'interface de ligne de commande

Cette section décrit comment exécuter les rapports répertoriés dans «Rapports historiques», à la page 642 à partir de la ligne de commande.

A l'aide d'une interface de ligne de commande, vous pouvez planifier ces rapports afin qu'ils s'exécutent en temps voulu.

Modèle de scénario métier

Pour éviter le ralentissement inattendu du traitement de la charge de travail, l'analyste d'une grande entreprise a besoin de rapports hebdomadaires collectant les informations d'historique sur la charge de travail traitée afin de déterminer et d'analyser les heures pleines éventuelles.

Pour répondre à cette demande, le TWSWEBUIDeveloper crée des rapports récapitulatifs de poste de travail de charge de travail (*Workload Workstation Summary Reports* (WWS)) et des rapports d'exécution de poste de travail de charge de travail (*Workload Workstation Runtimes Reports* (WWR)).

Pour accomplir cette tâche, il exécute les étapes suivantes :

1. Il personnalise les fichiers de propriétés associés aux rapports Workload Workstation Summary et Workload Workstation Runtimes, en spécifiant le format et le contenu de la sortie de rapport.

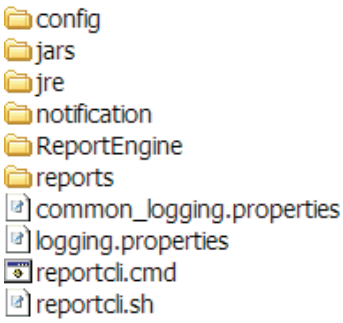
2. Il planifie les travaux pour obtenir des rapports WWS et WWR :
   - Le premier travail génère un rapport WWS à sauvegarder localement.
   - Le second travail génère un rapport WWR pendant la nuit sur les périodes horaires d'heures pleines de charge de travail. La sortie de rapport est envoyée à l'analyste par mail. Les informations collectées permettent d'optimiser l'équilibre de la charge de travail sur les systèmes.
3. Il ajoute les deux travaux à un flot de travaux planifié pour s'exécuter chaque semaine et ajoute le plan.

Configuration de la génération de rapports de ligne de commande

Avant d'exécuter des rapports de traitement par lots, vous devez exécuter quelques étapes de configuration :

1. Le logiciel nécessaire pour exécuter des rapports de traitement par lots est contenu dans un module nommé `TWSBatchReportCli`, contenu dans l'image d'installation Tivoli Workload Scheduler, dans le répertoire `TWSBatchReportCli`. Si vous prévoyez d'exécuter des rapports de traitement par lots à partir d'un travail planifié, procédez à l'extraction du fichier de module sur l'un des systèmes d'exploitation répertoriés dans Document relatif à la configuration requise, à l'adresse http://www-01.ibm.com/support/docview.wss?rs=672&uid=swg27041008.

   Après avoir décompressé le module, vous disposerez de la structure de fichiers suivante :

   L'utilitaire natif d'archivage sur bande UNIX ne prenant pas en charge les longs noms de fichier, veillez à ce que la version GNU la plus récente d'archivage sur bande (gtar) soit installée si vous souhaitez extraire avec succès les fichiers sur AIX, Solaris ou les systèmes HP-UX.

   Remarque :

   a. Vérifiez que vous exécutez les commandes suivantes dans le répertoire où vous avez extrait les fichiers :

      Sur UNIX

      ```
      chmod -R +x *
      chown -R nom_utilisateur *
      ```

      Sur Windows

      Assurez-vous que Tivoli Workload Scheduler est installé.

      ```
      setown -u nom_utilisateur *
      ```

      où *nom_utilisateur* est l'utilisateur Tivoli Workload Scheduler qui exécutera les rapports.

b. Si vous prévoyez de planifier des travaux qui exécutent des rapports de traitement par lots, le système sur lequel vous extrayez le module doit être accessible en tant que système de fichiers réseau à partir d'un agent tolérant aux pannes défini dans l'environnement de planification local.

2. Configurez le fichier modèle `.\config\common.properties` en spécifiant les informations pour :
   a. Connectez-vous à la base de données sur laquelle les données d'historique sont stockées.
   b. Définir le format de date et d'heure, y compris le fuseau horaire. Le fichier `.\config\timezone.txt` contient une liste des fuseaux horaires pris en charge par Tivoli Workload Scheduler et des informations indiquant comment les définir. Les noms des fuseaux horaires sont sensibles à la casse.
   c. Rendez le résultat de rapport disponible sur l'URL spécifiée dans la zone **ContextRootUrl**. Voici un exemple de paramètres de configuration :

```
#####################################################################
 # Informations de serveur HTTP
 #####################################################################

 #Spécifiez la racine de contexte où le rapport sera disponible
 #Pour tirer parti de cette fonction, il faut spécifier dans le répertoire de sortie
de rapport
 #le répertoire auquel fait référence votre serveur HTTP avec cette racine de
contexte

 ContextRootUrl=http://myserver/reportoutput
```

   Dans ce cas, vérifiez que le *rép_rapport_sortie* spécifié lors de l'exécution de la commande des rapports de traitement par lots pointe vers le répertoire spécifié dans la zone **ContextRootUrl**.
   d. Envoyez la sortie de rapport à l'aide d'un courriel. Voici un exemple de paramètres de configuration :

```
#####################################################################
 # Configuration du serveur de messagerie
 #####################################################################
 PARAM_SendReportByEmail=true

 #SMTP server
 mail.smtp.host=myhost.mydomain.com
 #IMAP provider
 mail.imap.socketFactory.fallback=false
 mail.imap.port=993
 mail.imap.socketFactory.port=993
 #POP3 provider
 mail.pop3.socketFactory.fallback=false
 mail.pop3.port=995
 mail.pop3.socketFactory.port=995

 ######################################################################
 # Propriétés de courriel
 ######################################################################
 PARAM_EmailFrom=user1@your_company.com
 PARAM_EmailTo=user2@your_company.com,user3@your_company.com
 PARAM_EmailCC=user4@your_company.com
 PARAM_EmailBCC=user5@your_company.com
 PARAM_EmailSubject=Test send report by email
 PARAM_EmailBody=This is the report attached
```

   Une explication de toutes les zones personnalisables est contenue dans le modèle de fichier.

Remarque : Si vous prévoyez d'exécuter les rapports Workstation Workload Runtime, vérifiez que le système de fichier dans lequel la base de données est installée possède assez d'espace libre. Si un manque d'espace disque se produit, une exception SQL du type suivant est déclenchée :

```
DB2 SQL error: SQLCODE: -968, SQLSTATE: 57011
```

Exécution des rapports de traitement par lots

Le répertoire `\reports\templates` contient un fichier modèle pour chaque type de rapport.

Avant d'exécuter l'un de ces rapports, vérifiez que vous personnalisez le fichier modèle correspondant.

Dans ce fichier appelé *nom_rapport*`.properties`, vous pouvez spécifier :

- Les informations à afficher dans l'en-tête du rapport.
- Comment filtrer les informations pour afficher le résultat attendu.
- Le format et le contenu de la sortie de rapport.

Pour plus d'informations sur les paramètres spécifiques, voir l'explication fournie dans le modèle de fichier à côté de chaque zone.

Si vous utilisez un jeu de caractères codé sur deux octets pour indiquer les paramètres dans le modèle de fichier de propriétés, veillez à enregistrer le fichier en codage UTF-8.

Après avoir configuré l'environnement tel qu'il est décrit dans «Configuration de la génération de rapports de ligne de commande», à la page 647, et avoir configuré le modèle de fichier rapport, utilisez la syntaxe suivante pour exécuter le rapport :

reportcli -p *nom_rapport.property*
[**-o** *rép_rapport_sortie*]
[**-r** *nom_sortie_rapport*]
[**-k** key=*valeur*]
[**-k** key=*valeur*]
.....

où :

-p *nom_rapport.property*
Indique le nom du chemin vers le fichier modèle de rapport.

-o *rép_rapport_sortie*
Indique le répertoire de sortie du résultat de rapport.

-r *nom_sortie_rapport*
Indique le nom de la sortie de rapport.

-k key=*valeur*
Indique la valeur d'un paramètre. Cette valeur remplace la valeur correspondante, si elle est définie, dans le fichier `common.properties` ou dans le fichier *nom_rapport*`.properties`.

Exemples

1. Dans cet exemple, `reportcli.cmd` est exécuté avec le paramètre par défaut et génère le rapport `jrh1` :

   ```
   reportcli.cmd -p D:\ReportCLI\TWSReportCli\reports\templates\jrh.properties
   -r jrh1
   ```

2. Dans cet exemple, reportcli.cmd est exécuté avec le paramètre -k pour remplacer les valeurs définies pour **PARAM_DateFormat** dans le fichier .\config\common.properties et génère le rapport jrh1 :

   ```
   reportcli.cmd -p D:\ReportCLI\TWSReportCli\reports\templates\jrh.properties
   -r jrh2 -k PARAM_DateFormat=short
   ```

3. Dans cet exemple, reportcli.cmd est exécuté à l'aide du paramètre -k pour remplacer le format spécifié pour la sortie de rapport dans le fichier PROPERTIES et génère le rapport jrh1 :

   ```
   ./reportcli.sh -p /TWSReportCli/REPCLI/reports/templates/wwr.properties
   -r wwr3 -k REPORT_OUTPUT_FORMAT=html -k OutputView=charts
   ```

4. Procédez comme suit si vous souhaitez exécuter un rapport SQL personnalisé et rendre disponible le résultat du rapport à l'URL suivante http://myserver/reportoutput/report1.html :

   a. Configurez le paramètre ContextRootUrl dans le fichier common.properties comme suit :

      ```
      ######################################################################
       # Informations de serveur HTTP
       ######################################################################

       #Spécifiez la racine de contexte où le rapport sera disponible
       #Pour tirer parti de cette fonction, il faut spécifier dans le répertoire de sortie
      de rapport
       #le répertoire auquel fait référence votre serveur HTTP avec cette racine de
      contexte

       ContextRootUrl=http://myserver/reportoutput
      ```

   b. Lorsque vous utilisez une commande de rapports de traitement par lots, indiquez comme *rép_rapport_sortie* un répertoire qui pointe vers le répertoire indiqué dans ContextRootUrl. Par exemple, si vous avez mappé localement http://myserver/ comme pilote R:, vous pouvez exécuter la commande suivante :

      ```
      reportclibat
         -p REPORT_CLI_DIR\reports\TWS\historical\templates\sql.properties
         -r report1
         -o R:\reportoutput
      ```

   c. Confirmant que l'exécution du rapport a réussi, le message suivant s'affiche :

      ```
      AWSBRC0106I
      Rapport disponible sur : http://myserver/reportoutput/report1.html
      ```

      Cette URL indique l'emplacement où la sortie de rapport est disponible.

Remarque : Si le rapport est exécuté via un travail Tivoli Workload Scheduler, le résultat de la commande est affiché dans la sortie de travail.

Journaux et traces des rapports de traitement par lots

Le fichier ./common_logging.properties contient les paramètres que vous pouvez utiliser pour configurer la fonction de trace et la journalisation.

Ce fichier contient les paramètres suivants :

```
logFileName=reportcli.log
traceFileName=trace.log
trace=off
birt_trace=off
```

où :

logFileName
Spécifie le nom du fichier contenant des informations génériques, un avertissement sur les problèmes potentiels et des informations sur les erreurs. Ce fichier est stocké dans `./log`.

traceFileName
Spécifie le nom du fichier qui contient les traces. Si vous définissez `trace=on`, le fichier de trace est stocké dans `./log`.

trace Spécifie d'activer ou non les traces. Activez les traces en définissant `trace=on` si vous souhaitez approfondir une enquête sur une erreur

birt_trace
Spécifie d'activer ou non les traces pour diagnostiquer les erreurs dans le moteur BIRT. Si vous définissez `birt_trace=on`, un fichier contenant la trace et nommé `ReportEngine_aaaa_mm_dd_hh_mm_ss.log` est stocké dans le dossier `/ReportEngine/logs`

Chapitre 16. Gestion des fuseaux horaires

Tivoli Workload Scheduler prend en charge plusieurs fuseaux horaires. Si vous activez les fuseaux horaires, vous pouvez gérer votre charge de travail dans un environnement à fuseaux horaires multiples.

Les deux notations (à 3 caractères et de longueur variable) sont prises en charge.

La notation à 3 caractères est prise en charge pour la compatibilité avec les versions antérieures de Tivoli Workload Scheduler.

Le format de notation à longueur variable est de type région/ville, par exemple, Europe/Paris, qui est l'équivalent d'ECT (European Central Time).

Le chapitre est composé des sections suivantes :

Activation de la gestion des fuseaux horaires

Vous pouvez activer ou désactiver la gestion des fuseaux horaires en modifiant le paramètre attribué à l'option globale ***enTimeZone*** sur le gestionnaire de domaine maître à l'aide de la ligne de commande **optman**. Ce paramètre prend effet après l'exécution suivante de **JnextPlan**. Vous trouverez ci-dessous les paramètres disponibles :

No Désactivation de la gestion des fuseaux horaires. En d'autres termes, les valeurs attribuées à tous les mots clés **timezone** dans les définitions sont ignorées. Toutes les restrictions de temps définies par `at`, `until` et `deadline` sont gérées individuellement par chaque agent tolérant aux pannes, y compris les gestionnaires de domaine, en ignorant par conséquent le fuseau horaire de l'agent qui planifie le travail ou flot de travaux. Il en résulte que, lorsque plusieurs fuseaux horaires sont impliqués :

- Pour les travaux, des informations incorrectes sont indiquées sur ces dépendances horaires lorsqu'elles sont considérées du point de vue d'un agent autre que le propriétaire de travail. Toutefois, ceci n'a aucune incidence sur le traitement du travail.
- Pour les flots de travaux, cela implique que chaque agent traite les dépendances horaires en fonction de son propre fuseau horaire, et donc à des heures différentes, ce qui entraîne l'exécution à des heures différentes des travaux appartenant à un même flot, mais définis sur des agents différents.

Yes Activation de la gestion des fuseaux horaires. En d'autres termes, les valeurs attribuées aux paramètres **timezone** sont utilisées pour calculer l'heure où les travaux et les flots de travaux s'exécuteront sur les postes de travail cibles.

Par défaut, la valeur **yes** est attribuée à l'option *enTimeZone*.

Pour plus d'informations sur l'utilisation de la ligne de commande **optman** pour gérer les options globales sur le gestionnaire de domaine maître, voir le manuel *IBM Tivoli Workload Scheduler - Guide d'administration*.

Comment Tivoli Workload Scheduler gère les fuseaux horaires

Lorsque le fuseau horaire est activé, vous pouvez utiliser les paramètres de fuseau horaire sur les définitions du poste de travail, du travail et du flot de travaux.

En exécutant les activités de gestion de plan, Tivoli Workload Scheduler convertit la valeur définie pour les fuseaux horaires en définitions de l'objet. Les conversions sont appliquées dans cet ordre :

1. Lorsque les instances de flot de travaux sont ajoutées au plan de préproduction, le fuseau horaire figurant dans les définitions de flot de travaux est converti en fuseau horaire GMT et les dépendances de prédécesseur/successeur externes sont résolues.
2. Lorsque le plan de préproduction est créé ou étendu, les instances de flot de travaux sont affectées aux postes de travail sur lesquels l'instance est programmée pour s'exécuter et le fuseau horaire est converti de GMT vers le fuseau horaire figurant dans la définition du poste de travail cible.

C'est la raison pour laquelle si vous utilisez les commandes **conman showsched** ou **conman showjobs** pour afficher les informations sur les travaux et flots de travaux planifiés, vous verrez les valeurs de fuseau horaire exprimées avec le fuseau horaire du poste de travail sur lequel le travail ou le flot de travaux doit s'exécuter. Selon la configuration de l'option globale *enLegacyStartOfDayEvaluation*, vous pouvez choisir la manière dont le produit va gérer les fuseaux horaires lors du traitement, et plus précisément :

Si la valeur *no* **est attribuée à** *enLegacyStartOfDayEvaluation*

La valeur affectée à l'option *startOfDay* sur le gestionnaire de domaine maître n'est pas convertie au fuseau horaire local défini sur chaque poste de travail du réseau. Par conséquent, si l'option *startOfDay* est définie sur 0600 sur le gestionnaire de domaine maître, il est 0600 sur le fuseau horaire local défini sur chaque poste de travail du réseau. Ceci signifie également que le jour de traitement commence à la même heure, mais pas au même moment, sur tous les postes de travail.

La figure 28, à la page 655 montre comment le début de la journée défini sur 0600 sur le gestionnaire de domaine maître, est appliqué aux différents fuseaux horaires sur les deux agents tolérants aux pannes. La même conversion de temps est appliquée aux trois instances du flot de travaux **JS1** planifié pour s'exécuter sur les trois machines et contenant une dépendance de temps **at** sur le fuseau horaire `0745 US/Central`. Le cadre temporel qui identifie le nouveau jour de traitement est en grisé dans la figure 28, à la page 655.

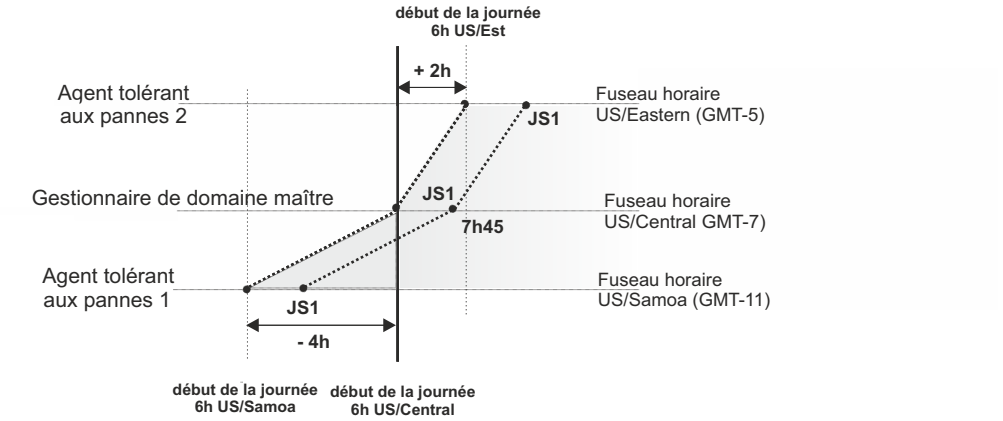


Figure 28. Exemple où la conversion de début de journée n'est pas appliquée

Si la valeur *yes* **est attribuée à** *enLegacyStartOfDayEvaluation*

La valeur affectée à l'option *startOfDay* sur le gestionnaire de domaine maître est convertie au fuseau horaire local défini sur chaque poste de travail du réseau. Ceci signifie que si l'option *startOfDay* est définie sur 0600 sur le gestionnaire de domaine maître, elle est convertie sur chaque poste de travail vers le temps correspondant du fuseau horaire local défini sur ce poste de travail. Ceci signifie également que le jour de planification commence au même moment, mais pas nécessairement à la même heure sur chaque poste de travail du réseau.

La figure 29 montre comment le début de la journée défini sur 0600 sur le gestionnaire de domaine maître, est appliqué aux différents fuseaux horaires sur les deux agents tolérants aux pannes. Cette figure montre également comment le délai des trois instances du flot de travaux **JS1** planifié pour s'exécuter sur les trois machines et contenant une dépendance de temps **at** sur le fuseau horaire `0745 US/Central` n'est pas modifiée du fait de la conversion *startOfDay*. Le cadre temporel qui identifie le nouveau jour de traitement est en grisé dans la figure 29.

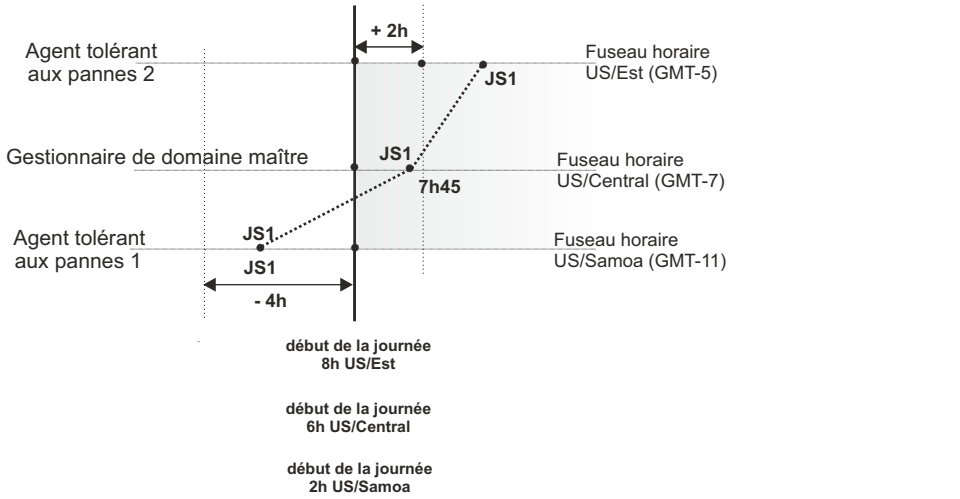


Figure 29. Exemple où la conversion de début de journée est appliquée

Remarque : A partir de la version 8.3 il n'y a pas de liaison entre le temps défini pour l'option *startOfDay* et le moment où **JnextPlan** est exécuté. **JnextPlan** peut

être exécuté à tout moment et l'option *startOfDay* indique uniquement le moment où le nouveau jour de traitement démarre.

Par défaut, la valeur **no** est attribuée à l'option globale *enLegacyStartOfDayEvaluation*.

Pour plus d'informations sur l'utilisation de la ligne de commande **optman** pour gérer les options globales sur le gestionnaire de domaine maître, voir le manuel *IBM Tivoli Workload Scheduler - Guide d'administration*.

Passage à l'heure d'été

Tivoli Workload Scheduler gère le passage à l'heure d'été lors de la génération du plan de production. La date et l'heure d'exécution assignées aux travaux et flots de travaux du plan sont donc déjà converties vers la date et l'heure correspondant à l'heure d'été.

L'exemple suivant explique comment Tivoli Workload Scheduler applique la conversion de temps lorsque **JnextPlan** est exécuté pour générer ou étendre le plan de production pendant le passage à l'heure d'été.

Si l'heure d'été est activée à 3:00, tous les flots de travaux planifiés pour commencer entre 2:00 et 2:59 sont définis pour démarrer une heure plus tard. Cette condition est due à la phase planifiée de Tivoli Workload Scheduler. Par exemple, un travail défini pour s'exécuter à 2:30 du matin s'exécutera à 03:30.

Désactivation de l'heure d'été

Pour la désactivation de l'heure d'été, l'horloge est reculée d'une heure par rapport à l'heure d'été. Pour maintenir la cohérence avec les critères de planification de production, Tivoli Workload Scheduler vérifie que les instances de flot de travaux planifiées pour s'exécuter pendant l'heure précédant le changement d'heure sont exécutées uniquement une fois. Parce que la conversion de temps est appliquée lors de la génération ou de l'extension du plan de production, la date et l'heure d'exécution affectées aux travaux et flots de travaux du plan sont déjà converties en date et heure correspondant à la désactivation de l'heure d'été.

Si un flot de travaux ou un travail s'exécute sur un fuseau horaire pour lequel l'heure d'été (DST) est désactivée, c'est-à-dire lorsque l'horloge est reculée d'une heure, et que vous définissez une dépendance à un autre fuseau horaire pour ces flux de travaux ou ces travaux, il est possible que cette dépendance horaire se produise lors du second intervalle de temps répété. Dans ce cas, la dépendance de temps sera résolue lors du premier intervalle de temps.

Tivoli Workload Scheduler reconnaît que la dépendance de temps se produit sur le second intervalle de temps répété et le résout en conséquence.

Règles générales

Lorsque le fuseau horaire est activé dans l'environnement Tivoli Workload Scheduler, quelle que soit la valeur définie pour l'option *enLegacyStartOfDayEvaluation*, certaines règles générales sont appliquées. Ces règles sont décrites ci-après, par sujet :

Identifier les paramètres de fuseau horaire par défaut pour les travaux et flots de travaux :

Vous pouvez définir un fuseau horaire dans une définition de flot de travaux pour l'ensemble du flot de travaux et pour les travaux contenus dans le flot de travaux. Ces fuseaux horaires peuvent être différents les uns des autres. Pour gérer tous les paramètres de fuseau horaire possibles la conversion de fuseau horaire est effectuée en respectant les critères suivants :

- Si aucun fuseau horaire n'est défini pour un travail contenu dans un flot de travaux, ce travail hérite du fuseau horaire défini sur le poste de travail où il est censé s'exécuter.
- Si aucun fuseau horaire n'est défini pour un flot de travaux, le fuseau horaire utilisé correspond à celui déjà défini sur le poste de travail où le flot de travaux est censé s'exécuter.
- Si aucun des fuseaux horaires mentionnés n'est défini, le fuseau horaire utilisé est celui défini sur le gestionnaire de domaine maître.

Choisir le fuseau horaire correct pour les postes de travail :

Pour éviter les incohérences, avant d'activer la fonction de gestion des fuseaux horaires sur le réseau Tivoli Workload Scheduler, vérifiez que, si un fuseau horaire est spécifié dans la définition de poste de travail, il est identique au fuseau horaire défini sur le système où est installé ce poste de travail.

Réglage de fuseau horaire par défaut pour le gestionnaire de domaine maître :

Si un fuseau horaire n'est pas défini dans la définition du gestionnaire de domaine maître, il hérite du fuseau horaire défini sur le système où est installé le gestionnaire de domaine maître. Pour voir quel fuseau horaire est défini sur le gestionnaire de domaine maître vous pouvez exécuter la commande suivante :

```
conman showcpu;info
```

Utilisation du fuseau horaire sur les agents étendus :

Les agents étendus héritent du fuseau horaire du gestionnaire de domaine maître.

Affichage du paramètre de fuseau horaire en production pour une dépendance de temps AT :

Si vous utilisez les commandes **conman**, **sj** ou **ss** pour afficher un travail ou un flot de travaux ayant une dépendance de temps **at** avec un fuseau horaire défini, l'heure spécifiée pour la dépendance **at** s'affiche avec le fuseau horaire défini sur le poste de travail où le travail ou le flot de travaux doit s'exécuter.

Appliquer un décalage à un fuseau horaire lors de la planification d'un flot de travaux :

Si vous soumettez en production un flot de travaux spécifiant une dépendance **at** avec un décalage de +n jour, Tivoli Workload Scheduler ajoute d'abord le décalage à la date puis convertit le fuseau horaire défini dans la dépendance **at** . Ceci est important particulièrement en faisant référence à l'heure lors du passage à l'heure d'été.

A titre de meilleure pratique, si vous activez la gestion de fuseau horaire, définissez un fuseau horaire sur chaque poste de travail de votre réseau Tivoli Workload Scheduler.

Chapitre 17. Définitions des méthodes d'accès des agents

Les méthodes d'accès permettent d'étendre les fonctions de planification de travaux de Tivoli Workload Scheduler aux autres systèmes et applications. Elles s'exécutent sur les systèmes suivants :

Agents étendus
Postes de travail logiques associés à une méthode d'accès hébergée par un poste de travail Tivoli Workload Scheduler physique (pas pour un agent étendu). Plusieurs postes de travail d'agent étendu peuvent être hébergés par le même poste de travail Tivoli Workload Scheduler et utiliser la même méthode d'accès. L'agent étendu s'exécute sur des agents tolérants aux pannes définis via une définition de poste de travail Tivoli Workload Scheduler standard, qui lui donne un nom et identifie la méthode d'accès. La méthode d'accès est un programme exécuté par le poste de travail hôte chaque fois que Tivoli Workload Scheduler soumet un travail à un système externe.

Des travaux sont définis pour un agent étendu de la même manière que pour les autres postes de travail Tivoli Workload Scheduler, sauf que les attributs des travaux sont dictés par l'application ou le système externe.

Des informations sur l'exécution des travaux sont envoyées à Tivoli Workload Scheduler à partir d'un agent étendu à l'aide du fichier `stdlist` du travail. Un fichier d'options de méthode peut spécifier des connexions de remplacement pour lancer des travaux et vérifier les dépendances de fichier *opens*. Pour plus d'informations, voir *Tivoli Workload Scheduler - Guide d'utilisation et de référence*.

Un poste de travail physique peut héberger 255 agents étendus au maximum.

agents dynamiques et agents Tivoli Workload Scheduler for z/OS
Ils communiquent avec les systèmes externes pour démarrer le travail et renvoyer son statut. Pour exécuter des méthodes d'accès sur les applications externes à l'aide d'agents dynamiques, vous devez définir un travail de type **méthode d'accès**.

Des méthodes d'accès sont disponibles sur les systèmes et applications suivants.
- Oracle E-Business Suite
- SAP R/3
- z/OS
- Méthodes personnalisées
- unixssh
- unixrsh
- UNIX en local (agents tolérants aux pannes uniquement)

Pour plus d'informations sur les méthode d'accès UNIX incluses avec Tivoli Workload Scheduler, voir Méthodes d'accès UNIX.

Si vous utilisez agents dynamiques, pour plus d'informations sur la définition des postes de travail Tivoli Workload Scheduler, voir la section qui explique comment définir des postes de travail dans la base de données dans *Tivoli Workload Scheduler: User's Guide and Reference*. Pour plus d'informations sur l'écriture de méthodes

d'accès, voir la section relative à l'interface de méthode d'accès dans *Tivoli Workload Scheduler - Guide d'utilisation et de référence*.

Interface de méthode d'accès

L'interface entre Tivoli Workload Scheduler et une méthode d'accès est composée des informations transmises à la méthode par la ligne de commande et des messages renvoyés à Tivoli Workload Scheduler dans **stdout**.

Syntaxe de la ligne de commande des méthodes

L'hôte de Tivoli Workload Scheduler exécute une méthode d'accès à l'aide de la syntaxe de ligne de commande suivante :

nom_méthode **-t** *tâche options* **--** *chaîne_tâche*

où :

nom_méthode
Désigne le nom de fichier de la méthode d'accès. Toutes les méthodes d'accès doivent figurer dans le répertoire : *racine_TWS*/methods

-t *tâche*
Désigne la tâche à effectuer, où *tâche* correspond à l'une des tâches suivantes :

LJ Lance un travail.

MJ Gère un travail lancé précédemment. Cette option permet de se resynchroniser si une tâche **LJ** précédente s'est arrêtée de manière inattendue.

CF Agents étendus uniquement. Vérifie la disponibilité d'un fichier. Cette option permet de vérifier les dépendances `opens` de fichier.

GS Agents étendus uniquement. Obtient le statut d'un travail. Cette option permet de vérifier les dépendances `follows` de fichier.

options Désigne les options associées à la tâche. Pour plus d'informations, voir «Options de tâche».

chaîne_tâche
Une chaîne comportant jusqu'à 255 caractères associée à la tâche. Voir «Options de tâche».

Options de tâche

Les options de tâche figurent dans le tableau 102. Un X signifie que l'option est admise pour la tâche.

Tableau 102. Options de tâche des commandes des méthodes

Tâche	-c	-n	-p	-r	-s	-d	-l	-o	-j	-q	-w	-S	Chaîne de la tâche
LJ	X	X	X	X	X	X	X	X	X			X	*chaîne_lj*
MJ	X	X	X	X	X	X	X	X	X				*chaîne_mj*
CF	X	X	X							X			*chaîne_cf*
GS	X	X	X	X		X					X		*chaîne_gs*

-c *agent_étendu*,*hôte*,*maître*
Indique les noms de l'agent, de l'hôte et du gestionnaire de domaine maître séparés par des virgules.

-n *nom_noeud*
Indique le nom de noeud de l'ordinateur associé à l'agent, le cas échéant. Ce nom est défini dans la zone **Noeud** de la définition de poste de travail de l'agent étendu.

-p *numéro_port*
Indique le numéro de port TCP/IP associé à l'agent, le cas échéant. Cette information est définie dans la zone d'**adresse TCP** de la définition de poste de travail d'agent.

-r *exécution_courante*,*exécution_spécifique*
Désigne le numéro d'exécution en cours de Tivoli Workload Scheduler et le numéro d'exécution spécifique associé au travail, séparés par une virgule. Les numéros d'exécution en cours et spécifiques peuvent être distincts si le travail a été reporté depuis une exécution antécédente.

-s *flot_travaux*
Désigne le nom du flot de travaux du travail.

-d *date_planification*,*période*
Désigne la date du flot de travaux (*aammjj*) ainsi que la période équivalente, séparées par une virgule.

-l *utilisateur*
Indique le nom d'utilisateur du travail. Ce nom est défini dans la zone **Connexion** de la définition du travail.

-o *liste_std*
Désigne le chemin d'accès complet du fichier de liste standard du travail. Toute sortie émanant de ce travail doit figurer dans ce fichier.

-j *nom_travail*,*ID*
Indique le nom du travail et l'identificateur unique affecté par Tivoli Workload Scheduler, séparés par une virgule. Ce nom est défini dans la zone **Nom du travail** de la définition du travail.

-q *qualifiant*
Désigne le qualifiant qui doit être utilisé dans une commande de test émise par une méthode pour un fichier.

-w *délai*
Indique le temps, exprimé en secondes, pendant lequel Tivoli Workload Scheduler attend une réponse sur un travail externe avant d'envoyer un signal SIGTERM à la méthode d'accès. La valeur par défaut est 300.

-S *nouveau_nom*
Indique que le travail est réexécuté à l'aide de ce nom, à la place du nom de travail d'origine. Dans le script d'un travail, vous pouvez utiliser la commande `jobinfo` pour renvoyer le nom du travail et exécuter le script différemment à chaque itération.

-- *chaîne_lj*
Utilisé avec la tâche **LJ**. La chaîne provenant de la zone **Fichier Script** ou la zone **Commande** de la définition du travail.

-- *chaîne_mj*
Utilisé avec la tâche **MJ**. Il s'agit des informations fournies à Tivoli Workload Scheduler par la méthode dans un message indiquant un

changement d'état du travail **%CJ** (pour plus d'informations sur les messages de ce type, voir «Messages de réponse des méthodes») suite à une tâche **LJ**. Généralement, elles identifient le travail qui a été lancé. Par exemple, une méthode UNIX peut fournir le processus d'identification (PID) du travail qu'elle a lancé et qui est ensuite envoyé par Tivoli Workload Scheduler en tant que partie d'une tâche **MJ**.

-- *chaîne_cf*
Utilisé avec la tâche **CF**. Pour une dépendance `opens` de fichier, correspond à la chaîne figurant dans la zone **Opens Files** de la définition du flot de travaux.

-- *chaîne_gs*
Utilisé avec la tâche **GS**. Désigne le travail dont le statut est vérifié. Le format est le suivant :

followsjob[**,***jobid*]

où :

followsjob
Correspond à la chaîne figurant dans la liste **Follows Sched/Job** de la définition de flot de travaux.

id_travail
Correspond à un identificateur de travail facultatif reçu par Tivoli Workload Scheduler dans une réponse **%CJ** à une tâche **GS** précédente.

Messages de réponse des méthodes

Les méthodes renvoient des informations à Tivoli Workload Scheduler sous la forme de messages écrits dans **stdout**. Toutes les lignes commençant par le symbole de pourcentage (%) et finissant par une nouvelle ligne sont considérées comme un message. Les messages présentent le format suivant :

%CJ *statut* [*chaîne_mj* | *id_travail*]

%JS [*temps_UC*]

%RC *cr*

%UT [*message_erreur*]

où :

CJ Change le statut du travail.

état Statut par lequel le travail est changé. Tous les statuts de travaux de Tivoli Workload Scheduler sont valides à l'exception de HOLD et READY. Pour la tâche **GS**, les statuts suivants sont également valides :

ERROR
Une erreur s'est produite.

EXTRN
Le statut est inconnu.

chaîne_mj
Chaîne pouvant comporter jusqu'à 255 caractères, que Tivoli Workload Scheduler inclut dans toute tâche **MJ** associée au travail.

id_travail
Chaîne pouvant comporter jusqu'à 64 caractères, que Tivoli Workload Scheduler inclut dans toute tâche **GS** associée au travail.

JS [*temps_UC***]**
Indique qu'un travail a abouti et fournit la durée de son exécution en secondes.

RC *rc* *cr* est un nombre qui est interprété par Tivoli Workload Scheduler comme étant le code retour du travail de l'agent étendu. Le code retour est pris en compte seulement si la condition du code retour à été indiquée dans la définition du travail de l'agent étendu. Dans le cas contraire, il est ignoré et un message apparaît pour indiquer que le travail de l'agent étendu a abouti. **%JS** *[temps_poste_de_travail]*. De la même manière, si cette méthode n'envoie pas le message **%RC**, un message apparaît pour indiquer que le travail de l'agent étendu a abouti **%JS** *[temps_UC]*.

UT [*message_erreur***]**
Indique que la tâche requise n'est pas supportée par la méthode. Affiche une chaîne pouvant comporter jusqu'à 255 caractères, que Tivoli Workload Scheduler inclura dans son message d'erreur.

Fichier d'options de méthode

Pour les agents étendus, les agents et l'agent Tivoli Workload Scheduler for z/OS, vous pouvez, à l'aide d'un fichier d'options de méthode, fournir des informations de connexion et indiquer d'autres options.

Un fichier d'options est un fichier texte situé dans le répertoire methods de l'installation de Tivoli Workload Scheduler et contenant un ensemble d'options destinées à personnaliser le comportement de la méthode d'accès. Vous devez indiquer une option par ligne en respectant le format suivant (n'insérez pas d'espace) :

```
option=valeur
```

Toutes les méthodes d'accès utilisent deux types de fichier d'options :

Agents étendus

Fichier d'options globales
Fichier de configuration commun créé par défaut pour chaque méthode d'accès installée, dont les paramètres s'appliquent à tous les postes de travail d'agent étendu définis pour cette méthode. Lors de sa création, le fichier d'options globales contient uniquement l'option **LJuser**, qui correspond à l'ID utilisateur du système d'exploitation utilisé pour lancer la méthode d'accès. Vous pouvez personnaliser ce fichier en ajoutant les options correspondant à la méthode d'accès.

Fichier d'options locales
Fichier de configuration spécifique à chaque poste de travail d'agent étendu au sein de l'installation particulière d'une méthode d'accès. Ce fichier est nommé *NOM_AGENT_ETENDU_méthode_accès*`.opts`, où :

NOM_AGENT_ETENDU
Correspond au nom du poste de travail d'agent étendu. La valeur de *NOM_AGENT_ETENDU* doit être indiquée en majuscules et en caractères alphanumériques. Le jeu de caractères codé sur deux octets, le jeu de caractères à simple octet et le texte bidirectionnel ne sont pas pris en charge.

méthode_accès
Correspond au nom de la méthode d'accès.

Si vous ne créez pas de fichier d'options locales, le système utilise par défaut le fichier d'options globales. Chaque poste de travail d'agent étendu (sauf pour z/OS) doit avoir un fichier d'options locales avec ses propres options de configuration.

Par exemple, si l'installation de la méthode d'accès inclut deux postes de travail d'agent étendu, `CPU1` et `CPU2`, les fichiers d'options locales sont respectivement nommés `CPU1`*`méthode_accès`*`.opts` et `CPU2`*`méthode_accès`*`.opts`.

Tivoli Workload Scheduler lit le fichier d'options s'il existe avant d'exécuter une méthode d'accès. Pour les agents étendus, si le fichier d'options est modifié après le démarrage de Tivoli Workload Scheduler, les changements sont pris en compte uniquement lors de l'arrêt et du redémarrage du programme.

agents Tivoli Workload Scheduler for z/OS et agents

Fichier d'options globales
Fichier de configuration commun créé par défaut pour chaque méthode d'accès installée, dont les paramètres s'appliquent à tous les postes de travail d'agent définis pour cette méthode. Lors de sa création, le fichier d'options globales contient uniquement l'option **LJuser**, qui correspond à l'ID utilisateur du système d'exploitation utilisé pour exécuter la méthode d'accès. Vous pouvez personnaliser ce fichier en ajoutant les options correspondant à la méthode d'accès.

Le fichier d'options globales est nommé *`méthode_accès`*`.opts`, où méthode_accès est le nom de la méthode en cours de création.

Fichier d'options locales
Fichier de configuration spécifique à chaque méthode d'accès. Ce fichier est nommé *`fichier_options_méthode_accès`*`.opts`.

Dans un environnement distribué :

- Si vous définissez un travail pour exécuter la méthode d'accès via Dynamic Workload Console, il s'agit du fichier d'options que vous avez spécifié dans l'onglet de tâche d'agent étendu **Nouveau** > **Définition du travail** > **ERP** > **Méthode d'accès** Tâche XA.
- Si vous définissez la méthode d'accès via **composer**, il s'agit du fichier d'options que vous avez spécifié dans l'attribut **target** de la définition de travail.

Si vous ne créez pas de fichier d'options locales, le système utilise par défaut le fichier d'options globales.

Dans un environnement z/OS :

- Si vous définissez un travail pour exécuter la méthode d'accès via Dynamic Workload Console, il s'agit du fichier d'options que vous avez spécifié dans l'onglet de tâche d'agent étendu **Nouveau** > **ERP** > **Méthode d'accès** Tâche XA.
- Si vous définissez la méthode d'accès via l'instruction **JOBREC**, il s'agit du nom du poste de travail où s'exécute la méthode d'accès.

Si vous ne créez pas de fichier d'options locales, le système utilise par défaut le fichier d'options globales.

Si vous n'indiquez pas d'option dans le fichier *méthode_accès_fichier_options*.opts, le produit utilise la valeur spécifiée pour cette option dans le fichier d'options globales. Si vous ne les indiquez ni dans le fichier *méthode_accès_fichier_options*.opts, ni dans le fichier d'options globales, le produit émet un message d'erreur.

Le fichier d'options doit présenter le même nom de chemin que ses méthodes d'accès, avec l'extension de fichier .opts. Par exemple, le nom du chemin d'accès Windows au fichier d'options de la méthode netmeth est

racine_TWS\methods\netmeth.opts

Tivoli Workload Scheduler lit le fichier d'options s'il existe avant d'exécuter une méthode d'accès.

Les options reconnues par Tivoli Workload Scheduler sont les suivantes :

LJuser=*nom_utilisateur*
Désigne la connexion à utiliser pour les tâches **LJ** et **MJ**. La valeur par défaut correspond à la connexion figurant dans la définition du travail. Pour plus d'informations, voir «Tâche de lancement d'un travail (LJ)», à la page 666 et «Tâche de gestion du travail (MJ)», à la page 667.

CFuser=*nom_utilisateur*
Agents étendus uniquement. Désigne la connexion à utiliser pour la tâche **CF**. La valeur par défaut pour UNIX est **root** et pour Windows, le nom d'utilisateur du compte dans lequel le produit a été installé. Voir awsrgcheckfiletask.dita.

GSuser=*nom_utilisateur*
Désigne la connexion à utiliser pour les tâches **GS**. La valeur par défaut pour UNIX est **root**, et pour Windows le nom d'utilisateur du compte dans lequel Tivoli Workload Scheduler a été installé. Voir Tâche d'obtention d'un statut (GS) - agents étendus uniquement.

GStimeout=*secondes*
Indique le temps, exprimé en secondes, pendant lequel Tivoli Workload Scheduler attend une réponse avant de supprimer la méthode d'accès. La valeur par défaut est **300** secondes.

nodename=*nom_noeud*
Indique le nom d'hôte ou l'adresse IP si la méthode en cours de définition l'exige. Pour la méthode d'accès **unixssh**, il s'agit du nom d'hôte ou de l'adresse IP utilisé(e) pour la connexion au moteur distant.

PortNumber=*numéro_port*
Indique le numéro de port si la méthode en cours de définition l'exige. Pour la méthode d'accès **unixssh**, il s'agit du port utilisé pour la connexion au moteur distant.

Pour les agents Tivoli Workload Scheduler for z/OS et les agents, vous pouvez indiquer le nom de noeud et le numéro de port également dans le fichier `JobManager.ini`.

Si vous ne les indiquez pas dans le fichier *méthode_accès_fichier_options*.`opts`, le produit utilise la valeur spécifiée dans le fichier d'options globales. Si vous ne les indiquez ni dans le fichier *méthode_accès_fichier_options*.`opts`, ni dans le fichier d'options globales, le produit utilise la valeur spécifiée dans la section *fichier_options* du fichier `JobManager.ini`.

Remarque : Si un hôte d'agent étendu est un ordinateur Windows, ces utilisateurs doivent être définis comme des objets utilisateur Tivoli Workload Scheduler.

Exécution des méthodes

Les sous-sections suivantes décrivent les échanges entre Tivoli Workload Scheduler et une méthode d'accès.

Tâche de lancement d'un travail (LJ)

La tâche **LJ** charge la méthode de l'agent étendu de lancer un travail sur une application ou un système externe. Avant d'exécuter la méthode, Tivoli Workload Scheduler crée un environnement d'exécution. Le paramètre **LJuser**, lu à partir du fichier d'options de la méthode, indique avec quel compte utilisateur la méthode doit être exécutée. Si le paramètre n'est pas présent ou que le fichier d'options n'existe pas, le compte utilisateur qui sera utilisé correspond à celui défini dans la zone **Connexion** de la définition du travail. En outre, les variables d'environnement suivantes sont définies :

HOME
Répertoire racine de l'utilisateur de la connexion.

LOGNAME
Nom d'utilisateur de la connexion.

PATH Pour UNIX, il s'agit du répertoire `/bin:/usr/bin`. Pour Windows, il s'agit du répertoire `%SYSTEM%\SYSTEM32`.

TWS_PROMOTED_JOB
Prend la valeur `YES` si le travail (un travail critique ou l'un de ses prédécesseurs) est promu.

TZ Fuseau horaire

Si la méthode ne peut pas être exécutée, l'état du travail devient FAIL.

Lorsqu'une méthode est en cours d'exécution, elle écrit des messages à son **stdout** pour indiquer le statut du travail sur le système externe. Les messages sont résumés dans le tableau 103, à la page 667.

Tableau 103. Messages de la tâche de lancement d'un travail (LJ)

Tâche	Réponse de la méthode	Action de Tivoli Workload Scheduler
LJ et **MJ**	**%CJ** *statut* [*chaîne_mj*]	Affecte l'état *état* au travail. Inclut *chaîne_mj* dans toutes les tâches **MJ** subséquentes.
	%JS [*temps_UC*]	Affecte l'état SUCC au travail.
	Exit code=non-zero	Affecte l'état ABEND au travail.
	%UT [*message_erreur*] et Exit code=2	Affecte l'état ABEND au travail et affiche *message_erreur*.

Une séquence typique est constituée d'un ou de plusieurs messages **%CJ** indiquant un changement dans l'état du travail, puis un message **%JS** avant que la méthode n'existe indiquant que le travail a abouti. Si le travail n'aboutit pas, la méthode doit exister sans écrire le message **%JS**. Une méthode qui ne supporte pas la tâche **LJ** écrit un message **%UT** vers **stdout** et renvoie le code de sortie **2**.

Tâche de gestion du travail (MJ)

La tâche **MJ**assure la synchronisation avec un travail précédemment lancé dans le cas où Tivoli Workload Scheduler détermine que la tâche **LJ** s'est achevée de façon inattendue. Tivoli Workload Scheduler configure l'environnement de la même façon que pour la tâche **LJ** et lui transmet *chaîne_mj*. Pour plus d'informations, voir «Tâche de lancement d'un travail (LJ)», à la page 666.

Si la méthode localise le travail indiqué, elle renvoie les mêmes messages qu'une tâche **LJ**. Si la méthode ne parvient pas à localiser le travail, elle quitte l'application avec un code de sortie différent de zéro. Tivoli Workload Scheduler affecte alors l'état ABEND au travail.

Suppression d'un travail

Lors de l'exécution d'une tâche **LJ** ou **MJ**, la méthode doit capter un signal SIGTERM (signal **15**). Ce signal est envoyé lorsqu'un opérateur lance une commande **kill** à partir du gestionnaire de console de Tivoli Workload Scheduler. A la réception du signal, la méthode doit essayer de supprimer (**kill**) le travail et sortir sans qu'un message **%JS** ne soit généré.

Tâche de vérification d'un fichier (CF) - agents étendus uniquement

La tâche **CF** charge la méthode de l'agent étendu de vérifier la disponibilité d'un fichier sur le système externe. Avant d'exécuter la méthode, Tivoli Workload Scheduler crée un environnement d'exécution. Le paramètre **CFuser**, lu à partir du fichier d'options de la méthode, indique avec quel compte utilisateur la méthode doit être exécutée. Si le paramètre ne figure pas dans le fichier d'options ou si celui-ci n'existe pas, le programme utilise l'utilisateur **root** (sous UNIX) et le nom de l'utilisateur du compte sur lequel Tivoli Workload Scheduler a été installé (sous Windows). Si la méthode ne peut pas être exécutée, la dépendance `opens` du fichier est marquée comme échouée. En fait, le programme définit le statut du fichier sur **NO** et ne peut pas exécuter les travaux ou flots de travaux dépendants.

Pendant la phase d'exécution, la méthode exécute une commande test ou son équivalent sur le fichier utilisant le qualifiant qui lui a été transmis via l'option de ligne de commande **-q**. Si la valeur du fichier test est true, la méthode existe avec

un code de sortie 0. Si la valeur du fichier test est false, la méthode existe avec un code de sortie différent de 0. Pour plus d'informations, voir tableau 104.

Tableau 104. Messages de la tâche de vérification d'un fichier (CF)

Tâche	Réponse de la méthode	Action de Tivoli Workload Scheduler
CF	Exit code=0	Définit le statut du fichier à **YES**.
	Exit code=nonzero	Définit le statut du fichier à **NO**.
	%UT [*message_erreur*] et Exit code=2	Définit le statut du fichier à **NO**.

Une méthode qui ne supporte pas la tâche **CF** écrit un message **%UT** vers **stdout** et sort en renvoyant le code de sortie **2**.

Tâche d'obtention d'un statut (GS) - agents étendus uniquement

La tâche **GS** charge la méthode de l'agent étendu de vérifier le statut d'un travail. Cette opération est nécessaire lorsque l'exécution d'un autre travail dépend du succès de l'exécution d'un travail externe. Avant l'exécution de la méthode, le paramètre **GSuser** est lu à partir du fichier d'options de la méthode pour déterminer le compte utilisateur avec lequel la méthode doit être exécutée. Si le paramètre n'est pas présent ou si le fichier des options n'existe pas sous UNIX l'utilisateur **root** est utilisé, et sous Windows, le nom d'utilisateur du compte dans lequel Tivoli Workload Scheduler a été installé. Si la méthode ne peut pas être lancée, le travail ou le flot de travaux dépendant ne peut pas être exécuté. Si l'*id_travail* d'une précédente tâche **GS** est disponible, il est transmis à la méthode.

La méthode vérifie le statut du travail indiqué et le renvoie dans un message **%CJ** écrit vers **stdout**. Elle existe alors avec le code de sortie 0. Conformément à la fréquence définie par l'option locale ***bm check status***, la méthode est réexécutée avec une tâche **GS** jusqu'à ce que l'un des statuts de travail suivants soit renvoyé :

abend Le travail s'est arrêté de manière anormale.

succ Le travail a abouti.

cancl Le travail a été annulé.

done Le travail est terminé, sans que l'on sache s'il a abouti ou échoué.

fail Le travail n'a pas pu être exécuté.

erreur Une erreur s'est produit dans la méthode lors de la vérification du statut du travail.

extrn La vérification du travail a échoué ou le statut du travail n'a pas pu être déterminé.

Notez que **GStimeout** dans le fichier d'options de la méthode indique le temps pendant lequel Tivoli Workload Scheduler attend une réponse avant de supprimer la méthode. Pour plus d'informations, voir «Fichier d'options de méthode», à la page 663.

Les réponses de la méthode sont résumées dans le tableau 105, à la page 669 :

Tableau 105. Messages de la tâche d'obtention d'un état (GS)

Tâche	Réponse de la méthode	Action de Tivoli Workload Scheduler
GS	**%CJ** *statut* [*jobid*]	Définit l'état du travail à *statut* et inclut *id_travail* dans toute tâche **GS** suivante.
	%UT [*message_erreur*] et Exit code=2	L'état du travail est inchangé.

Une méthode qui ne prend pas en charge la tâche **GS** écrit un message **%UT** à **stdout** et sort avec le code de sortie **2**.

Commande Cpuinfo pour agents étendus uniquement

La commande **cpuinfo** peut être utilisée dans une méthode d'accès pour renvoyer des informations à partir d'une définition de poste de travail. Pour plus d'informations sur la commande complète, voir «Commande Cpuinfo pour agents étendus uniquement».

Identification des incidents

Les sections suivantes décrivent comment identifier et déboguer des problèmes survenus au niveau d'agents étendus et de méthodes d'accès.

Messages d'erreur de la liste standard d'un travail

Tous les messages en sortie émanant d'une méthode d'accès, exceptés ceux commençant par un symbole de pourcentage (%), sont écrits dans le fichier de la liste standard (**stdlist**) du travail. Pour les tâches **GS** et **CF** qui ne sont pas associées aux travaux Tivoli Workload Scheduler, les messages sont écrits dans le fichier de liste standard Tivoli Workload Scheduler. Pour obtenir des informations relatives à tout problème, reportez-vous à ces fichiers.

Méthode non exécutable

Si une méthode d'accès ne peut pas être exécutée, les événements suivants se produisent :

- Pour des tâches **LJ** et **MJ**, le travail se voit affecter l'état FAIL.
- Pour la tâche **CF**, la dépendance de fichier n'est pas résolue et le travail dépendant reste à l'état HOLD.
- Pour la tâche **GS**, la dépendance de travail n'est pas résolue et le travail dépendant reste à l'état HOLD.

Pour plus d'informations, consultez la liste des fichiers de liste standard (**stdlist**) concernant le travail et Tivoli Workload Scheduler.

Messages du gestionnaire de console pour agents étendus uniquement

Ce message d'erreur est affiché si vous émettez une commande **start**, **stop**, **link**, ou **unlink** pour un agent étendu :

```
AWSBHU058E La commande émise pour le poste de travail : nom_poste_travail,
           ne peut pas être exécutée parce que le poste de travail est un agent étendu,
           sur lequel la commande n'est pas prise en charge.
```

Messages des programmes Composer et Compiler pour agents étendus uniquement

Le programme génère les messages d'erreur suivants dès que le programme **composer** rencontre une syntaxe incorrecte dans une définition de poste de travail :

```
AWSDEM045E Il y a une erreur dans la définition du poste de travail. Le mot clé ACCESS
           n'a pas été suivi par une méthode valide. Les méthodes valides correspondent au
           nom d'un fichier dans le répertoire des méthodes TWS_home/
           (le fichier n'a pas besoin d'être présent lorsque la méthode d'accès est définie).

AWSDEM046E Il y a une erreur dans la définition du poste de travail. Le mot clé ACCESS
           a été indiqué plusieurs fois.

AWSDEM047E Il y a une erreur dans la définition du poste de travail. Le mot clé ACCESS
           n'a pas été suivi par une méthode valide. Les méthodes valides correspondent au
           nom d'un fichier dans le répertoire des méthodes TWS_home/
           (le fichier n'a pas besoin d'être présent lorsque la méthode d'accès est définie).
```

Si un agent étendu est défini avec une méthode d'accès mais sans hôte, le message suivant s'affiche :

```
AWSBIA140E Pour un agent étendu, vous devez spécifier l'hôte et la méthode d'accès.
```

Messages de Jobman pour agents étendus uniquement

Pour les agents étendus, les messages d'information, d'avertissement ou d'erreur sont consignés dans le fichier **stdlist** de **jobman**.

Un travail dont le lancement a abouti génère le message suivant :

```
AWSBDW019I Launched job nom_travail, #Jnum_exécution pour l'utilisateur ID_utilisateur.
```

Un travail dont le lancement n'a pas abouti génère le message suivant :

```
AWSBDW057E Le travail job_name n'a pas été lancé pour le motif suivant :
           message_erreur
```

L'échec d'une tâche de vérification de fichier génère le message suivant :

```
AWSBDW062E Jobman n'a pas réussi à appeler le fichier de méthode suivant nom_méthode
           pour l'agent étendu. L'erreur du système d'exploitation est :
           erreur_système
```

L'échec d'une tâche de gestion de fichier génère le message suivant :

```
AWSBDW066E Planman a demandé à jobman d'exécuter une tâche qui n'est pas prise en charge sur
           l'agent ciblé. Le fichier d'options de méthode suivant a été utilisé :
           fichier_options_méthode. L'identificateur de travail et le PID du moniteur sont
           les suivants : travail, #Jpid_moniteur
```

Lorsqu'une méthode envoie un message non reconnu à **jobman**, le message suivant est généré :

```
AWSBDW064E Un travail contrôlé par jobman a renvoyé le
           message non reconnu suivant : message_incorrect. L'identificateur de travail,
           le PID du moniteur et le fichier de méthode sont les suivants : nom_travail, #Jpid_monite
           utilisant le fichier de méthode.
```

Chapitre 18. Gestion des dépendances interréseaux

Les ***dépendances interréseau*** de Tivoli Workload Scheduler permettent aux travaux et aux flots de travaux d'un réseau local d'utiliser les travaux et flots de travaux d'un réseau distant sous la forme de dépendances de *prédécesseur/successeur*. Le présent chapitre explique comment personnaliser votre environnement afin de pouvoir définir les dépendances interréseau et la gestion des dépendances interréseaux.

Il comprend les sections suivantes :

Remarque : Selon vos besoins et exigences, vous pouvez choisir entre des dépendances interréseau et des dépendances croisées pour établir une dépendance entre un travail s'exécutant sur le moteur local et un travail s'exécutant sur un moteur Tivoli Workload Scheduler distant. Voir «Définition de dépendances», à la page 23 pour une description des différences entre ces deux types de dépendances.

Présentation sur les dépendances interréseaux

Avant d'indiquer une dépendance interréseau, vous devez créer une définition de poste de travail pour l'***agent réseau***. Un agent réseau est un poste de travail Tivoli Workload Scheduler qui gère les dépendances de prédécesseur/successeur entre son réseau local et un réseau Tivoli Workload Scheduler distant.

Le réseau Tivoli Workload Scheduler local peut contenir plusieurs agents réseau, chacun représentant un réseau Tivoli Workload Scheduler distant spécifique où les travaux et les flots de travaux concernant les dépendances interréseaux définies en local sont spécifiées. Les dépendances interréseau sont affectées aux travaux et aux flots de travaux de la même façon que les dépendances de prédécesseur/successeur locales, si ce n'est que le nom de l'agent réseau est inclus afin d'identifier le travail ou le flot de travaux suivi.

Un flot de travaux spécial appelé ***EXTERNAL*** est automatiquement créé par Tivoli Workload Scheduler pour chaque agent réseau dans le réseau local. Il contient des travaux en guise de marques de réservation pour représenter chaque dépendance interréseau.

Un travail `EXTERNAL` est créé pour chaque dépendance interréseau appartenant aux flots de travaux dont le démarrage est planifié à des jours différents et à des dates de planification différentes. Ainsi, un travail `EXTERNAL` diffère d'un autre par :

- Le nom du fichier script qui identifie le travail ou le flot de travaux distant dont le travail ou le flot de travaux local dépend.
- La date prévue pour le démarrage du flot de travaux local contenant la dépendance interréseau. Si la dépendance est définie dans un travail du flot de travaux, la date prévue pour le démarrage du flot de travaux est prise en compte.

La vérification de la dépendance interréseau ne démarre pas tant que le flot de travaux ne correspond pas à sa dépendance horaire ou qu'il n'est pas libéré.

Prenons le cas de deux travaux appartenant à différents flots de travaux et désignant la même dépendance interréseau. Si l'un de leurs flots de travaux est libéré et si le travail démarre, la dépendance interréseau est vérifiée et libérée, le cas échéant. Dans ce cas, lorsque le deuxième travail commence à vérifier sa dépendance interréseau, il trouve que la dépendance est déjà résolue, mais pas obligatoirement le jour prévu. Si vous souhaitez éviter cette situation, vous devez réexécuter le travail représentant la dépendance interréseau une fois qu'elle a été résolue pour la première fois.

Tivoli Workload Scheduler vérifie le statut des travaux et flots de travaux concernés dans le réseau distant, puis mappe leur statut dans les travaux représentant les dépendances interréseaux dans le flot de travaux EXTERNAL. Le statut de ces travaux et flots de travaux est vérifié sur un intervalle de temps fixe jusqu'à ce que le travail ou le flot de travaux distant soit à l'état SUCC, CANCL ou ERROR.

Affichage d'une dépendance interréseau

La présente section décrit un exemple de scénario sur les dépendances interréseau et la liaison du travail représentant la dépendance interréseau au flot de travaux où elle est définie. Supposons que :

- Vous avez défini un flot de travaux nommé ELISCHED qui s'exécute sur le poste de travail TWS206 contenant un travail nommé ELI avec une dépendance interréseau à partir d'un flot de travaux TWS207#FINAL.MAKEPLAN qui s'exécute sur un réseau Tivoli Workload Scheduler différent.
- XA_MAST est l'agent réseau défini dans le réseau local pour gérer les dépendances interréseau à partir des travaux et des flots de travaux définis dans le réseau distant.

Utilisez la commande **conman sj** pour afficher l'ensemble des dépendances interréseaux :

```
                                    (Est) (Est)
CPU  Schedule SchedTime Job State Pr Start Elapse RetCode Deps

TWS206#ELISCHE 0600 03/31 **** HOLD 10 (03/31)
                          ELI  HOLD 10                      XA-MAST::"TWS207#MYJS.JOB1"
```

où (03/31) représente la restriction de temps **at** définie dans TWS206#ELISCHE. En commençant à partir de (03/31), le programme vérifie le statut de TWS207#MYJS.JOB1 dans le réseau distant pour savoir si la dépendance interréseau XA-MAST::"TWS207#MYJS.JOB1" est satisfaite.

Si vous exécutez la commande :

```
%sj XA-MAST#EXTERNAL;info
```

Le programme affiche la liste des travaux représentant les dépendances interréseaux définies dans les flots ou les flots de travaux s'exécutant dans le réseau local à partir des travaux et des flots de travaux définis dans le réseau distant accessible via l'agent réseau XA-MAST :

```
CPU   Schedule SchedTime Job   JobFile        Opt Job
 Prompt
XA-MAST #EXTERNAL
                           E8802332 TWS207#MYJS.JOB1
```

Vous pouvez afficher les détails sur le travail ou le flot de travaux dépendant de TWS207#MYJS.JOB1 dans la dépendance interréseau représentée par le travail E8802332 dans le flot de travaux EXTERNAL en exécutant la commande suivante :

```
%sj @#EXTERNAL.E8802332;deps
```

La sortie montre le lien entre le travail dépendant et la dépendance interréseau :

```
                                        (Est) (Est)
 CPU  Schedule SchedTime Job State Pr Start Elapse RetCode Deps

XA-MAST#EXTERNAL.E8802332 Les dépendances sont les suivantes :


 TWS206#ELISCHE 0600 03/31 **** HOLD 10 (03/31)
                           ELI  HOLD 10                      XA-MAST::"TWS207#MYJS.JOB1"
```

La vérification de la dépendance interréseau ne démarre pas tant que le flot de travaux TWS206#ELISCHE ne correspond pas à sa dépendance horaire (03/31) ou qu'il n'est pas libéré.

Si un autre travail est défini dans un autre flot de travaux dans le réseau local avec une dépendance sur TWS2007#MYJS.JOB1 et si le démarrage du flot de travaux local est prévu le même jour, 31/03/06, la dépendance de cet autre travail sur TWS2007#MYJS.JOB1 sera répertoriée dans le travail E8802332 dans le flot de travaux XA-MAST#EXTERNAL.

Configuration d'un agent réseau

Les postes de travail d'agent réseau sont définis comme des agents étendus et nécessitent un poste de travail physique hôte et une méthode d'accès. La méthode d'accès des agents réseau est nommée **netmth**.

Le processus **batchman** sur le gestionnaire de domaine maître demande **netmth** sur l'agent réseau à des intervalles de temps fixes pour obtenir le statut du travail ou du flot de travaux remplacé distant. Vous pouvez personnaliser l'intervalle de temps entre deux vérifications consécutives en définissant l'option globale *bm check status* dans le fichier localopts sur le gestionnaire de domaine maître. Tivoli Workload Scheduler continue à effectuer les vérifications jusqu'à ce que le travail ou le flot de travaux distant soit à l'état SUCC, CANCL ou ERROR.

Vous devez créer un fichier d'options nommé **netmth.opts** sur le poste de travail où l'agent de réseau s'exécute. L'utilisateur sous lequel la méthode d'accès s'exécute est défini dans ce fichier, ainsi que le délai d'attente d'une réponse de la méthode d'accès avant sa fermeture. Ce fichier d'options doit disposer du même chemin que la méthode d'accès :

racine_TWS/methods/netmth.opts

Le contenu du fichier netmth.opts possède la structure suivante :

```
GSuser=nom_connexion
GStimeout=secondes
```

où :

nom_connexion
: Est le nom de connexion utilisé pour exécuter la méthode. Si l'hôte de l'agent réseau est un ordinateur Windows, cet utilisateur doit être défini dans Tivoli Workload Scheduler.

secondes

Est le nombre de secondes pendant lequel Tivoli Workload Scheduler attend une réponse avant d'arrêter la méthode d'accès. Le paramètre par défaut est 300 secondes. La prochaine fois que **batchman** doit vérifier le statut du travail ou du flot de travaux remplacé distant, la méthode d'accès démarre automatiquement.

Les modifications apportées à ce fichier ne prennent effet qu'une fois que vous avez arrêté puis redémarré Tivoli Workload Scheduler.

Exemple de définition d'un agent réseau

L'exemple suivant montre comment définir un poste de travail d'agent réseau pour un réseau distant, `Réseau A`, qui permet au réseau local, `Réseau B`, d'utiliser les travaux et flots de travaux du réseau distant sous la forme de dépendances interréseaux.

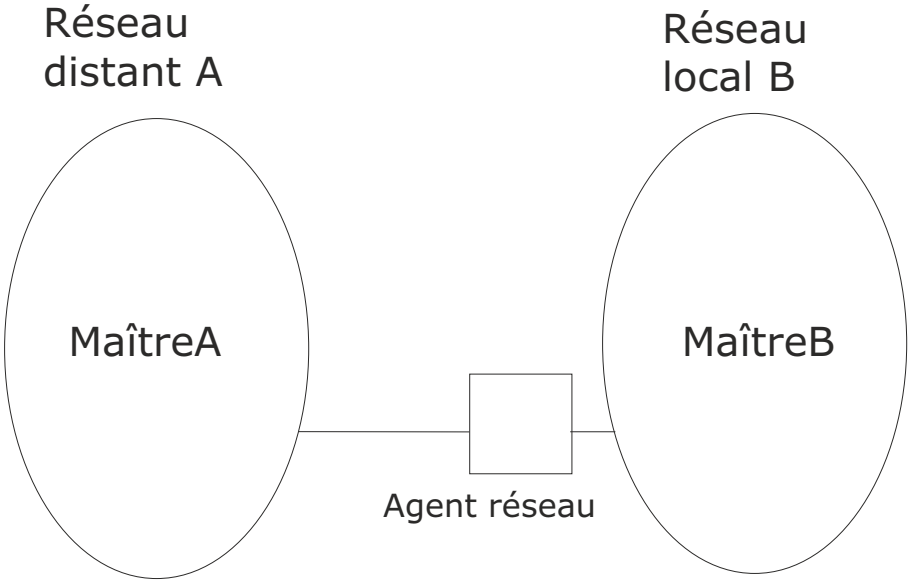


Figure 30. Réseaux local et distant

Supposons que :

- `MasterA` est le gestionnaire de domaine maître du réseau distant, `Réseau A`, et que :
  - **MasterA_tws** est l'utilisateur *utilisateur_TWS* défini sur `MasterA`.
  - Le numéro de port TCP de `MasterA` est 12345.
  - Le noeud sur lequel `MasterA` est défini est `MasterA.rome.tivoli.com`.
- `MasterB` est le gestionnaire de domaine maître du réseau local, `Réseau B`, et que :
  - **MasterB_tws** est l'utilisateur *utilisateur_TWS* défini sur `MasterB`.
  - Le noeud sur lequel `MasterB` est défini est `MasterB.rome.tivoli.com`.

Un poste de travail d'agent réseau nommé `Agent réseau`, défini sur `MasterB` pour gérer les dépendances interréseaux sur les travaux ou flots de travaux définis sur `Réseau A` peut être le suivant :

```
CPUNAME NETAGT
  DESCRIPTION "AGENT RESEAU"
  OS OTHER
  NODE MASTERA.ROME.TIVOLI.COM
```

```
  TCPADDR 12345
  FOR maestro
    HOST MASTERB
    ACCESS netmth
END
```

Important : Saisissez le nom de l'accès réseau `netmth` en minuscule sur les systèmes d'exploitation sensibles à la casse.

Ce fichier d'options, **`netmth.opts`** défini sur `MasterB` peut être :

```
GSuser=maîtreB_tws
GStimeout=600
```

Remarque : L'agent réseau peut être défini sur le gestionnaire de domaine maître ou sur un agent tolérant aux pannes.

Définition d'une dépendance interréseau

La syntaxe utilisée pour définir une dépendance interréseau dans une définition de flot de travaux se présente comme suit :

follows *nom_agent_réseau*::*poste_de_travail_distant#nom_flot_travaux*(*heure* [*date*]).*nom_travail*

où (*heure* [*date*]) sont spécifiques au fuseau horaire utilisé sur le poste de travail du réseau distant auquel l'agent réseau est connecté ; dans notre exemple, le fuseau horaire de `MasterA`. Si (*heure* [*date*]) n'est pas spécifié dans cette syntaxe ou si plus d'un flot de travaux contient les mêmes valeurs (*heure*[*date*]), le premier flot de travaux trouvé est pris en compte.

Supposons que :

- `calendrierA` est un flot de travaux défini dans la base de données `MasterA`.
- `travailA` est un travail défini dans la base de données `MasterA`.
- `calendrierB` est un flot de travaux défini dans la base de données `MasterB`.
- `travailB` est un travail défini dans la base de données `MasterB`.

Vous pouvez définir des dépendances interréseaux à l'aide des instructions **follows** suivantes :

Pour définir une dépendance interréseau de `calendrierB` à partir de l'instance du flot de travaux `calendrierA(1100)`

Utilisez l'instruction suivante :

```
schedule calendrierB
      on everyday
      follows AgtRéseau::MasterA#calendrierA(1100)
      :
end
```

Pour définir une dépendance interréseau de `travailB` à partir de `travailA` contenu dans l'instance de flot de travaux `calendrierA(1100)`

Utilisez l'instruction suivante :

```
schedule calendrierB
      on everyday
      :
      travailB
           follows AgtRéseau::MasterA#planA(1100).jobA
end
```

Vous pouvez également définir des dépendances interréseaux d'un travail sur un flot de travaux ou vice versa.

Gestion des dépendances interréseau dans le plan

Les dépendances interréseau sont gérées dans le plan à partir de la ligne de commande **conman** par gestion du flot de travaux EXTERNAL. Dans le flot de travaux EXTERNAL, les dépendances sont répertoriées en tant que travaux, indépendamment du fait qu'elles soient définies pour des travaux ou des flots de travaux. Il existe un flot de travaux EXTERNE pour chaque agent réseau du plan.

Dans le flot de travaux EXTERNAL, les noms de travaux uniques représentant les dépendances interréseau sont générés comme suit :

Ennnmmss

où :

nnn est un nombre aléatoire.

mm représente les minutes en cours.

ss représente les secondes en cours.

Le nom réel du travail ou du flot de travaux est enregistré dans la spécification des fichiers script de l'enregistrement de travail.

Remarque : Les travaux et les flots de travaux distants sont définis et exécutés sur leur réseau local de façon standard. Qu'ils soient utilisés comme dépendances interréseau n'a aucun effet sur leur comportement local.

Etats des travaux définis dans le flot de travaux EXTERNAL

Le statut des travaux définis dans le flot de travaux EXTERNAL est déterminé par la méthode d'accès et répertorié dans la zone Statut de libération du flot de travaux EXTERNAL. Le statut indiqué se rapporte à la dernière heure où le réseau distant a été vérifié. Les travaux peuvent donner l'impression de sauter des états lorsque les états changent entre deux vérifications.

Tous les états des travaux ou flots de travaux, sauf FENCE, sont répertoriés. Outre ces états, il existe trois états spécifiques aux travaux EXTERNAL :

CANCL
Le travail ou flot de travaux correspondant dans le réseau distant a été annulé.

ERROR
Une erreur est survenue lors de la vérification du statut distant.

EXTRN
Il s'agit de l'état initial. Si le travail est introuvable dans le réseau distant, il reste à l'état EXTRN.

Remarque : Vous pouvez annuler dans le réseau local les instances de travaux ou de flots de travaux dépendants de la même instance d'un travail ou d'un flot de travaux défini dans le réseau distant. Dans ce cas, veillez aussi à annuler manuellement le travail représentant cette dépendance interréseau dans le flot de travaux EXTERNAL pour éviter les reports continus du flot de travaux EXTERNAL.La même remarque s'applique lorsque le flot de travaux local dépendant du travail ou du flot de travaux défini dans le réseau distant n'est pas reporté vers le nouveau plan.

Gestion des travaux définis dans le flot de travaux EXTERNAL

Vous pouvez effectuer les actions ci-dessous sur des travaux dans un flot de travaux EXTERNAL :

Annuler
Annule le travail EXTERNAL en libérant la dépendance interréseau de tous les travaux ou flots de travaux locaux. La vérification du statut de la dépendance prend fin.

Réexécuter
Demande à **conman** de redémarrer la vérification du statut du travail EXTERNE. Après toute exécution de la commande **rerun**, le travail passe immédiatement à l'état EXTRN.

La réexécution est intéressante pour les travaux EXTERNAL ayant l'état ERROR. Par exemple, si un travail EXTERNAL ne peut pas être lancé parce que la méthode d'accès réseau n'accorde pas l'autorisation d'exécution, le travail acquiert l'état ERROR et sont statut cesse d'être vérifié. Lorsque vous aurez corrigé les autorisations, la méthode pourra démarrer, mais **conman** ne démarrera la vérification du statut du travail EXTERNE que lorsque vous aurez effectué une réexécution (**rerun**).

Confirm SUCC / ABEND
Définit le statut du travail EXTERNAL sur SUCC ou ABEND en libérant la dépendance de tous les travaux et flots de travaux locaux dépendants. La vérification du statut de la dépendance prend fin.

Remarque : Aucune de ces commandes n'a d'effet sur le travail ou sur le flot de travaux distant dans le réseau distant. Elles permettent simplement de manipuler les dépendances du réseau local.

Exemples de scénarios de gestion des dépendances interréseaux

La présente section fournit des exemples de scénarios décrivant la gestion des dépendances interréseau dans la production à l'aide des commandes de la ligne de commande **conman**.

Supposons que vous avez déjà défini ce qui suit :

- Un poste de travail local appelé local1
- Un flot de travaux défini pour le poste de travail local local1 appelé plan1
- Un travail défini dans local1#plan1 appelé travail1
- Un agent réseau appelé agtréseau défini dans le réseau local pour gérer la dépendance interréseau à partir des travaux et des flots de travaux définis dans le réseau distant.
- Un poste de travail dans le réseau distant appelé distant1
- Un flot de travaux défini pour le poste de travail distant distant1 appelé pland
- Un travail situé dans distant1#pland appelé travaild

Vous pouvez effectuer plusieurs opérations à partir de la ligne de commande **conman** dans le réseau local. La liste suivante contient plusieurs exemples :

Ajout d'une dépendance interréseau d'un travail distant à un travail local.
Par exemple, pour ajouter le travail distant travaild en tant que dépendance interréseau pour travail1, exécutez la commande suivante :

```
adj local1#plan1.travail1;follows=agtréseau::distant1#pland.travaild
```

Ajout d'une dépendance interréseau d'un flot de travaux distant à un flot de travaux local.

Par exemple, pour ajouter le flot de travaux distant `pland` en tant que dépendance interréseau pour le flot de travaux `plan1`, exécutez la commande suivante :

```
ads local1#plan1;follows=agtréseau::distant1#pland
```

Annulation des dépendances interréseau gérées par un agent réseau.

Par exemple, pour annuler tous les travaux `EXTERNAL` d'un agent réseau `agtréseau`, exécutez l'une des deux commandes suivantes :

```
cj netagt#EXTERNAL.@
```

```
cj agtréseau::@
```

Confirmation de la terminaison correcte de la dépendance interréseau.

Par exemple, pour confirmer que le travail distant `distant1#pland.travaild` s'est correctement terminé et libérer ainsi la dépendance interréseau correspondante, exécutez la commande suivante :

```
confirm agtréseau::distant1#pland.travaild;succ
```

Suppression d'une dépendance interréseau d'un travail vers un autre travail.

Par exemple, pour supprimer la dépendance interréseau du travail distant `distant1#pland.travaild` vers le travail local `local1#plan1.travail1`, exécutez la commande suivante :

```
ddj local1#plan1.travail1;follows=agtréseau::distant1#pland.travaild
```

Suppression d'une dépendance interréseau d'un travail vers un flot de travaux.

Par exemple, pour supprimer la dépendance interréseau du travail distant `distant1#pland.travaild` vers le flot de travaux local `local1#plan1`, exécutez la commande suivante :

```
dds local1#plan1;follows=agtréseau::distant1#pland.travaild
```

Libération d'un travail local à partir d'une dépendance interréseau d'un travail distant.

Par exemple, pour libérer un travail à partir d'une dépendance interréseau d'un travail distant, exécutez la commande suivante :

```
rj
local1#plan1.travail1;follows=agtréseau::distant1#pland.travaild
```

Libération d'un flot de travaux local à partir d'une dépendance interréseau d'un travail distant.

Par exemple, pour libérer un flot de travaux à partir d'une dépendance interréseau d'un travail distant, exécutez la commande suivante :

```
rs
local1#plan1;follows=agtréseau::distant1#pland.travaild
```

Réexécution d'un travail dans le flot de travaux `EXTERNAL` pour redémarrer la vérification d'une dépendance.

Par exemple, pour réexécuter un travail appartenant au flot de travaux `EXTERNAL` afin de redémarrer la vérification de la dépendance interréseau à partir du travail distant `distant1#pland.travaild`, exécutez l'une des deux commandes suivantes :

```
rr agtréseau#EXTERNAL.travaild
```

```
rr agtréseau::distant1#pland.travaild
```

Affichage des dépendances interréseaux à partir des travaux ou flots de travaux définis dans un réseau distant.

Par exemple, pour afficher toutes les dépendances interréseau définies d'un agent réseau avec leurs noms d'origine et leurs noms de travaux générés, exécutez la commande suivante :

```
sj netagt#EXTERNAL.@;info
```

Soumission d'un travail avec une dépendance interréseau à partir d'un flot de travaux défini dans un réseau distant

Par exemple, pour soumettre une commande `rm` dans le flot de travaux `JOBS` avec une dépendance interréseau sur un flot de travaux distant, exécutez la commande suivante :

```
sbd "rm apfile";follows=agtréseau::distant1#pland
```

Dépendances interréseaux dans un environnement mixte

Le tableau 106 montre la configuration prise en charge pour les dépendances interréseaux définies dans un environnement mixte de version 8.3. Les termes clés du tableau sont les suivants :

Net_A Agent de réseau défini sur le réseau local.

Wks_B

Poste de travail sur le réseau distant auquel l'agent de réseau `Net_A` est connecté. `Wks_B` représente le poste de travail qui identifie et contrôle l'état du travail ou du flot de travaux distant spécifié dans la dépendance interréseau.

Sym_A

Fichier `Symphony` traité sur le réseau local.

Sym_B

Fichier `Symphony` traité sur le réseau distant.

niveau antérieur

Version 8.1, 8.2 ou 8.2.1

Tableau 106. Dépendances interréseaux dans un environnement mixte

	Niveau ant. Net_A Niveau ant. Sym_A	**Net_A 8.3 Niveau ant. Sym_A**	**Niveau ant. Net_A Sym_A 8.3**	**Net_A 8.3 Sym_A 8.3**
Niveau ant. Wks_B Niveau ant. Sym_B	Ce n'est pas un environnement mixte de la version 8.3.	`Net_A` envoie les informations à `Wks_B` comme si les deux versions étaient identiques à `Wks_B`.	`Net_A` envoie les informations à `Wks_B` au format 8.1, 8.2 ou 8.2.1. L'utilisation du mot clé *schedtime* dans la définition du travail n'est pas prise en charge.	`Net_A` envoie les informations à `Wks_B` comme si les deux versions étaient identiques à `Wks_B`. Si le mot clé *schedtime* est défini dans la définition du travail, il est automatiquement supprimé par `Net_A`.
Wks_B 8.3 Niveau ant. Sym_B	`Wks_B` fonctionne comme si sa version était identique à celle de `Net_A`.	`Net_A` envoie les informations à `Wks_B`. Si le mot clé *schedtime* est défini dans la définition du travail, il est automatiquement supprimé par `Wks_B`.	`Net_A` envoie les informations à `Wks_B`. Si le mot clé *schedtime* est défini dans la définition du travail, il est automatiquement supprimé par `Wks_B`.	`Net_A` envoie les informations à `Wks_B`. Si le mot clé *schedtime* est défini dans la définition du travail, il est automatiquement supprimé par `Wks_B`.

Tableau 106. Dépendances interréseaux dans un environnement mixte (suite)

	Niveau ant. Net_A Niveau ant. Sym_A	**Net_A 8.3 Niveau ant. Sym_A**	**Niveau ant. Net_A Sym_A 8.3**	**Net_A 8.3 Sym_A 8.3**
Niveau ant. Wks_B Sym_B 8.3	Non pris en charge.	Non pris en charge.	Non pris en charge.	Non pris en charge.
Wks_B 8.3 Sym_B 8.3	Non pris en charge.	Non pris en charge.	`Net_A` envoie les informations à `Wks_B`. Si le mot clé *schedtime* est défini, il est analysé par `Wks_B`.	C'est un environnement de la version 8.3.

Chapitre 19. Définition et gestion des dépendances croisées

Les ***dépendances croisées*** Tivoli Workload Scheduler vous aident à intégrer et automatiser le traitement des travaux lorsque :

- La charge de travail est répartie sur différents environnements de planification, car certaines des activités s'exécutent sur différents sites ou impliquent différentes unités d'organisation, ou requièrent des compétences différentes pour s'exécuter.
- Même si la plus grande partie de la charge de travail par lots est gérée localement, aucun de ces environnements n'est complètement isolé des autres car ils interagissent fréquemment pour échanger ou synchroniser des données et activités.

Plus spécifiquement, la fonction de dépendance croisée est essentielle lorsque vous avez besoin de synchroniser facilement des tâches entre différents environnements de planification afin de pouvoir :

- Définir dans un environnement de planification des dépendances sur des tâches par lots gérées par un autre environnement de planification.
- Surveiller l'état des travaux prédécesseurs distants comme s'ils s'exécutaient dans votre environnement local.

De plus, vous pouvez contrôler le statut de ces dépendances en naviguant à partir d'une interface utilisateur unique sur les différents environnements de planification.

Ce chapitre décrit comment définir et utiliser les dépendances croisées.

Il contient les sections suivantes :

Remarque : Selon vos besoins et exigences, vous pouvez choisir entre des dépendances interréseau et des dépendances croisées pour établir une dépendance entre un travail s'exécutant sur le moteur local et un travail s'exécutant sur un moteur Tivoli Workload Scheduler distant. Voir «Définition de dépendances», à la page 23 pour une description des différences entre ces deux types de dépendances.

Présentation des dépendances croisées

Une dépendance croisée est, d'un point de vue logique, une dépendance qu'un travail local possède sur une instance de travail qui est planifiée pour s'exécuter sur le moteur distant.

Utilisez les dépendances croisées pour intégrer la charge de travail s'exécutant sur différents moteurs, qui peuvent être les moteurs Tivoli Workload Scheduler for z/OS (contrôleur) ou les moteurs Tivoli Workload Scheduler (gestionnaire de domaine maître et gestionnaire de domaine maître de sauvegarde).

Les objets et termes suivants sont utilisés pour décrire et implémenter des dépendances croisées :

Poste de travail de moteur distant
Nouveau type de poste de travail qui représente localement un moteur Tivoli Workload Scheduler distant, distribué ou z/OS. Ce type de poste de travail utilise une connexion basée sur le protocole HTTP ou HTTPS pour permettre à l'environnement local de communiquer avec l'environnement distant.

Travail distant
Travail planifié pour s'exécuter sur un moteur Tivoli Workload Scheduler distant.

Travail reflet
Travail défini localement, sur un poste de travail de moteur distant, qui est utilisé pour mapper un travail distant. La définition de travail reflet contient toutes les informations nécessaires pour associer correctement l'instance de travail distante dans le plan de moteur distant.

Liaison
Processus associant un travail reflet avec une instance de travail distant dans le plan du moteur Tivoli Workload Scheduler distant.

D'un point de vue logique dans l'environnement local :

- Le poste de travail du moteur distant permet de mapper le moteur Tivoli Workload Scheduler distant.
- Le travail reflet défini sur ce poste de travail de moteur distant, est utilisé pour mapper une instance de travail distant planifiée pour s'exécuter sur ce moteur Tivoli Workload Scheduler.

Définissez une dépendance croisée lorsque vous souhaitez qu'un *travail local* (exécution sur votre moteur local) soit dépendant d'un *travail distant* (exécution sur un moteur distant).

Pour ce faire, procédez comme suit :

1. Créez un travail *reflet* qui s'exécute sur votre moteur local.
2. Définissez une dépendance normale qui rend votre travail *local* dépendant du travail *reflet*.

Lorsque vous créez le reflet, tenez compte des éléments suivants :

- Il doit être défini sur un poste de travail de type moteur distant qui pointe vers le moteur distant (qui est le moteur sur lequel le travail distant est planifié pour s'exécuter).
- Vous devez vous assurer qu'il pointe vers le travail distant avec lequel vous créez la dépendance croisée.

La figure 31, à la page 683 présente le flux logique implémentant des dépendances croisées :

1. Lors du processus de liaison, le travail reflet est associé à l'instance du travail distant.

2. Une fois que la liaison est établie, l'état du travail reflet est mis à jour en fonction de la transition d'état du travail distant.
3. Lorsque l'état du travail reflet devient SUCC, la dépendance normale du travail local est libérée, ainsi que la dépendance croisée de ce travail local sur le travail distant.

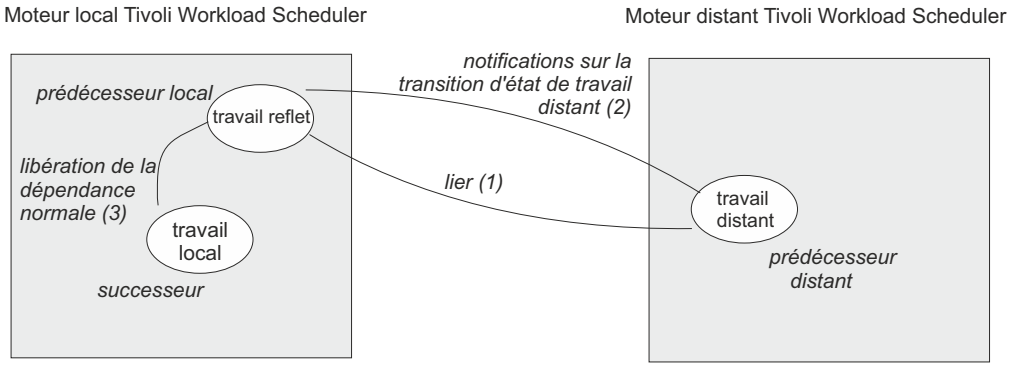


Figure 31. Logique de dépendance croisée

Flux de processus dans l'environnement de planification distribué

Selon que le moteur local émet ou reçoit une demande de liaison, le flux de processus et les composants impliqués changent. Dans les deux cas, le poste de travail courtier de l'environnement local doit être prêt et fonctionner pour permettre la gestion des demandes de liaison.

Flux de processus lorsque le moteur local envoie une demande de liaison sur un moteur distant

Lorsque vous définissez un travail reflet, vous spécifiez les informations nécessaires pour établir une liaison avec un travail dans le plan du moteur distant.

Lorsque l'heure planifiée du travail reflet est atteinte, si le travail reflet ne possède pas de dépendances, il est sélectionné par la commande **batchman** locale pour soumission et son statut est défini par INTRO.

La demande de liaison est envoyée au moteur distant. L'état du travail reflet est défini par WAIT.

Dès que le processus de liaison est achevé, le moteur distant renvoie au moteur local un notification avec le résultat de liaison.

Le tableau 107, à la page 684 présente l'état du travail reflet selon que :

- L'instance à lier existe ou non dans le plan du moteur distant.
- L'état du travail distant est lié.

Tableau 107. Transition d'état de travail reflet

Etat du travail reflet dans le plan de production :	**Sur le moteur distant :**
BOUND	**z/OS** L'instance de flot de travaux distant pour la liaison a été trouvée dans le plan à long terme ou dans le plan courant. **Réparti** L'instance de flot de travaux distant pour la liaison a été trouvée dans le plan de préproduction.
ERROR	**z/OS** L'une des situations suivantes s'est produite : • L'instance du flot de travaux distant pour la liaison n'existe ni dans le plan à long terme, ni dans le plan en cours. • L'instance du flot de travaux distant pour la liaison a été trouvée dans le plan à long terme mais, lorsqu'elle est incluse dans le plan en cours, elle ne contient pas l'instance du travail demandé. **Réparti** L'une des situations suivantes s'est produite : • L'instance du flot de travaux distant à lier n'existe pas dans le plan de préproduction. • L'instance du flot de travaux distant pour la liaison a été trouvée dans le plan de préproduction, mais lorsqu'elle est incluse dans le plan de production, elle ne contient pas l'instance du travail demandé. • L'utilisateur de liaison distant n'est pas autorisé à accéder à l'instance de travail demandée dans le plan de production.
EXEC	L'état du travail distant est EXEC.
SUCC	L'état du travail distant est SUCC.
FAIL	L'état du travail distant est FAIL.
ABEND	L'état du travail distant est ABEND.
SUCC	L'état du travail distant est CANCELED.

Remarque : L'état du travail reflet est FAIL lorsque sa soumission a également échoué.
Pour plus de détails sur la transition d'état du travail reflet, voir «Modification de l'état du travail reflet après que la liaison est établie», à la page 688 et «Modification de l'état du travail reflet après que la liaison est établie», à la page 693.

Flux de processus lorsque le moteur distant reçoit une demande de liaison du moteur local

Lorsque le moteur distant reçoit une demande de liaison du moteur local, les informations contenues dans la demande sont utilisées pour exécuter la liaison dans le plan de préproduction distant.

La demande de liaison contient également une liste ordonnée des URL que le moteur distant utilise pour envoyer des notifications au moteur local. Si le moteur local est distribué, la liste est constituée des URL spécifiées dans la propriété JDURL du fichier nommé *rép_base_TDWB*/config/JobDispatcherConfig.properties.

Remarque : Par défaut, Tivoli Workload Scheduler utilise TWSUser pour exécuter la liaison dans le plan de production. Si vous souhaitez limiter et contrôler les travaux qui peuvent être liés, vous pouvez indiquer un autre utilisateur à l'aide de l'option globale **bindUser**. L'utilisateur spécifié n'a pas besoin d'être défini comme utilisateur sur le système d'exploitation, ni même de posséder un mot de passe, mais il doit exister en tant qu'entrée dans le fichier de sécurité avec les autorisations suivantes :

- Autorisation DISPLAY sur les objets *job* et *schedule* qui peuvent être liés
- Autorisation LIST sur les objets *job* qui peuvent être liés. Cette autorisation est requise uniquement si l'option globale enListSecChk est définie sur yes.

Si les autorisations requises ne sont pas spécifiées, une notification comportant une erreur est renvoyée au moteur qui a demandé la liaison. Le moteur distant renvoie au moteur local :

Une notification avec l'état BOUND
Si le plan de préproduction contient au moins une instance du flot de travaux spécifié dans la demande de liaison et que la définition de ce flot de travaux contient le travail à lier.

Une notification avec l'état de l'instance du travail lié
Si l'instance du travail à lier est trouvée dans le plan de production et dès que son état change.

Une notification avec une erreur
Si l'instance du travail à lier est introuvable ou si l'utilisateur de liaison n'a pas d'autorisation.

Le processus distant **batchman** écrit une entrée dans la file d'attente PlanBox.msg dès que l'état d'un travail distant est modifié.

Toutes les 30 secondes, le système analyse la file d'attente PlanBox.msg et y recherche de nouvelles entrées documentant un changement d'état des travaux distants liés. Dès qu'un changement d'état est trouvé, une notification contenant l'état du travail distant lié est renvoyée au moteur qui a demandé la liaison.

Remarque : Pour modifier le délai d'interrogation, indiquez une valeur en secondes pour **com.ibm.tws.planner.monitor.checkPlanboxInterval** dans le fichier *RACINE_WAS*/profiles/*profile_name*/properties/TWSConfig.properties puis redémarrez WebSphere Application Server.

Définition d'une dépendance croisée

Effectuez ces étapes pour définir une dépendance croisée entre un travail qui s'exécute dans votre environnement et un autre travail s'exécutant sur un moteur Tivoli Workload Scheduler différent :

1. Création d'un poste de travail de moteur distant

   Vous pouvez créer un poste de travail de moteur distant pour un moteur distant spécifique lorsque vous avez besoin de définir des dépendances à des instances de travail qui s'exécutent sur ce moteur distant. Sur un poste de travail de moteur distant, vous ne pouvez exécuter que des travaux reflet.

   Pour une utilisation optimale, Si le moteur distant Tivoli Workload Scheduler est distribué, vous pouvez définir un pool dynamique contenant une liste ordonnée des postes de travail de moteur distant pointant vers le maître distant et vers ses maîtres de sauvegarde afin de garantir que les fonctions de reprise et de modification de gestionnaires sont appliquées. Pour de plus amples informations sur le pool de postes de travail, voir «Poste de travail», à la page 11.

   Remarque : Il est recommandé que :

   - Tous les environnements distribués impliqués aient activé leur fonction de fuseau horaire. Pour de plus amples informations, voir «Activation de la gestion des fuseaux horaires», à la page 653.
   - Vous spécifiez une propriété TIMEZONE sur les postes de travail de moteur distant où le fuseau horaire est défini sur le système d'exploitation des gestionnaires de domaine maîtres ou les gestionnaires de domaines maîtres de sauvegarde vers lesquels ils pointent.

   Pour de plus amples informations sur les paramètres spécifiques à utiliser lors de la définition d'un poste de travail de moteur distant, voir «Définition de poste de travail», à la page 149.
2. Définition d'un travail reflet s'exécutant sur le poste de travail de moteur distant

   Créez un travail reflet pointant vers une instance de travail spécifique définie sur une moteur distant lorsque vous souhaitez suivre l'état de ce travail distant dans votre environnement local et définir des dépendances croisées sur ce travail distant.

   Sur les environnements Tivoli Workload Scheduler distribués, vous pour utiliser des alias pour les nom de travaux et de flots de travaux. Si vous définissez un travail reflet distribué, assurez-vous que :

   - Le nom du flot de travaux distant spécifié contient le nom du flot de travaux tel qu'il est défini dans la base de données.
   - Le nom de travail distant spécifié contient l'alias du travail distant à lier, s'il est défini.

   Si vous ne suivez pas ces consignes, la liaison échoue et l'état du travail reflet devient ERROR.

   Dans la définition du travail reflet, définissez COMPLETE_IF_BIND_FAILS dans la zone *rccondsucc* pour spécifier si l'état du travail reflet doit être forcé à être SUCC ou ERROR en cas d'échec de la liaison dans le plan du moteur distant.

   Pour de plus amples informations sur les paramètres spécifiques à utiliser lors de la définition d'un travail reflet, voir «Définition de travail», à la page 169.

   Selon que le moteur distant est z/OS ou distribué, vous pouvez utiliser différents critères de correspondance :

Si le moteur distant est distribué

Vous pouvez choisir l'un des critères de correspondance suivants :

Tableau 108. Critères de correspondance pour les travaux reflet distribués

Sur le Dynamic Workload Console	Mot clé correspondant utilisé dans composer
Valeur précédente la plus proche	précédent
Dans un intervalle relatif	relative from=*temps_avant_heure_planifiée* to=*temps_après_heure_planifiée*
Dans un intervalle absolu	absolute from=*début_intervalle* to=*fin_intervalle*
Même date de planification	sameDay

Pour de plus amples informations sur ces critères de correspondance, voir «Gestion des dépendances externes de prédécesseur/successeur des travaux et des flots de travaux», à la page 65.

Si le moteur distant est z/OS

La valeur précédente la plus proche est le seul critère de correspondance pris en charge par Tivoli Workload Scheduler for z/OS.

L'heure planifiée de l'instance de flot de travaux contenant le travail reflet est utilisée pour le critère de correspondance dans le plan du moteur distant.

La transition d'état du travail reflet est mappée vers la transition d'état de l'instance de travail distant.

3. Ajoutez une dépendance sur le travail reflet

   Vous ajoutez une dépendance croisée pour un travail local sur le travail distant en définissant une dépendance du travail local sur le travail reflet qui :

   - Pointe vers l'instance de travail distant.
   - Est définie sur un poste de travail local qui pointe vers le moteur distant sur lequel le travail distant est défini.

   La dépendance croisée sur l'instance du travail distant est libérée lorsque la dépendance locale sur le travail reflet est libérée.

Surveillance d'une résolution de dépendance croisée dans le plan de production

Les travaux reflet sont ajoutés au plan comme suit :

- Lors de l'exécution si l'une des situations suivantes se produit :
  - Une définition de travail reflet est soumise à l'aide de la commande **sbj**.
  - Un flot de travaux contenant une définition de travail reflet est soumis à l'aide de la commande **sbs**.

    Remarque : Lorsque vous soumettez un travail reflet, spécifiez un flot de travaux de destination avec une heure planifiée connue pour contrôler de façon plus efficace l'instance de travail distante qui sera liée.
- Lorsque le plan de préproduction est étendu ou créé et que son cadre temporel inclut l'heure planifiée du travail reflet.

Lorsqu'une instance de travail reflet est ajoutée au plan, vous pouvez commencer à surveiller son état.

Modification de l'état du travail reflet après que la liaison est établie

La figure 32 résume comment l'état d'un travail reflet est modifié après que la liaison est établie.

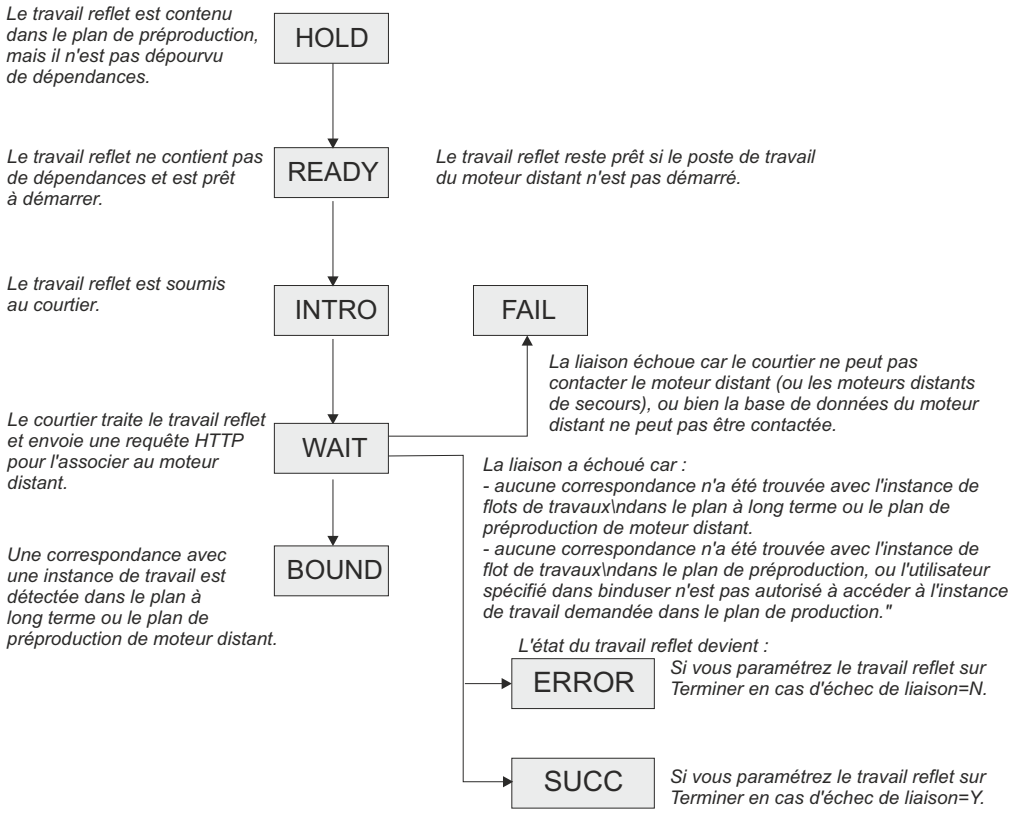


Figure 32. Transition de l'état d'un travail reflet jusqu'à ce que la liaison soit établie

Comme pour tout autre travail, l'état initial du travail reflet est HOLD et devient READY lorsque le travail est libéré des dépendances et prêt à démarrer.

Le planificateur envoie ensuite une demande HTTP sur le moteur distant contenant à la fois les informations permettant d'identifier le travail reflet dans le plan de production local et les informations permettant d'identifier de manière unique l'instance du travail distant à lier dans le plan du moteur distant, y compris les critères de correspondance. Le planificateur doit également être notifié de l'état de l'instance de travail distant liée.

Le planificateur tente de contacter le moteur distant, à intervalles réguliers, jusqu'à l'expiration d'un délai spécifique. Si, d'ici là le moteur distant n'a pas pu être atteint, l'état du travail reflet est défini sur FAIL. Pour modifier le délai et l'intervalle, spécifiez une valeur en secondes pour MaxWaitingTime et StatusCheckInterval dans le fichier *REP_BASE_TDWB*/config/ResourceAdvisorConfig.properties, puis redémarrez le courtier.

Si le plan de préproduction n'existe pas sur le moteur distant lorsque la demande de liaison est reçue, l'état du travail reflet distribué reste WAIT jusqu'à ce que la génération du plan de préproduction soit terminée et que la demande de liaison soit traitée. Cela peut se produire, par exemple, lorsque le plan de préproduction est recréé entièrement sur le moteur distant.

Pour plus d'informations sur la raison de l'échec (FAIL) de l'état du travail reflet, voir «Affichage des raisons de l'état d'échec du travail reflet (FAIL)», à la page 695.

Lorsque le moteur distant reçoit la demande HTTP, il tente d'identifier l'instance de flot de travaux à utiliser pour la liaison dans son plan ; il s'agit du plan de préproduction si le moteur distant est distribué, ou du plan à long terme si le moteur distant est z/OS. La définition du flot de travaux doit contenir la définition du travail distant à lier.

Pour plus d'informations sur l'établissement de la correspondance dans un plan de moteur distant distribué, voir «Liaison d'un travail reflet distribué».

Pour plus d'informations sur l'établissement de la correspondance dans un plan de moteur distant z/OS, voir «Liaison d'un travail reflet z/OS», à la page 690.

Liaison d'un travail reflet distribué

Si le moteur distant est un gestionnaire de domaine maître ou un gestionnaire de domaine maître se sauvegarde Tivoli Workload Scheduler, la recherche de l'instance de travail distant à lier est effectuée dans le plan de préproduction. Les instances de travail distant distribué, appartenant aux flots de travaux JOBS ou USERJOBS, ne sont pas impliquées dans le processus de liaison. Toutefois, les travaux distants déplacés dans USERJOBS après la liaison continuent d'envoyer des notifications de changement de statut.

L'intervalle correspondant, excepté pour le critère de correspondance précédent le plus proche qui ne nécessite pas de calcul d'intervalle, est calculé sur le moteur distant à l'aide des paramètres spécifiés dans la définition du travail reflet distribué.

Par exemple, lorsque le critère de correspondance **sameDay** est spécifié, le jour auquel il est fait référence est le jour spécifié sur le moteur distant en termes de [*startOfDay*, *startOfDay*+23:59].

Lors de l'utilisation de critères de correspondance basés sur intervalle, la demande HTTP envoyée sur le moteur distant contient les informations suivantes pour permettre au moteur distant de calculer l'intervalle de correspondance :

Pour les critères de correspondance d'intervalle absolus :

Les valeurs *hhmm*, *HHMM* et, facultativement, *d* et *D*, spécifiés dans la clause :

```
<dshadow:matching>
<dshadow:absolute from="hhmm [+/-d day[s]]" to="HHMM [+/-D day[s]]"/>
</dshadow:matching>
```

Les valeurs de limite sont *hhmm* -6 days et *HHMM* +6 days.

Le fuseau horaire utilisé pour les critères de correspondance est celui du travail reflet.

Pour les critères de correspondance relatifs :

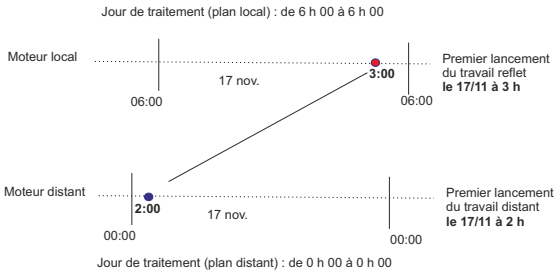
L'heure planifiée du travail reflet et les valeurs [*hh*]*hmm* et [*HH*]*HMM* spécifiées dans la clause :

```
<dshadow:matching>
<dshadow:relative from="+/-[hh]hmm" to="+/-[HH]HMM"/>
</dshadow:matching>
```

Les valeurs limites sont +/-167:59 heures.

Par exemple, pour créer un travail reflet qui correspond à une instance de travail distant dont le début le plus tôt est le 17 novembre à 2:00 AM, vous pouvez spécifier l'un des critères de correspondance suivants :

- **Même date planifiée**
- **Dans un intervalle absolu** en spécifiant comme décalage : **1 jour avant la première date de début.**



L'instance de travail distant pour laquelle une correspondance doit être trouvée est identifiée sur le moteur distant en fonction des règles mentionnées pour les dépendances de prédécesseur/successeur externes. Pour plus de détails sur les critères de résolution des dépendances de prédécesseur/successeur externes, voir «Gestion des dépendances externes de prédécesseur/successeur des travaux et des flots de travaux», à la page 65.

Pour plus d'informations sur la définition des travaux reflet, voir «Définition de travail», à la page 169.

Liaison d'un travail reflet z/OS

Si le moteur distant est un contrôleur IBM Tivoli Workload Scheduler for z/OS, la recherche de l'instance distante à lier est effectuée comme suit :

- L'instance est d'abord recherchée dans le plan à long terme dans la partie de l'intervalle de liaison qui suit l'heure de fin du plan en cours et précède l'heure planifiée du travail reflet.
- Si aucune instance n'est trouvée, l'instance est recherchée dans le plan en cours, dans la partie de l'intervalle de liaison qui précède la fin du plan en cours.

Remarque : Si le contrôleur distant reçoit une demande de liaison avec l'URI de notification du client qui n'est pas définie parmi les destinations HTTP, la demande de liaison est supprimée et le message EQQHT62W est consigné dans le MLOG.

Les sections suivantes décrivent les scénarios qui peuvent se produire lors de la liaison d'un travail reflet z/OS ayant les éléments suivants :

- Heure planifiée : 18:00
- Informations de travail distant :
  - ID d'application : **JS2**
  - Numéro d'opération : **OP2**

Dans les figures :

- La boîte blanche indique l'intervalle de temps couvert par le plan à long terme.
- La boîte gris clair indique l'intervalle de temps couvert par le plan en cours.
- La boîte gris foncé indique l'intervalle dans le plan de moteur distant durant lequel l'instance de travail à lier doit être recherchée.
- L'occurrence **JS2** mise en évidence en gras est l'instance sélectionnée pour la liaison.

Scénario 1 : l'intervalle du plan en cours contient l'heure planifiée du travail reflet et des occurrences JS2 existent.

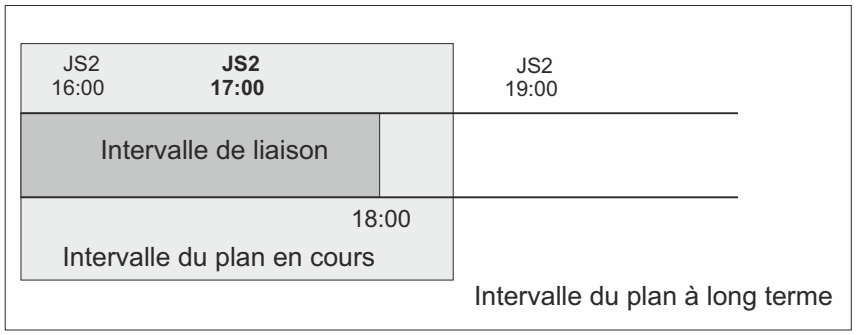


Figure 33. Instance à lier si l'heure planifiée du travail reflet est incluse dans l'intervalle du plan en cours

La figure 33 indique, mise en évidence en gras, l'instance **JS2** qui précède de la façon la plus proche l'heure planifiée du travail reflet. Cette instance est sélectionnée pour la liaison car l'heure planifiée est contenue dans le plan en cours. Le travail reflet et l'instance de travail distant sont associés. Si, par la suite, une nouvelle instance de **JS2** qui précède de la façon la plus proche l'heure planifiée du travail reflet est soumise ad hoc dans le plan du moteur distant, la correspondance avec l'instance **JS2** sélectionnée pour la liaison n'est *pas* modifiée.

A ce stade, l'une des situations suivantes peut se produire :

L'instance JS2 sélectionnée contient OP2.
La liaison avec **OP2** appartenant à **JS2** est établie et une notification contenant :

- Les informations de travail distant identifiant l'instance **OP2** dans le plan de moteur distant
- L'état en cours de cette instance **OP2**

est renvoyée, l'instance du travail distant est mise à jour avec les informations de travail distant et son état est mis à jour en conséquence.

L'instance JS2 sélectionnée ne contient plus OP2 parce qu'il a été supprimé et qu'un plan quotidien l'a supprimé du plan en cours ou parce qu'il n'a jamais été contenu dans JS2.
La liaison échoue. Une notification informant que la liaison a échoué est renvoyée et l'état du travail reflet est mis à jour selon ce que vous avez défini dans la zone **Terminer en cas d'échec de liaison**.

L'instance JS2 sélectionnée contient l'OP2 qui a été supprimée mais pas encore retirée du plan en cours.
La liaison est établie et une notification informant de l'état d'exécution réussi est renvoyée. L'instance du travail reflet est marquée comme SUCC. Ses successeurs peuvent démarrer.

Scénario 2 : l'intervalle du plan en cours contient l'heure planifiée du travail reflet, l'instance JS2 qui précède l'heure planifiée du travail reflet de la façon la plus proche existe dans le plan à long terme mais a été annulée du plan en cours.

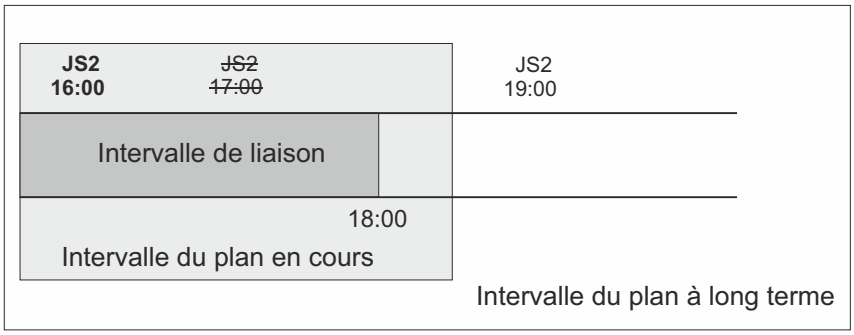


Figure 34. Instance à lier si l'instance qui précède de la façon la plus proche l'heure planifiée du travail reflet existe dans le plan à long terme mais a été annulée du plan en cours.

La figure 34 indique, mise en évidence en gras, l'instance **JS2** qui est sélectionnée pour la liaison, parce que l'occurrence qui correspondait le mieux a été supprimée.

La liaison avec **OP2** appartenant à **JS2** est établie et une notification contenant :

- Les informations de travail distant identifiant l'instance **OP2** dans le plan de moteur distant
- L'état en cours de cette instance **OP2**

est renvoyée, l'instance du travail distant est mise à jour avec les informations de travail distant et son état est mis à jour en conséquence.

Scénario 3 : l'intervalle du plan en cours contient l'heure planifiée du travail reflet mais aucune occurrence JS2 n'existe.

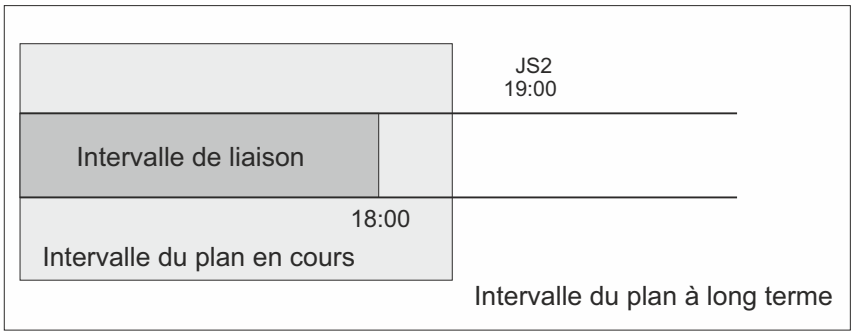


Figure 35. L'heure planifiée du travail reflet est incluse dans le plan en cours mais aucune instance à lier n'existe

La figure 35 indique que l'instance **JS2** qui précède l'heure planifiée de travail reflet de la façon la plus proche n'existe pas.

La liaison échoue. Une notification informant que la liaison a échoué est renvoyée et l'état du travail reflet est mis à jour selon ce que vous avez défini dans la zone **Terminer en cas d'échec de liaison**.

Scénario 4 : L'intervalle du plan à long terme contient l'heure planifiée du travail en cours et le plan en cours ne comprend pas encore l'instance JS2 immédiatement précédente.

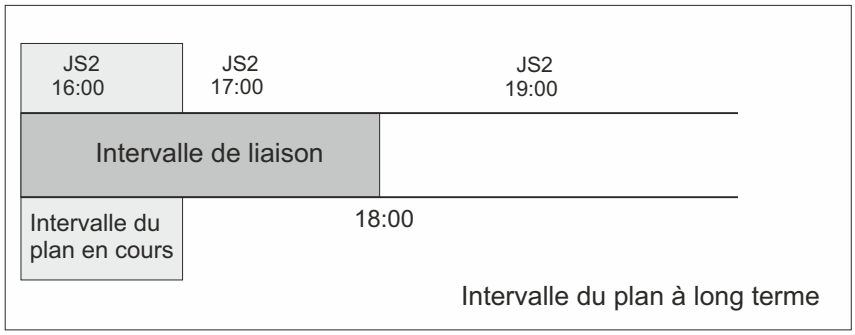


Figure 36. L'instance à lier existe mais n'est pas encore incluse dans le plan en cours

La figure 36 indique l'instance **JS2** qui peut être associé avec le travail reflet, même lorsque le travail **JOB2** n'est pas encore présent dans le plan en cours.

Une notification informant que la liaison est établie est renvoyée et l'état du travail reflet est défini par BOUND.

Scénario 5 : l'intervalle du plan à long terme ne contient toujours pas l'heure planifiée du travail reflet.

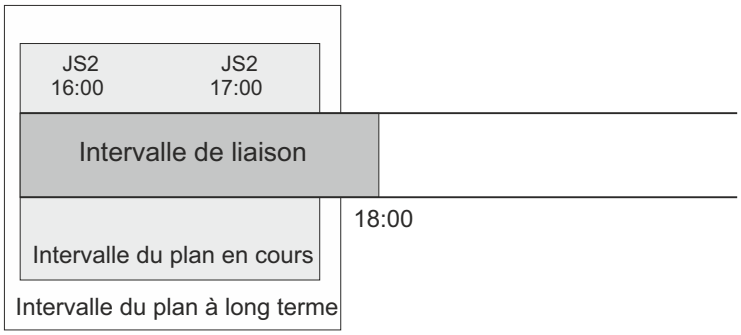


Figure 37. L'intervalle du plan à long terme ne contient toujours pas l'heure planifiée du travail reflet.

La figure 37 indique qu'aucune instance **JS2** ne peut être associée au travail reflet car, jusqu'au moment où le plan à long terme inclut l'heure planifiée du travail reflet, des instances **JS2** précédentes plus proches peuvent encore être ajoutées.

Dans ce cas, la demande de liaison est mise en attente jusqu'à ce que le plan à long terme soit étendu pour inclure l'heure planifiée du travail reflet. Avant cela, l'état du travail reflet reste WAIT.

Modification de l'état du travail reflet après que la liaison est établie

Lorsqu'une liaison est établie, le moteur distant renvoie une notification HTTP contenant l'état de la liaison et, si la liaison réussit, les informations identifiant l'instance de travail distant liée. Ces informations apparaissent dans les détails de l'instance du travail reflet.

Selon le type de moteur distant, les informations suivantes sur l'instance de travail distant apparaissent dans les propriétés de travail reflet :

Le type de moteur distant est distribué

- Nom du flot de travaux
- Heure planifiée
- Poste de travail du flot de travaux
- Nom du travail

Le type de moteur distant est z/OS

- ID d'application
- Heure planifiée
- Numéro d'opération
- Poste de travail
- Nom de travail, s'il a été défini sur le moteur distant.

Lorsque l'instance de travail reflet est mappée vers une instance de travail distant existante, des notifications sur les modifications d'état de travail sont envoyées de façon asynchrone à partir du moteur distant. Elles servent à mapper la transition d'état de travail distant vers la transition d'état de travail reflet. Le mécanisme de stockage et d'acheminement garantit la livraison des messages et la récupération en cas de panne. La figure 38 présente comment l'état d'un travail reflet distribué change, depuis le moment où une liaison est établie jusqu'à ce que l'état du travail reflet devienne SUCC ou ERROR. Seuls les états SUCC et ERROR sont considérés comme l'état final du travail reflet.

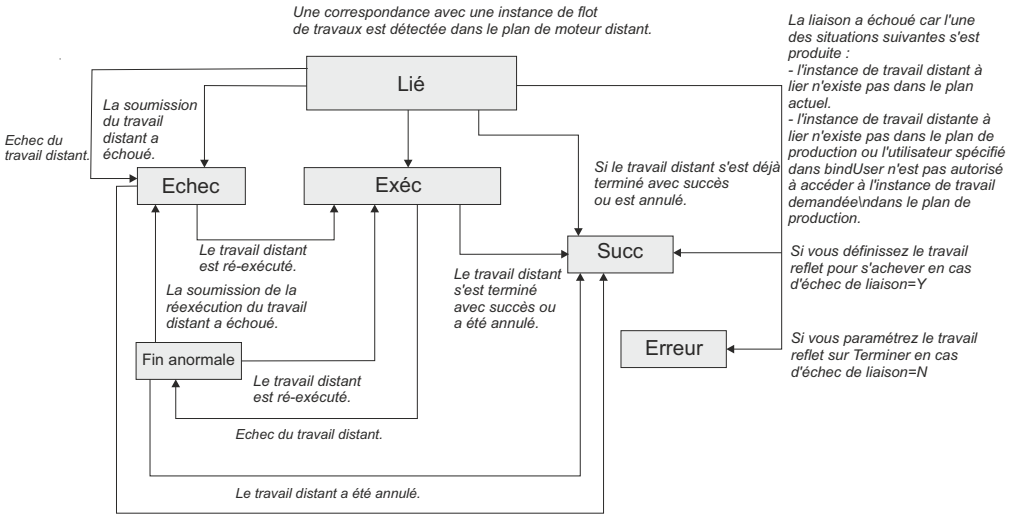


Figure 38. Chaîne de transition d'état de travail reflet après que la liaison a été établie

Si l'instance de travail distant est déjà terminée lorsque la correspondance est trouvée, l'état du travail reflet devient immédiatement SUCC.

Pour plus d'informations sur la raison de l'échec (FAIL) de l'état du travail reflet, voir «Affichage des raisons de l'état d'échec du travail reflet (FAIL)», à la page 695.

Lorsque l'état du travail reflet satisfait à la règle de dépendance, la dépendance du travail local sur le travail reflet est résolue et la dépendance croisée du travail local sur le travail distant est également résolue.

Affichage des raisons de l'état d'échec du travail reflet (FAIL)

L'état du travail reflet peut être FAIL dans une des situations suivantes :

- La soumission du travail reflet a échoué.
- La soumission du travail distant a échoué.

Pour déterminer la raison pour laquelle le statut du travail reflet est FAIL, consultez le journal du travail reflet soit en exécutant la commande **showjobs** avec l'option `;stdlist`, soit en cliquant sur **Journal du travail...** pour l'instance du travail reflet dans la vue **Surveillance des travaux** de Dynamic Workload Console.

Etat d'un travail reflet lors de la récupération ou la réexécution d'un travail distant

Une fois que la liaison est établie, il peut arriver que le travail distant lié soit réexécuté. Dans ce cas, l'état du travail reflet reflète l'état du travail réexécuté. L'état du travail reflet reste EXEC tant que la récupération du travail distant est en cours.

L'état du travail reflet est mis à jour uniquement lorsque le travail distant atteint l'un des états suivants :

ABEND
Lorsque le travail distant ne réussit pas à s'exécuter.

SUCC Lorsque le travail distant réussit.

FAIL Lorsque la soumission du travail distant échoue.

Vous pouvez consulter plus de détails sur le travail distant dans les propriétés du travail reflet. Pour afficher ces détails :

- Exécutez la commande **conman showjobs** avec l'option **props** sur le travail reflet.
- Accédez au panneau des propriétés du travail reflet dans Dynamic Workload Console.

Application du report aux dépendances croisées

Le report fonctionne de la même manière avec les travaux reflet qu'avec les autres types de travail. Les travaux reflet avec le statut **WAIT** et **BOUND** sont traités de la même manière que les travaux avec le statut **EXEC**. Les travaux reflets avec le statut **ERROR** sont traités de la même manière que les travaux avec le statut **FAIL** ou **ABEND**.

L'état d'un travail reflet, lié à un travail distant qui n'est pas reporté, est défini sur ERROR lorsque le plan de production distant est étendu.

Remarque : Une meilleure pratique consiste à utiliser les dépendances croisées avec le report de flots de travaux sur les environnements de planification distribués et distants.

Pour plus d'informations sur l'option globale **carryStates**, voir le *Guide d'administration*.

Gestion des travaux reflet dans le plan de production

Selon l'état du travail reflet, vous pouvez exécuter les commandes suivantes :

Kill Vous pouvez arrêter un travail reflet dont l'état est BOUND, EXEC ou WAIT. L'association établie par la liaison avec le travail distant est annulée automatiquement et l'état du travail reflet est défini sur ABEND avec le code retour 0.

Rerun Vous pouvez réexécuter un travail reflet dont l'état est ABEND, ERROR, SUCC ou FAIL. Lorsque vous réexécutez un travail reflet, une nouvelle demande de liaison est déclenchée.

Chapitre 20. Gestion d'un environnement dynamique IBM i

Gestion des agents IBM i dans un environnement dynamique et planification des travaux avec options avancées sur les agents IBM i.

Définition des agents sur les systèmes IBM i

Pour commencer à planifier des travaux sur les agents IBM i, l'agent doit se trouver dans le réseau Tivoli Workload Scheduler. A la fin du processus d'installation, l'agent est automatiquement enregistré dans la base de données Tivoli Workload Scheduler.

Vous pouvez vérifier l'existence de la définition de poste de travail de l'agent installé sur un système IBM i soit à l'aide de Dynamic Workload Console, soit à l'aide de la ligne de commande **composer**.

Pour plus d'informations sur l'utilisation de la console pour visualiser les définitions de poste de travail, voir le manuel *Dynamic Workload Console - Guide de l'utilisateur, section sur la modification de définitions de poste de travail*.

Pour plus d'informations sur l'utilisation de l'interface de ligne de commande pour visualiser des définitions de poste de travail, voir «Définition de poste de travail», à la page 149.

Pour inclure l'agent IBM i dans le plan, voir *Tivoli Workload Scheduler - Guide de planification et d'installation : Partie 3. Tivoli Workload Scheduler sur IBM i - Configuration d'un agent dynamique*.

Définition des travaux sur les systèmes IBM i

Sur les agents IBM i, vous pouvez définir les types de travail suivants avec options avancées :

Travaux de services Web
Pour définir des travaux de services Web, voir «Définition de travail - Travaux de services Web», à la page 180.

Travaux de transfert de fichiers
Pour définir des travaux de transfert de fichier, voir «Définition de travail - Travaux de transfert de fichier», à la page 183.

Travaux J2EE
Pour définir des travaux J2EE, voir «Définition de travail - Travaux J2EE», à la page 192.

Travaux de base de données
Pour définir des travaux de base de données, voir «Définition de travail - Travaux de base de données», à la page 194.

Travaux Java
Pour définir des travaux Java, voir «Définition de travail - Travaux Java», à la page 198.

Travaux exécutables
Pour définir des travaux exécutables, voir «Définition de travail - Travaux exécutables», à la page 199.

Travaux IBM i
Pour définir des travaux IBM i qui exécutent des commandes natives des systèmes d'exploitation IBM i, voir «Définition de travail - Travaux IBM i», à la page 205.

Travaux de commande distante
Pour définir des travaux de commande distante, voir «Définition de travail - Travaux de type commande distante», à la page 200.

Travaux TPM (Provisioning)
Pour définir des travaux TPM (Provisioning), voir «Définition de travail - Travaux de type TPM (Provisioning)», à la page 188.

Vous pouvez définir des travaux avec options avancées sur un agent IBM i soit à l'aide de Dynamic Workload Console, soit à l'aide de la ligne de commande **composer**.

Pour plus d'informations sur la procédure de définition des travaux IBM i, voir les sections sur les étapes préalables de création des types de travaux avec options avancées et sur la création de définitions de travail dans *Tivoli Dynamic Workload Console - Guide d'utilisation*.

Pour plus d'informations sur l'utilisation de l'interface de ligne de commande pour créer des définitions de travail, voir «Définition de travail», à la page 169.

Gestion des agents sur les systèmes IBM i

Vous pouvez utiliser Tivoli Workload Scheduler sur les agents IBM i pour démarrer et arrêter uniquement les processus d'agent. Pour plus d'informations sur le démarrage et l'arrêt des agents IBM i, voir «Démarrage et arrêt des agents sur les systèmes IBM i».

Pour gérer l'agent IBM i, utilisez les utilitaires décrits dans «Utilisation des commandes d'utilitaire pour les agents sur les systèmes IBM i».

Démarrage et arrêt des agents sur les systèmes IBM i

Vous pouvez utiliser Tivoli Workload Scheduler sur les agents IBM i pour démarrer et arrêter uniquement les processus d'agent.

Démarrage des agents sur les systèmes IBM i :
Utilisez l'utilitaire **StartUpLwa**.

Pour plus d'informations sur l'utilitaire permettant de démarrer les agents sur IBM i, voir «StartUpLwa», à la page 577.

Arrêt des agents sur les systèmes IBM i :
Utilisez l'utilitaire **ShutDownLwa**.

Pour plus d'informations sur l'utilitaire permettant d'arrêter les agents sur IBM i, voir «ShutDownLwa», à la page 576.

Utilisation des commandes d'utilitaire pour les agents sur les systèmes IBM i

Pour les agents sur les systèmes IBM i, vous pouvez utiliser les utilitaires suivants :

param Pour utiliser l'utilitaire param, voir «param», à la page 592.

twstrace
Pour utiliser l'utilitaire twstrace, voir «twstrace», à la page 606.

resource
Pour utiliser l'utilitaire resource, voir «Ressource», à la page 595.

cpuinfo
Pour utiliser l'utilitaire cpuinfo, voir «cpuinfo», à la page 544.

version
Pour utiliser l'utilitaire version, voir «version», à la page 578.

Planification des travaux sur les systèmes IBM i

Lors de la planification d'un travail sur les systèmes IBM i, le travail lance une commande native qui peut être soit une commande système, soit une commande utilisateur. La commande native contient une commande système SBMJOB, qui lance un travail par lots. La commande native démarre un ou plusieurs travaux par lots. Les travaux par lots peuvent être surveillés seulement s'ils sont démarrés par la commande native. L'agent de surveillance IBM i peut surveiller un maximum de 130 travaux par lots.

L'agent journal des travaux et la variable d'envionnement TWSASPOOLS

Par défaut, toutes les informations relatives à l'exécution des travaux sont stockées dans l'agent journal des travaux. La plupart de ces informations se composent de fichiers spoules. Pour sélectionner les types de fichier spoule qui vous souhaitez inclure dans le journal des travaux de l'agent, utilisez la variable système **TWSASPOOLS**, qui fonctionne au niveau de l'agent IBM i pour n'importe quel travail à soumettre.

La variable système **TWSASPOOLS** force l'agent IBM i à ignorer tous les fichiers spoule ou à inclure un ou plusieurs de ces fichiers.

Sur l'agent IBM i, créez une nouvelle variable d'environnement au niveau système nommée TWSASPOOLS et définissez-la dans une liste des types de fichier spoule à inclure. La liste doit commencer par le marqueur **SPOOLS:**.

Par exemple, pour forcer l'agent IBM i à ignorer tous les fichiers spoules, créer la variable TWSASPOOLS comme suit.

```
ADDENVVAR ENVVAR(TWSASPOOLS) VALUE(SPOOLS:) LEVEL(*SYS)
```

où la liste après le marqueur SPOOL: est vide. Dans ce cas, tout rapport de journal des travaux de l'agent pour l'agent IBM i est limité au rapport d'activités que le moniteur d'agent produit pour suivre la soumission et la surveillance de l'action, ainsi qu'au journal des travaux IBM i du moniteur d'agent qui est toujours ajouté à la fin du journal des travaux de l'agent.

Pour autoriser l'agent IBM i à inclure seulement les types de fichier spoule QPRINT et QPJOBLOG, c'est-à-dire, tout fichier spoule produit par des instructions **printf** dans tout programme ILE-C et tout journal des travaux produit, créez la variable TWSASPOOLS comme suit :

```
ADDENVVAR ENVVAR(TWSASPOOLS) VALUE('SPOOLS: QPRINT QPJOBLOG') LEVEL(*SYS)
```

Si la variable TWSASPOOLS existe déjà, modifiez-la de la manière suivante :

```
CHGENVVAR ENVVAR(TWSASPOOLS) VALUE('SPOOLS: QPRINT QPJOBLOG') LEVEL(*SYS)
```

Si tout paramètre VALUE parameter est défini dans une chaîne incorrecte, l'agentIBM i ignore l'option de variable environnement TWSASPOOLS. Vous

pouvez créer et modifier la variable d'environnement tandis que l'agent IBM i est actif mais aucune charge de travail ne doit être en cours d'exécution.

Surveillance des travaux enfants sur les agents IBM i

Lorsque vous soumettez une commande sur un agent IBM i, la commande risque de démarrer un ou plusieurs travaux par lots. L'agent IBM i surveille ces travaux par lots, également appelés travaux enfants.

Lors de la recherche et de la surveillance de travaux enfants démarrés, l'agent IBM i utilise un pourcentage élevé de son temps de traitement.

Si vous savez que votre planification des travaux ne démarre aucun travail enfant ou si vous ne vous préoccupez pas de la surveillance des travaux enfants, vous pouvez ordonner à l'agent IBM i de ne pas chercher et surveiller des travaux enfants, et donc de ne pas améliorer les performances de l'agent.

Vous pouvez exclure la surveillance des travaux enfants soit au niveau de l'agent pour toutes les commandes, soit au niveau de la définition de travail pour une commande unique. Si vous souhaitez activer la surveillance des travaux enfants pour certaines commandes soumises uniquement, vous pouvez définir cette option au niveau de la définition de travail pour une commande unique.

Vous pouvez exécuter l'une ou les deux procédures suivantes pour exclure ou inclure la surveillance des travaux enfants :

Exclure les travaux enfants de la surveillance des travaux au niveau de l'agent

Par défaut, les travaux enfants sont surveillés. Vous pouvez exclure des travaux enfants de la surveillance des travaux pour toutes les commandes soumises en créant la variable d'environnement système `TWS_NOCHILDS` à l'aide de la commande de système IBM i suivante :

```
ADDENVVAR ENVVAR(TWS_NOCHILDS)LEVEL(*SYS)
```

Si l'agent IBM i trouve `TWS_NOCHILDS` sur le système IBM i, il ne surveillera pas les travaux enfants pour aucune commande soumise.

Exclure ou inclure les travaux enfants de la surveillance des travaux au niveau de la définition de travail

Vous pouvez exclure ou inclure des travaux enfants de la surveillance des travaux pour un travail spécifique en utilisant `:NOCHILDS` ou `:CHILDS` comme jetons de fin de la chaîne de commande pour la commande en question.

- Si vous ajoutez le jeton de fin `:NOCHILDS` à la fin de la commande native que vous soumettez, l'agent IBM i ignore tous les travaux enfants démarrés par la commande.
- Si vous ajoutez le jeton de fin `:CHILDS` à la fin de la commande que vous soumettez, l'agent IBM i trouve et surveille tous les travaux enfants démarrés par la commande.

Remarque : Le paramètre au niveau de la définition de travail remplace le paramètre au niveau de l'agent.

Lorsqu'elle est soumise, la commande de système **SBMJOB** démarre toujours un travail par lots. N'essayez pas d'exclure la surveillance des travaux, car si l'agent IBM i trouve la commande `SBMJOB` dans la définition de travail, il supprime et ignore le jeton de fin `:CHILDS` ou `:NOCHILDS` dans la définition de travail et ignore également le paramètre de la variable système `TWS_NOCHILDS`.

Exemples

Pour surveiller les travaux enfants démarrés lors de l'exécution du programme PAYROLL, définissez la commande suivante dans la définition de travail :

- Si la variable système TWS_NOCHILDS est définie sur le système IBM i :
  ```
  CALL PGM(MYLIB/PAYROLL):CHILDS
  ```
- Si la variable système TWS_NOCHILDS n'est pas définie sur le système IBM i :
  ```
  CALL PGM(MYLIB/PAYROLL)
  ```

Pour ne pas surveiller les travaux enfants démarrés lors de l'exécution du programme MYSCHEDULE, définissez la commande suivante dans la définition de travail :

- Si la variable système TWS_NOCHILDS n'est pas définie sur le système IBM i :
  ```
  CALL PGM(MYLIB/MYSCHEDULE):NOCHILDS
  ```
- Si la variable système TWS_NOCHILDS est définie sur le système IBM i :
  ```
  CALL PGM(MYLIB/MYSCHEDULE)
  ```

Remarque : La commande SBMJOB démarre toujours un travail enfant. L'agent IBM i surveille le travail enfant même si vous définissez une commande SBMJOB, comme dans la définition de travail suivante :

```
SBMJOB CMD(CALL PGM(MYLIB/USERPGM)):NOCHILDS
```

Informations sur la surveillance des travaux enfants dans les journaux des travaux de l'agent IBM i

Si vous avez inclus la surveillance des travaux enfants sur les agents IBM i, comme décrit dans la section «Surveillance des travaux enfants sur les agents IBM i», à la page 700, vous pouvez visualiser les informations relatives à la surveillance des travaux enfants dans le journal des travaux de l'agent IBM i.

Exemples

L'exemple ci-dessous présente les informations relatives à la surveillance des travaux enfants incluse au niveau du travail pour le travail CHILDLONG_CHILD sur l'agent NC117025 :

```
Travail CHILDLONG_CHILD

Poste de travail (travail) NC117025

Flot de travaux AS400ENVSET

Poste de travail (flot de travaux) NC117025


================================================================
= JOB       : NC117025#AS400ENVSET[(1417 10/30/12),(AS400ENVSET)].CHILDLONG_CHILD
= TASK      : <?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
xmlns:jsdlibmi="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdlibmi" name="ibmi">
  <jsdl:variables>
    <jsdl:stringVariable name="tws.jobstream.id">AS400ENVSET</jsdl:stringVariable>
    <jsdl:stringVariable name="tws.job.workstation">NC117025</jsdl:stringVariable>
    <jsdl:stringVariable name="tws.job.iawstz">201210301417</jsdl:stringVariable>
  </jsdl:variables>
  <jsdl:application name="ibmi">
    <jsdlibmi:ibmi>
      <jsdlibmi:IBMIParameters>
         <jsdlibmi:Task>
          <jsdlibmi:command>CALL PGM(MINERMA/SBM5JOBS) :CHILDS</jsdlibmi:command>
```

```
        </jsdlibmi:Task>
      </jsdlibmi:IBMIParameters>
    </jsdlibmi:ibmi>
  </jsdl:application>
  <jsdl:resources>
    <jsdl:orderedCandidatedWorkstations>
      <jsdl:workstation>805E5EAC1F5911E2B9DB6F8202778C47</jsdl:workstation>
    </jsdl:orderedCandidatedWorkstations>
  </jsdl:resources>
</jsdl:jobDefinition>
= TWSRCMAP  :
= AGENT     : NC117025
= Numéro de travail : 760232858
= Tue Oct 30 14:16:31 CET 2012
===============================================================
Le travail de surveillance soumis par l'agent dynamique est qualifié sous la forme :
   JobName=DYNAMICMON JobUser=CLAUDIO JobNumber=361743
La chaîne de commande de l'utilisateur est la suivante
   <CALL PGM(MINERMA/SBM5JOBS) :CHILDS>
2012/10/30 14:16:28.844 - Le travail de l'agent dynamique a soumis la commande utilisateur
   CALL PGM(MINERMA/SBM5JOBS)
LES 5 TRAVAUX SUIVANTS ONT DEMARRE sous la commande utilisateur soumise
  JobName=CLAUDIO    JobUser=CLAUDIO    JobNumber=361765
  JobName=CLAUDIO    JobUser=CLAUDIO    JobNumber=361756
  JobName=CLAUDIO    JobUser=CLAUDIO    JobNumber=361762
  JobName=CLAUDIO    JobUser=CLAUDIO    JobNumber=361775
  JobName=CLAUDIO    JobUser=CLAUDIO    JobNumber=361774
Message CPF1241 (Réussite) reçu sur MsgQueue CLAUDIO   QUSRSYS
   pour le travail CLAUDIO   CLAUDIO   361765
Message CPF1241 (Réussite) reçu sur MsgQueue CLAUDIO   QUSRSYS
   pour le travail CLAUDIO   CLAUDIO   361756
Message CPF1241 (Réussite) reçu sur MsgQueue CLAUDIO   QUSRSYS
   pour le travail CLAUDIO   CLAUDIO   361762
Message CPF1241 (Réussite) reçu sur MsgQueue CLAUDIO   QUSRSYS
   pour le travail CLAUDIO   CLAUDIO   361775
Message CPF1241 (Réussite) reçu sur MsgQueue CLAUDIO   QUSRSYS
   pour le travail CLAUDIO   CLAUDIO   361774
*** Codes de FIN regroupés par le travail de surveillance ***
 > Code de statut de FIN (Statut) : 0
 > Code retour du PROGRAMME (Prc) : 0
 > Code retour de l'Utilisateur (Urc) : 0
   Code de référence d'unité récupéré via SYSAPI

2012/10/30 14:21:37.890 - Le travail de l'agent dynamique a mis fin à la surveillance
de la commande utilisateur
*** Le code retour pour la commande soumise est 0 ***
*** La commande utilisateur s'est terminée correctement ***
```

L'exemple ci-dessous présente le journal des travaux pour le travail CHILDLING_NC sur l'agent NC117025 lorsque la surveillance des travaux enfants est exclue au niveau du travail :

```
Travail CHILDLING_NC

Poste de travail (travail) NC117025

Flot de travaux AS400ENVSET

Poste de travail (flot de travaux) NC117025


===============================================================
= JOB       : NC117025#AS400ENVSET[(1417 10/30/12),(AS400ENVSET)].CHILDLING_NC
= TASK      : <?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
xmlns:jsdlibmi="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdlibmi" name="ibmi">
  <jsdl:variables>
```

```
    <jsdl:stringVariable name="tws.jobstream.id">AS400ENVSET</jsdl:stringVariable>
    <jsdl:stringVariable name="tws.job.workstation">NC117025</jsdl:stringVariable>
    <jsdl:stringVariable name="tws.job.iawstz">201210301417</jsdl:stringVariable>
  </jsdl:variables>
  <jsdl:application name="ibmi">
    <jsdlibmi:ibmi>
       <jsdlibmi:IBMIParameters>
         <jsdlibmi:Task>
           <jsdlibmi:command>CALL PGM(MINERMA/SBM5JOBS) :NOCHILDS</jsdlibmi:command>
         </jsdlibmi:Task>
       </jsdlibmi:IBMIParameters>
        </jsdlibmi:ibmi>
  </jsdl:application>
  <jsdl:resources>
    <jsdl:orderedCandidatedWorkstations>
      <jsdl:workstation>805E5EAC1F5911E2B9DB6F8202778C47</jsdl:workstation>
    </jsdl:orderedCandidatedWorkstations>
  </jsdl:resources>
</jsdl:jobDefinition>
= TWSRCMAP  :
= AGENT     : NC117025
= Numéro du travail : 760232857
= Tue Oct 30 14:17:01 CET 2012
==============================================================
Le travail de surveillance soumis par l'agent dynamique est qualifié sous la forme :
   JobName=DYNAMICMON JobUser=CLAUDIO JobNumber=361817
La chaîne de commande de l'utilisateur est la suivante
   <CALL PGM(MINERMA/SBM5JOBS) :NOCHILDS>
2012/10/30 14:16:58.330 - Le travail de l'agent dynamique a soumis la commande utilisateur
   CALL PGM(MINERMA/SBM5JOBS)
Par choix de l'utilisateur, AUCUN travail démarré sous la commande soumise ne sera surveillé
*** Codes de FIN regroupés par le travail de surveillance ***
 > Code de statut de FIN (Statut) : 0
 > Code retour du PROGRAMME (Prc) : 0
 > Code retour de l'Utilisateur (Urc) : 0
   Code de référence d'unité récupéré via SYSAPI

2012/10/30 14:17:10.220 - Le travail de l'agent dynamique a mis fin à la surveillance
de la commande utilisateur
*** Le code retour pour la commande soumise est 0 ***
*** La commande utilisateur s'est terminée correctement ***
```

Récupération du code retour agent

Le modèle de programmation IBM i avait pour origine un vieux modèle à orientation objet dans lequel les programmes communiquaient par transmission de messages et non par utilisation de code retour. L'introduction du modèle ILE (Integrated Language Environment) a mené la définition de zones communes d'échange de données comme les codes retour dans le même environnement de travail : les codes retour utilisateur.

Pour de plus amples informations sur les codes retour utilisateur, voir «Contrôle de l'environnement de travail à l'aide du code retour utilisateur», à la page 704.

Lorsque l'agent IBM i vérifie qu'une commande soumise ou qu'un travail est achevé, il assigne un code retour au travail en fonction de l'état du travail achevé. Le code retour est défini en fonction du message d'achèvement de la commande ou du travail. Si la commande ou le travail se termine avec succès, le code retour est défini sur 0. Si la commande ou le travail ne s'est pas achevé avec succès, le code retour est réglé en fonction de la valeur de la gravité du message lié à l'exception qui a causé la terminaison anormale du travail. L'agent IBM i peut également définir le code retour sur la valeur du code retour utilisateur lorsqu'il

est récupéré par la commande soumise. S'il est récupéré, le code retour utilisateur est utilisé comme valeur pour définir le code retour.

La valeur du code retour affectée au travail est incluse dans le journal des travaux de l'agent IBM i pour le travail et renvoyée à l'interface utilisateur du planificateur (panneaux WEB UI ou z/OS ISPF) sous la forme d'un code retour, pour des raisons de compatibilité avec les agents sur d'autres systèmes d'exploitation.

Contrôle de l'environnement de travail à l'aide du code retour utilisateur

Avec l'introduction du modèle ILE IBM i ILE, il est possible de récupérer une valeur retournée par un programme appelé au sein du même travail.

Lorsque l'agent de surveillance vérifie qu'une commande soumise est terminée, il récupère les codes de fin de travail suivants à l'aide d'un système API IBM i :

Code d'état de fin ou <Status> (0 en cas de succès)

Il indique si le système a envoyé une annulation contrôlée pour le travail. Les valeurs admises sont les suivantes :

1 Le sous-système ou le travail est annulé.

0 Le sous-système ou le travail n'est pas annulé.

vide Le travail n'est pas en cours d'exécution.

Code retour de programme ou <Prc> (0000 en cas de succès)

Il indique le code achèvement du dernier programme (par exemple, un utilitaire de création-maintenance de fichiers ou un programme RPG ou COBOL appelé par le travail).

Si le travail n'inclut aucun programme, le code retour de programme est 0.

Code retour utilisateur ou <Urc> (0000 en cas de succès)

Il indique le code retour défini par l'utilisateur et mis en place par les constructions du langage évolué ILE. Par exemple, le code retour d'un programme écrit en langage C.

Il représente le code retour le plus récent défini par une unité d'exécution dans un travail.

Si la commande soumise est un appel à un programme ILE utilisateur renvoyant une valeur à la sortie, cette valeur peut être trouvée dans le code de référence d'unité du travail.

Vous pouvez décider des modalités de contrôle de l'environnement de travail des travaux que vous avez soumis en préparant les commandes pour qu'elles soient soumises comme des CALLs pour vos programmes ILS, où le flux interne est contrôlé et le statut de fin est décidé par le biais de valeurs de sortie correctes. Si un programme utilisateur se termine en erreur suite à un contrôle du débit incorrect, sans renvoyer de valeur, l'agent de surveillance ne définit pas le code retour comme un code de référence d'unité (Urc) mais suit les critères décrits dans «Récupération du code retour agent», à la page 703.

L'exemple suivant montre un programme utilisateur ILE C dans lequel sont lancés deux travaux par lots et où une valeur de 10 est renvoyée au demandeur, indépendamment du statut d'achèvement des travaux par lots.

```
=============================================================================
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void main(int argc, char *argv[])
{
   int  EnvVarRC=0;
   printf("issuing SBMJOB CMD(CALL MYLIB/DIVBY0)...\n");
   system("SBMJOB CMD(CALL MYLIB/DIVBY0)");
   printf("issuing SBMJOB CMD(WRKACTJOB OUTPUT(*PRINT))...\n");
   system("SBMJOB CMD(WRKACTJOB OUTPUT(*PRINT)) LOG(4 0 *SECLVL)");
   exit(10);
   return;
}
=============================================================================
```

Méthode alternative de définition du code retour utilisateur

Dans certains environnements IBM i, le système API récupérant le code retour utilisateur (Urc) du code de l'analyse de l'agent ne récupère pas la bonne valeur pour l'Urc. Il est donc déconseillé d'utiliser les systèmes d'interfaces de programme d'application IBM i pour récupérer le code retour utilisateur. Pour recevoir une valeur renvoyée par un programme appelé, il est préférable de fournir un paramètre destiné à recevoir cette valeur.

Même si l'analyse de l'agent parvient à récupérer le code retour utilisateur avec un système d'interfaces de programme d'application, une méthode alternative de récupération de code retour utilisateur a été implémentée dans le code de l'analyse de l'agent. La méthode de récupération alternative possède la logique suivante. La variable d'environnement du travail `USERRC` est créée et définie sur la valeur *INI* avant de soumettre la commande utilisateur. Lorsque la commande prend fin, l'agent de surveillance récupère le code retour de son utilisateur à l'aide du système APIs, mais elle vérifie également si la variable d'environnement du travail `USERRC` a été mise à jour au niveau du programme de l'utilisateur. Si une valeur autre que *INI* est trouvée, elle est considérée comme étant le code retour de l'utilisateur et la valeur récupérée à l'aide des API système est ignorée car le programme utilisateur a modifié la valeur de la variable d'environnement du travail `USERRC`.

La modification de la variable `USERRC` au niveau du programme utilisateur nécessite de modifier la valeur `USERRC` avant de quitter l'application du code utilisateur. Dans le cas d'ILE C, vous pouvez réaliser cette tâche à l'aide de l'instruction **putenv**, dans laquelle la récupération du code retour est définie.

L'exemple suivant montre comment le code utilisateur retransmet le code retour utilisateur à l'aide de la variable d'environnement `USERRC` du travail réservé de l'agent IBM i. Ce code a été obtenu à partir du code de l'exemple dans «Contrôle de l'environnement de travail à l'aide du code retour utilisateur», à la page 704 en remplaçant l'instruction **exit** par l'instruction **putenv**.

```
=============================================================================
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void main(int argc, char *argv[])
{
   int  EnvVarRC=0;
   printf("issuing SBMJOB CMD(CALL MYLIB/DIVBY0)...\n");
   system("SBMJOB CMD(CALL MYLIB/DIVBY0)");
   printf("issuing SBMJOB CMD(WRKACTJOB OUTPUT(*PRINT))...\n");
   system("SBMJOB CMD(WRKACTJOB OUTPUT(*PRINT)) LOG(4 0 *SECLVL)");
   EnvVarRC = putenv("USERRC=10");
```

```
   return;
}
============================================================================
```

Annexe A. Automatisation de la charge de travail commandée par les événements, définitions des événements et des actions

Cette annexe documente les fournisseurs d'événements et d'actions utilisables pour l'automatisation de la charge de travail commandée par les événements et donne des définitions détaillées des événements et actions.

Fournisseurs d'événements et définitions

Cette section détaille les types d'événement des fournisseurs d'événement suivants :

- `TWSObjectsMonitor`
- `FileMonitor`
- `TWSApplicationMonitor`

Datetime
Contient à la fois la date et l'heure. Vous pouvez spécifier l'une de ces deux valeurs, ou les deux, dans le filtre.

Plusieurs prédicats de filtrage sont autorisés
Vous pouvez définir plusieurs prédicats de filtrage pour cette propriété. L'événement correspond à la condition d'événement si tous les prédicats sont satisfaits.

Valeurs multiples autorisées
Vous pouvez définir plusieurs valeurs pour cette propriété au sein d'un prédicat de filtrage unique. Le filtre est satisfait lorsqu'il existe une correspondance avec l'une des valeurs.

Caractères génériques autorisés
Les caractères génériques pris en charge sont l'astérisque (*) et le point d'interrogation (?).

Evénements TWSObjectsMonitor

Les événements `TWSObjectsMonitor` sont :

- Statut du flux de travaux modifié
- Travail Until
- Travail soumis
- Travail annulé
- Travail relancé
- Travail en retard
- Travail promû
- Niveau de risque du travail modifié
- Le travail a dépassé la durée maximale
- Le travail n'a pas atteint la durée minimale
- Statut du flot de travaux modifié
- Flot de travaux terminé
- Flot de travaux Until
- Flot de travaux soumis
- Flot de travaux annulé
- Flot de travaux en retard
- Statut du poste de travail modifié
- Statut du serveur d'applications modifié

- Lien vers le poste de travail enfant modifié
- Lien vers le poste de travail parent modifié
- Statut de l'invite modifié
- ProductAlert

Ces événements sont générés par `batchman` (ou `mailman` pour les postes de travail) et écrits dans un fichier de boîte aux lettres nommé `monbox.msg`. Les objets de planification sont surveillés comme suit : Les objets de planification sont surveillés comme suit :

- • Les travaux sont contrôlés par le poste de travail sur lequel ils sont exécutés
- • Les flots de travaux sont contrôlés par le gestionnaire de domaine maître
- • Les postes de travail se contrôlent eux-mêmes
- • Les invites locales sont contrôlées par le poste de travail exécutant le travail ou le flot de travaux qui a une dépendance sur l'invite
- • Les invites globales sont contrôlées par le gestionnaire de domaine maître

Cliquez ici pour afficher les zones Dynamic Workload Console de chaque type d'événement.

Remarque : Utilisateurs PDF, les tables de paramètre ci-dessus constituent un fichier html référencé dans le fichier PDF. Il n'est pas sauvegardé localement avec le fichier PDF dans le centre de documentation. Vous devez dans un premier temps le consulter dans le centre de documentation avant de le sauvegarder ou de l'imprimer.

Utilisation des événements WorkstationStatusChanged

L'événement est démarré lorsqu'un poste de travail est démarré ou arrêté. Mais les différences opérationnelles suivantes existent en fonction du type de poste de travail qui est contrôlé :

- Pour un agent tolérant aux pannes, l'événement est envoyé lorsque le poste de travail est démarré ou arrêté.
- Pour un poste de travail Tivoli Dynamic Workload Broker, l'événement est également envoyé lorsqu'il est lié ou non lié (lorsque ces commandes démarrent ou arrêtent également le poste de travail).
- Pour un poste de travail de pool dynamique, l'événement n'est jamais envoyé (même si Tivoli Dynamic Workload Broker est arrêté) car il n'y a pas de contrôle pour ce type de poste de travail.

Exemples

La règle dans l'exemple suivant soumet le flot de travaux `RJS_102739750` sur le poste de travail `NC125102` dès que tous les travaux du flot de travaux `RCF_307577430` du poste de travail `NA022502` ont le statut RUNNING ou SUCCESSFUL.

```
<?xml version="1.0"?>
<eventRuleSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns="http://www.ibm.com/xmlns/prod/tws/1.0/event-management/rules"
      xsi:schemaLocation="http://www.ibm.com/xmlns/prod/tws/1.0/
                          event-management/rules/EventRules.xsd">
   <eventRule name="TWS_PLAN_EVENTS_JOB_STATUS_CHANGED" ruleType="filter" isDraft="no">
      <description>Event: Job Status Changed; Action: Submit job stream</description>
      <timeZone>Europe/Rome</timeZone>
      <validity from="2011-04-24" to="2012-04-24" />
      <activeTime start="00:00:00" end="12:00:00" />
      <eventCondition name="jobStatChgEvt1"
                      eventProvider="TWSObjectsMonitor"
                      eventType="JobStatusChanged">
```

```
      <scope>* # JOBSTREAMVALUE . * [RUNNING, SUCCESSFUL]</scope>
         <filteringPredicate>
          <attributeFilter name="JobStreamWorkstation" operator="eq">
         <value>NA022502</value>
          </attributeFilter>
         <attributeFilter name="JobStreamName" operator="eq">
         <value>RCF_307577430</value>
         </attributeFilter>
         <attributeFilter name="JobName" operator="eq">
         <value>*</value>
         </attributeFilter>
         <attributeFilter name="Priority" operator="ge">
         <value>10</value>
         </attributeFilter>
         <attributeFilter name="Monitored" operator="eq">
         <value>true</value>
         </attributeFilter>
         <attributeFilter name="Status" operator="eq">
         <value>Running</value>
         <value>Successful</value>
         </attributeFilter>
         <attributeFilter name="Login" operator="eq">
         <value>TWS_user</value>
         </attributeFilter>
         </filteringPredicate>
      </eventCondition>
      <action actionProvider="TWSAction" actionType="sbs" responseType="onDetection">
         <description>Launch an existing TWS job stream</description>
         <scope>SBS NC125102#RJS_102739750</scope>
         <parameter name="JobStreamWorkstationName">
            <value>NC125102</value>
         </parameter>
         <parameter name="JobStreamName">
            <value>RJS_102739750</value>
         </parameter>
      </action>
   </eventRule>
</eventRuleSet>
```

La règle dans l'exemple suivant soumet le travail RJR_30411 sur le poste de travail NC122160 dès que le flot de travaux RJS_102739750 du poste de travail NC125102 est soumis.

```
<?xml version="1.0"?>
<eventRuleSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns="http://www.ibm.com/xmlns/prod/tws/1.0/event-management/rules"
      xsi:schemaLocation="http://www.ibm.com/xmlns/prod/tws/1.0/
                          event-management/rules/EventRules.xsd">
   <eventRule name="TWS_PLAN_EVENTS_JOB_STREAM_SUBMITTED" ruleType="filter" isDraft="no">
      <description>Event: Job Stream Submitted; Action: Submit job</description>
      <eventCondition name="jsSubEvt1"
                      eventProvider="TWSObjectsMonitor"
                      eventType="JobStreamSubmit">
      <scope>WORKSTATIONVALUE # JOBSTREAMVALUE</scope>
         <filteringPredicate>
          <attributeFilter name="JobStreamWorkstation" operator="eq">
         <value>NC125102</value>
          </attributeFilter>
         <attributeFilter name="JobStreamName" operator="eq">
         <value>RJS_102739750</value>
         </attributeFilter>
         <attributeFilter name="Priority" operator="range">
         <value>15</value>
         <value>30</value>
         </attributeFilter>
         <attributeFilter name="LatestStart" operator="le">
         <value>2011-04-26</value>
         </attributeFilter>
         </filteringPredicate>
      </eventCondition>
      <action actionProvider="TWSAction" actionType="sbj" responseType="onDetection">
```

```
        <description>Launch an existing TWS job stream</description>
        <scope>SBJ NC122160#RJR_30411 INTO NC122160#JOBS</scope>
        <parameter name="JobUseUniqueAlias">
           <value>true</value>
        </parameter>
        <parameter name="JobDefinitionName">
           <value>RJR_30411</value>
        </parameter>
        <parameter name="JobDefinitionWorkstationName">
           <value>NC122160</value>
        </parameter>
      </action>
   </eventRule>
</eventRuleSet>
```

Evénements FileMonitor

Les événements FileMonitor sont :

- FileCreated
- FileDeleted
- ModificationCompleted
- LogMessageWritten

Lorsque vous interceptez des fichiers à l'aide des événements FileCreated, FileDeleted et LogMessageWritten, la mémoire consommée par les processus ssmagent.bin et ssmagent.exe augmente linéairement en fonction du nombre de fichiers interceptés et du nombre d'événements créés. Par conséquent, gardez à l'esprit que plus vous ferez appel aux caractères génériques dans ces types d'événement, et donc, plus vous intercepterez de fichiers , plus la mémoire consommée par les processus ssmagent.bin et ssmagent.exe sera importante.

Les événements FileMonitor ne sont pas pris en charge sur :

- Les pools, pools dynamiques et postes de travail de moteur distant.
- IBM i.

Cliquez ici pour afficher les zones Dynamic Workload Console de chaque type d'événement.

Remarque : Utilisateurs PDF, les tables de paramètre ci-dessus constituent un fichier html référencé dans le fichier PDF. Il n'est pas sauvegardé localement avec le fichier PDF dans le centre de documentation. Vous devez dans un premier temps le consulter dans le centre de documentation avant de le sauvegarder ou de l'imprimer.

Utiliser la propriété MatchExpression de la règle d'événement LogMessageWritten

Le plug-in événement LogMessageWritten utilise l'expression régulière spécifiée dans la propriété MatchExpression property afin d'effectuer les correspondances de sous-chaîne dans les entrées du fichier journal sous surveillance. La valeur de MatchExpression doit être une expression régulière valide conforme aux règles de syntaxe des expressions régulières de l'agent Netcool/SSM utilisés par le plug-in événement.

Les tables suivantes décrivent la syntaxe des marqueurs d'expressions régulières pris en charge par Netcool/SSM. Remarquez que pour écrire une expression régulière valide pour la propriété MatchExpression, vous devez écrire le caractère d'échappement \ (barre oblique inversée) avant chaque marqueur utilisé dans la

syntaxe d'expression régulière (par exemple, \^ ou \$). Lorsque le marqueur comporte déjà une barre oblique inversée, vous devez en écrire une seconde (par exemple, \\< ou \\b).

Tableau 109. Syntaxe d'expression régulière.

Marqueur	Correspondances
.	Tout caractère.
^	Le début de ligne (une chaîne nulle).
$	La fin d'une ligne ; une nouvelle ligne ou la fin du tampon de recherche.
\<	Le début d'un mot (où un mot est une chaîne de caractères alphanumériques).
\>	La fin d'un mot (la chaîne nulle entre un caractère alphanumérique et un caractère non alphanumérique).
\b	Toute frontière de mot (équivalent à (\<¦\>)).
\d	Un caractère numérique.
\D	Tout caractère non numérique.
\w	Un caractère de mot (alphanumérique ou trait de soulignement).
\W	Tout caractère qui n'est pas un caractère de mot (alphanumérique ou trait de soulignement).
\s	Un caractère blanc.
\S	Tout caractère non blanc.
\c	Caractères spéciaux et échappement. Les caractères suivants sont interprétés selon les conventions du langage C : \0, \a, \f, \n, \r, \t, \v. Pour indiquer un caractère en hexadécimal, employez la syntaxe \xNN syntax. Par exemple, \x41 est le caractère ASCII A.
\	A l'exception des caractères décris au-dessus, tous les caractères peuvent être fermés à l'aide de la barre oblique inversée, qui compose le préfixe d'échappement. Par exemple, pour indiquer un simple crochet gauche, utilisez \[.

Tableau 109. Syntaxe d'expression régulière. (suite)

Marqueur	Correspondances
[]	Tout caractère spécifié dans un jeu. Un jeu de caractères explicite peut être spécifié, comme dans [aeiou], tout comme des intervalles de caractères, comme [0-9A-Fa-f], qui correspondent à n'importe quel caractère numérique hexadécimal. Le trait d'union (-) perd sa signification particulière lorsqu'il est accompagné du caractère d'échappement, comme dans [A\-Z] ou lorsqu'il est le premier ou le dernier caractère d'un jeu, comme par exemple dans [-xyz0-9]. Toutes les règles d'échappement à l'aide de la barre oblique inversée peuvent être utilisée à l'intérieur des []. Par exemple, l'expression [\x41-\x45] équivaut à l'ASCII [A-D]. Pour utiliser un crochet de fermeture dans un jeu, vous pouvez soit le fermer à l'aide de [\]] ou l'utiliser comme premier caractère d'un jeu, comme dans []xyz]. Les classes de caractères POSIX-style aussi autorisées dans un jeu de caractères. La syntaxe pour les classes de caractères est [:class:]. Les classes de caractères pris en charge sont : • [:alnum:] - caractères alphanumériques. • [:alpha:] - caractères alphabétiques. • [:blank:] - caractères de tabulation et espace. • [:cntrl:] - caractères de commande. • [:digit:] - caractères numériques. • [:graph:] - caractères imprimables et visibles. • [:lower:] - caractères alphabétiques minuscules. • [:print:] - caractères imprimables (qui ne sont pas des caractères de commande). • [:punct:] - caractères de ponctuation (qui ne sont ni des lettres, ni des chiffres, ni des caractères de commande, ni des espaces). • [:space:] - caractères espaces (tel que l'espace, la tabulation et le saut de page). • [:upper:] - caractères alphabétiques majuscules. • [:xdigit:] - caractères numériques hexadécimaux. Les crochets sont autorisés à l'intérieur des crochets de jeu. Par exemple,[a-z0-9!] équivaut à [[:lower:][:digit:]!] dans l'environnement local C.
[^]	Inverse le comportement d'un jeu de caractères [] comme décrit au-dessus. Par exemple, [^[:alpha:]] correspond à tout caractère qui n'est pas alphabétique. Le symbole caret ^ n'a cette signification particulière que lorsqu'il occupe la première place d'un jeu de caractères entre crochets.
{n}	Exactement n occurrences de la précédente expression, où 0 <= n <= 255. Par exemple, a{3}à aaa.
{n,m}	Entre n et m occurrences de la précédente expression, où 0 <= n <= m <= 255. Par exemple, un nombre hexadécimal de 32-bit peut être décrit comme 0x[[:xdigit:]]{1,8}.
{n,}	Au moins n occurrences ou plus (jusqu'à une infinité) de la précédente expression.
*	Zéro occurrence ou plus de la précédente expression.
+	Une occurrence ou plus de la précédente expression.
?	Zéro ou une occurrence de la précédente expression.

Tableau 109. Syntaxe d'expression régulière. (suite)

Marqueur	Correspondances
(exp)	Groupement ; toute série d'expression peut être regroupée entre parenthèses afin d'appliquer un opérateur postfix ou barre pour un groupe d'expressions successives. Par exemple : • à ab+ correspondent tous les abbb • à (ab)+ correspondent tous les ababab
¦	Les autres expressions (logique OR). La barre verticale (¦) possède la priorité la plus basse de tous les marqueurs du langage des expressions régulières. Cela signifie que, à ab¦cd, correspondent tous les cd mais pas les abd (dans ce cas, utilisez a(b¦c)d).

Conseil : Lorsque vous définissez des expressions régulières pour la correspondance de caractères multi-octets, entourez chaque caractère multi-octets de parenthèses ().

Le tableau 110 fournit un ensemble d'exemples d'expressions régulières, tout comme des exemples de chaînes, ainsi que les résultats de l'application des expressions régulières à ces chaînes.

Il y a deux cas importants dans la correspondance d'expressions régulières et de chaînes. Une expression régulière peut correspondre à une chaîne entière (cas connu comme un *correspondance de chaîne*) ou seulement à une partie de cette chaîne (cas connu comme un *correspondance de sous-chaîne*). Par exemple, l'expression régulière `\<int\>` va générer une correspondance de sous-chaîne pour la chaîne `int x` mais ne générera pas une correspondance de chaîne. Cette distinction est importante parce que certains sous-agents ne prennent pas en charge la correspondance de sous-chaîne. Lorsqu'ils sont applicables, les résultats listés dans les exemples différencient les correspondances de chaîne et de sous-chaîne.

Tableau 110. Exemples d'expressions régulières.

Cette expression...	Appliquée à cette chaîne...	Résulte en...
.	a	Correspondance de chaîne
	!	Correspondance de chaîne
	abcdef	Correspondance de sous-chaîne pour a
	chaîne vide	Pas de correspondance
M..COUNT	MINCOUNT	Correspondance de chaîne
	MXXCOUNTY	Correspondance de sous-chaîne pour MXXCOUNT
	NONCOUNT	Pas de correspondance
.*	chaîne vide	Correspondance de chaîne
	Animal	Correspondance de chaîne
.+	Toute chaîne non vide	Correspondance de chaîne
	chaîne vide	Pas de correspondance
^	chaîne vide	Correspondance de chaîne
	hello	Correspondance de sous-chaîne de longueur 0 en position 0 (position 0 = premier caractère d'une chaîne)

Tableau 110. Exemples d'expressions régulières. (suite)

Cette expression...	Appliquée à cette chaîne...	Résulte en...
$	chaîne vide	Correspondance de chaîne
	hello	Correspondance de sous-chaîne de longueur 0 en position 5 (position 0 = premier caractère d'une chaîne)
^$	chaîne vide	Correspondance de chaîne
	hello	Pas de correspondance
\bee	tee	Pas de correspondance
	Paid fee	Pas de correspondance
	feel	Pas de correspondance
	eel	Correspondance de sous-chaîne pour ee
.*thing.*	The thing is in here	Correspondance de chaîne
	there is a thing	Correspondance de chaîne
	it isn't here	Pas de correspondance
	thinxxx	Pas de correspondance
a*	chaîne vide	Correspondance de chaîne
	aaaaaaaaa	Correspondance de chaîne
	a	Correspondance de chaîne
	aardvark	Correspondance de sous-chaîne pour aa
	this string	Correspondance de sous-chaîne
((ab)*c)*	chaîne vide	Correspondance de chaîne
	cccccccc	Correspondance de chaîne
	ccccabcccabc	Correspondance de chaîne
a+	chaîne vide	Pas de correspondance
	aaaaaaaaa	Correspondance de chaîne
	a	Correspondance de chaîne
	aardvark	Correspondance de sous-chaîne pour aa
	this string	Pas de correspondance
(ab)+c)*	chaîne vide	Correspondance de chaîne
	abababababcabc	Correspondance de chaîne
(ab){2}	abab	Correspondance de chaîne
	cdababababab	Correspondance de sous-chaîne pour abab
[0-9]{4,}	123	Pas de correspondance
	a1234	Correspondance de sous-chaîne pour 1234
a{0}	chaîne vide	Correspondance de chaîne
	a	Pas de correspondance
	hello	Correspondance de sous-chaîne de longueur 0 en position 0 (position 0 = premier caractère d'une chaîne)
[0-9]{1,8}	this is not a number	Pas de correspondance
	a=4238, b=4392876	Correspondance de sous-chaîne pour 4238

Tableau 110. Exemples d'expressions régulières. (suite)

Cette expression...	Appliquée à cette chaîne...	Résulte en...
([aeiou][^aeiou])+	Hello	Correspondance de sous-chaîne pour el
	!!! Supacalafraglistic	Correspondance de sous-chaîne pour upacalaf
[+-]?1	1	Correspondance de chaîne
	+1	Correspondance de chaîne
	-1	Correspondance de chaîne
	.1	Correspondance de sous-chaîne pour 1
	value+1	Correspondance de sous-chaîne pour +1
a¦b	a	Correspondance de chaîne
	b	Correspondance de chaîne
	c	Pas de correspondance
	Daniel	Correspondance de sous-chaîne pour a
abcd¦efgh	abcd	Correspondance de chaîne
	efgh	Correspondance de chaîne
	abcdfgh	Correspondance de sous-chaîne pour abcd
[0-9A-F]+	BAADF00D	Correspondance de chaîne
	C	Correspondance de chaîne
	baadF00D	Correspondance de sous-chaîne pour F00D
	c	Pas de correspondance
	G	Pas de correspondance
	g	Pas de correspondance
x = \d+	x = 1234	Correspondance de chaîne
	x = 0	Correspondance de chaîne
	x = 1234a	Correspondance de sous-chaîne pour x = 1234
	x = y	Pas de correspondance
	x^=^ où ^ représente un caractère espace	Pas de correspondance
\D\d	a1	Correspondance de chaîne
	a11	Correspondance de sous-chaîne pour a1
	-9	Correspondance de chaîne
	a	Pas de correspondance
	8	Pas de correspondance
	aa	Pas de correspondance
	4t	Pas de correspondance

Tableau 110. Exemples d'expressions régulières. (suite)

Cette expression...	Appliquée à cette chaîne...	Résulte en...
\s+	Hello_w0rld	Pas de correspondance
	Hello^^^world où ^ représente un caractère espace	Correspondance de sous-chaîne pour ^^^ où ^ représente un caractère espace
	Widget^ où ^ représente un caractère espace	Correspondance de sous-chaîne ^ où ^ représente un caractère espace
	^^^^^ où ^ représente un caractère espace	Correspondance de chaîne
\S+	Hello_w0rld	Correspondance de sous-chaîne de longueur 11 pour Hello_w0rld
	Hello^^^world où ^ représente un caractère espace	Correspondance de sous-chaîne pour Hello
	Widget^ où ^ représente un caractère espace	Correspondance de sous-chaîne pour Widget
	^^^^^ où ^ représente un caractère espace	Pas de correspondance
\w+	D4n_v4n Vugt	Correspondance de sous-chaîne pour D4n_v4n
	^^^hello où ^ représente un caractère espace	Correspondance de sous-chaîne pour hello
	blah	Correspondance de chaîne
	x#1	Pas de correspondance
	foo bar	Pas de correspondance
\W	Hello there	Correspondance de sous-chaîne de longueur 1 pour les caractères espaces de séparation
	~	Correspondance de chaîne
	aa	Pas de correspondance
	a	Pas de correspondance
	-	Pas de correspondance
	^^^^444 == 5 où ^ représente un caractère espace	Correspondance de sous-chaîne de longueur 1 pour le premier ^ où ^ représente un caractère espace
\w+\s*=\s*\d+	x = 123	Correspondance de chaîne
	count0=555	Correspondance de chaîne
	my_var=66	Correspondance de chaîne
	0101010=0	Correspondance de chaîne
	xyz = e	Pas de correspondance
	delta=	Pas de correspondance
	==8	Pas de correspondance
[[:alnum:]]+	1234	Correspondance de chaîne
	...D4N13L	Correspondance de sous-chaîne pour D4N13L

Tableau 110. Exemples d'expressions régulières. (suite)

Cette expression...	Appliquée à cette chaîne...	Résulte en...
[[:alpha:]]+	Bubble	Correspondance de chaîne
	...DANI3L	Correspondance de sous-chaîne pour DANI
	69	Pas de correspondance
[:blank:]]+	alpha^^^^and beta où ^ représente un caractère espace	Correspondance de sous-chaîne pour ^^^^ où ^ représente un caractère espace
	Animal	Pas de correspondance
	chaîne vide	Pas de correspondance
[[:space:]]+	alpha^^^^and beta où ^ représente un caractère espace	Correspondance de sous-chaîne pour ^^^^ où ^ représente un caractère espace
	Animal	Pas de correspondance
	chaîne vide	Pas de correspondance
[[:cntrl:]]+	...Hello W0rld!	Pas de correspondance
	chaîne vide	Pas de correspondance
[[:graph:]]+	hello world	Correspondance de sous-chaîne pour hello
	^^^^^ où ^ représente un caractère espace	Pas de correspondance
	^^^!? où ^ représente un caractère espace	Correspondance de sous-chaîne pour !?
[:lower:]]+	Animal	Correspondance de sous-chaîne pour nimal
	ABC	Pas de correspondance
	0123	Pas de correspondance
	foobar	Correspondance de chaîne
	^^^0blaH! où ^ représente un caractère espace	Correspondance de sous-chaîne pour bla
[_[:lower:]]+	foo_bar	Correspondance de chaîne
	this_thinG!!!	Correspondance de sous-chaîne pour _thin
[[:upper:]]+	OUI	Correspondance de chaîne
	#define MAX 100	Correspondance de sous-chaîne pour MAX
	f00 b4r	Pas de correspondance
[[:print:]]+	hello world	Correspondance de chaîne
	^^^^^ où ^ représente un caractère espace	Correspondance de chaîne
[[:punct:]]+	didn't	Correspondance de sous-chaîne pour '
	Animal	Pas de correspondance
[[:xdigit:]]+	432987424323921 87ffe	Correspondance de chaîne
	x = bAAdF00d	Correspondance de sous-chaîne pour bAAdF00d
	4327afeffegokpoj	Correspondance de sous-chaîne pour 4327afeffe
c:\\temp	c:\temp	Correspondance de chaîne

Exemple

La règle dans l'exemple suivant envoie un courrier électronique à une liste de destinataires dès que le fichier `/home/book.txt` est créé sur le poste de travail `editor_wrkstn`.

```
<?xml version="1.0"?>
<eventRuleSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns="http://www.ibm.com/xmlns/prod/tws/1.0/event-management/rules"
      xsi:schemaLocation="http://www.ibm.com/xmlns/prod/tws/1.0/
                          event-management/rules/EventRules.xsd">
   <eventRule name="FILE_MONITOR_FILE_CREATED" ruleType="filter" isDraft="no">
      <description>Event: File Created; Action: Send mail</description>
      <validity to="2012-04-22" />
      <eventCondition name="fileCrtEvt1" eventProvider="FileMonitor" eventType="FileCreated">
      <scope>/HOME/BOOK.TXT ON EDITOR_WRKSTN</scope>
         <filteringPredicate>
          <attributeFilter name="FileName" operator="eq">
         <value>/home/book.txt</value>
          </attributeFilter>
         <attributeFilter name="SampleInterval" operator="eq">
         <value>60</value>
         </attributeFilter>
         <attributeFilter name="Workstation" operator="eq">
         <value>editor_wrkstn</value>
         </attributeFilter>
         <attributeFilter name="Hostname" operator="eq">
         <value>ceditor</value>
         </attributeFilter>
                  </filteringPredicate>
      </eventCondition>
      <action actionProvider="MailSender" actionType="SendMail" responseType="onDetection">
         <description>Send an eMail</description>
         <scope>SAUL.FELLOW@US.IBM.COM, ISAAC.LINGER@US.IBM.COM : THE EXPECTED FILE
                HAS BEEN CREATED!</scope>
         <parameter name="Cc">
            <value>william.waulkner@us.ibm.com</value>
         </parameter>
         <parameter name="Bcc">
            <value>ernest.demingway@us.ibm.com</value>
         </parameter>
         <parameter name="Body">
            <value>The expected file was created!
                   The book is ready to be published.</value>
         </parameter>
         <parameter name="To">
            <value>saul.fellow@us.ibm.com, isaac.linger@us.ibm.com</value>
         </parameter>
         <parameter name="Subject">
            <value>The expected file was created!</value>
         </parameter>
      </action>
   </eventRule>
</eventRuleSet>
```

Evénements TWSApplicationMonitor

Les événements `TWSApplicationMonitor` concernent les processus Tivoli Workload Scheduler, le système de fichiers et la boîte de messages. Il s'agit de :

- `MessageQueuesFilling`
- `TivoliWorkloadSchedulerFileSystemFilling`
- `TivoliWorkloadSchedulerProcessNotRunning`

Les événements `TWSApplicationMonitor` ne sont pas pris en charge sur les systèmes IBM i.

Cliquez ici pour afficher les zones Dynamic Workload Console de chaque type d'événement.

Remarque : Utilisateurs PDF, les tables de paramètre ci-dessus constituent un fichier html référencé dans le fichier PDF. Il n'est pas sauvegardé localement avec

le fichier PDF dans le centre de documentation. Vous devez dans un premier temps le consulter dans le centre de documentation avant de le sauvegarder ou de l'imprimer.

Exemple

La règle dans l'exemple suivant consigne des messages d'avertissement LOGMSG01W dès que les files d'attente des messages de intercom ou de mailbox sur le poste de travail NC122160 atteignent 70 pour cent de leur taille.

```
<?xml version="1.0"?>
<eventRuleSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns="http://www.ibm.com/xmlns/prod/tws/1.0/event-management/rules"
      xsi:schemaLocation="http://www.ibm.com/xmlns/prod/tws/1.0/
                          event-management/rules/EventRules.xsd">
   <eventRule name="TWS_APPL_MONITOR_MESSAGE_QUEUES_FILLING" ruleType="filter" isDraft="no">
      <description>Event: Message queues filling; Action: Message logger</description>
      <timeZone>America/Los_Angeles</timeZone>
      <validity from="2011-04-25"/>
      <activeTime end="17:00:00"/>
      <eventCondition name="twsMesQueEvt1" eventProvider="TWSApplicationMonitor"
 eventType="TWSMessageQueues">
      <scope>INTERCOM, MAILBOX FILLED UP 70% ON NC122160</scope>
         <filteringPredicate>
          <attributeFilter name="MailboxName" operator="eq">
         <value>intercom</value>
         <value>mailbox</value>
          </attributeFilter>
         <attributeFilter name="FillingPercentage" operator="ge">
         <+value>70</value>
         </attributeFilter>
         <attributeFilter name="Workstation" operator="eq">
         <value>NC122160</value>
         </attributeFilter>
         <attributeFilter name="SampleInterval" operator="eq">
         <value>60</value>
         </attributeFilter>
                  </filteringPredicate>
      </eventCondition>
      <action actionProvider="MessageLogger" actionType="MSGLOG" responseType="onDetection">
         <description>Write a warning message log</description>
         <scope>OBJECT=LOGMSG01W MESSAGE=MAILBOX AND/OR INTERCOM QUEUE
                HAS REACHED 70% OF FILLING</scope>
         <parameter name="ObjectKey">
            <value>LOGMSG01W</value>
         </parameter>
         <parameter name="Message">
            <value>Mailbox and/or Intercom queue has reached 70% of filling</value>
         <parameter name="Severity">
            <value>Warning</value>
      </action>
   </eventRule>
</eventRuleSet>
```

Fournisseurs d'actions et définitions

Cette section détaille les types d'action des fournisseurs d'actions suivants :

- GenericAction
- MailSender
- MessageLogger
- SmartCloud Control Desk
- TBSMEventForwarder
- TECEventForwarder
- TWSAction
- «TWSForZosAction», à la page 723

Actions GenericAction

Ce fournisseur met en oeuvre une action unique nommée RunCommand qui exécute des commandes non Tivoli Workload Scheduler. Les commandes sont exécutées sur le même ordinateur que celui où s'exécute le processeur d'événements.

Seul l'*utilisateur_TWS* est autorisé à exécuter la commande.

Important : Si la commande inclut la redirection de sortie (par l'utilisation d'un ou de deux signes >), insérez-la dans le fichier exécutable, puis définissez le nom de fichier comme l'argument de la propriété Command.

Cliquez ici pour afficher les zones Dynamic Workload Console de RunCommand.

Remarque : Utilisateurs PDF, les tables de paramètre ci-dessus constituent un fichier html référencé dans le fichier PDF. Il n'est pas sauvegardé localement avec le fichier PDF dans le centre de documentation. Vous devez dans un premier temps le consulter dans le centre de documentation avant de le sauvegarder ou de l'imprimer.

Exemple

La règle dans l'exemple suivant exécute la commande **ps -ef** pour lister tous les processus en cours d'exécution sur un poste de travail UNIX lorsqu'un paramètre invalide est découvert sur ce poste de travail. Remarquez que cette règle repose sur un événement personnalisé développé à l'aide du fournisseur d'événements GenericEventPlugIn. Pour de plus amples informations sur les types le développement de types d'événement personnalisable, voir «Définition d'événements personnalisés», à la page 145.

```
<?xml version="1.0"?>
<eventRuleSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns="http://www.ibm.com/xmlns/prod/tws/1.0/event-management/rules"
      xsi:schemaLocation="http://www.ibm.com/xmlns/prod/tws/1.0/
                          event-management/rules/EventRules.xsd">
   <eventRule name="CUSTOM_EVENT_GENERIC_EVENT" ruleType="filter" isDraft="yes">
      <description>Event: Generic Event; Action: Run Command</description>
      <activeTime start="08:30:00" end="17:30:00"/>
      <eventCondition name="genericEvt3" eventProvider="GenericEventPlugIn"
      eventType="Event1">
      <scope>INVALID PARAMETER ON WORKSTATIONVALUE</scope>
         <filteringPredicate>
           <attributeFilter name="Param1" operator="ne">
            <value>Invalid Parameter</value>
           </attributeFilter>
           <attributeFilter name="Workstation" operator="eq">
            <value>WorkstationValue</value>
           </attributeFilter>
        </filteringPredicate>
      </eventCondition>
      <action actionProvider="GenericActionPlugin" actionType="RunCommand"
      responseType="onDetection">
         <description>Run a command</description>
         <scope>PS -EF</scope>
         <parameter name="Command">
            <value>ps -ef</value>
         </parameter>
         <parameter name="WorkingDir">
            <value>/home</value>
         </parameter>
      </action>
   </eventRule>
</eventRuleSet>
```

Actions MailSender

Ce fournisseur met en oeuvre une action unique nommée SendMail qui se connecte à un serveur SMTP pour envoyer un courrier électronique. Utilisez optman pour personnaliser les attributs associés suivants (pour plus d'informations sur optman, voir *Guide d'administration*) :

- Mail sender
- SMTP server
- SMTP port number
- Mail user name
- Mail user password
- SSL

Cliquez ici pour afficher les zones Dynamic Workload Console de SendMail.

Remarque : Utilisateurs PDF, la table de paramètres ci-dessus est un fichier html référencé dans le fichier PDF. Il n'est pas sauvegardé localement avec le fichier PDF dans le centre de documentation. Vous devez dans un premier temps le consulter dans le centre de documentation avant de le sauvegarder ou de l'imprimer.

Actions MessageLogger

Ce fournisseur met en oeuvre une action unique nommée MMSGLOG qui journalise l'occurrence d'une situation dans une base de données d'audit interne. Le nombre d'entrées de la base de données d'audit est configurable. Il y a un nettoyage automatique basé sur une règle FIFO.

Cliquez ici pour afficher les zones Dynamic Workload Console de MSGLOG.

Remarque : Utilisateurs PDF, la table de paramètres ci-dessus est un fichier html référencé dans le fichier PDF. Il n'est pas sauvegardé localement avec le fichier PDF dans le centre de documentation. Vous devez dans un premier temps le consulter dans le centre de documentation avant de le sauvegarder ou de l'imprimer.

Actions SmartCloud Control Desk

Ce fournisseur met en oeuvre une action unique, nommée OpenTicket, qui ouvre un ticket sur un SmartCloud Control Desk par défaut. Utilisez optman pour spécifier le serveur SmartCloud Control Desk en définissant les options globales sccdUrl, sccdName et sccdUserPassword. Pour plus d'informations sur optman, voir le document *Guide d'administration*.

Cliquez ici pour afficher les zones Dynamic Workload Console de OpenTicket.

Remarque : Utilisateurs PDF, la table de paramètres ci-dessus est un fichier html référencé dans le fichier PDF. Il n'est pas sauvegardé localement avec le fichier PDF dans le centre de documentation. Vous devez dans un premier temps le consulter dans le centre de documentation avant de le sauvegarder ou de l'imprimer.

Actions TBSMEventForwarder

Ce fournisseur met en oeuvre une action unique, nommée TBSMFWD, qui transmet l'événement à un serveur Tivoli Business Systems Manager externe ou à toute autre application capable d'écouter les événements au format TBSM (par exemple, Netcool/OMNIbus). Le fournisseur utilise un serveur Sonde EIF par défaut dont le nom d'hôte et le port peuvent être définis dans les options globales TECServerName et TECServerPort à l'aide de l'utilitaire optman.

Remarque : Les options TECServerName et TECServerPort sont toutes les deux utilisées pour les applications traitant les événements au format TEC ou TBSM. Pour plus d'informations sur optman, voir le document *Guide d'administration*.

La sonde IEF utilisée en tant que destinataire peut être remplacée par des paramètres d'action.

Cliquez ici pour afficher les zones Dynamic Workload Console de TBSMFWD.

Remarque : Utilisateurs PDF, la table de paramètres ci-dessus est un fichier html référencé dans le fichier PDF. Il n'est pas sauvegardé localement avec le fichier PDF dans le centre de documentation. Vous devez dans un premier temps le consulter dans le centre de documentation avant de le sauvegarder ou de l'imprimer.

Configuration de Tivoli Business Services Manager pour la réception d'événements

Pour configurer Tivoli Business Systems Manager en vue de la réception d'événements de Tivoli Workload Scheduler, sur la Sonde EIF, vous devez copier le fichier utilities/tivoli_eif_tws.rules fourni avec le DVD de Tivoli Workload Scheduler. Ensuite, vous devez éditer le fichier tivoli_eif.rules en ajoutant la ligne suivante :

```
include "tivoli_eif_tws.rules"
```

Actions TECEventForwarder

Ce fournisseur met en oeuvre une action unique, nommée TECFWD, qui transmet l'événement à un serveur TEC (Tivoli Enterprise Console) externe ou à toute autre application capable d'écouter les événements au format TEC. Le fournisseur utilise un serveur Sonde EIF par défaut dont le nom d'hôte et le port peuvent être définis dans les options globales TECServerName et TECServerPort à l'aide de l'utilitaire optman. Pour plus d'informations sur optman, voir le document *Guide d'administration*.

Le TEC utilisé en tant que destinataire peut être remplacé par des paramètres d'action.

Cliquez ici pour afficher les zones Dynamic Workload Console de TECFWD.

Remarque : Utilisateurs PDF, la table de paramètres ci-dessus est un fichier html référencé dans le fichier PDF. Il n'est pas sauvegardé localement avec le fichier PDF dans le centre de documentation. Vous devez dans un premier temps le consulter dans le centre de documentation avant de le sauvegarder ou de l'imprimer.

Actions TWSAction

Les actions TWSAction sont :
- SubmitJobStream
- SubmitJob
- SubmitAdHocJob
- ReplyPrompt

Cliquez ici pour afficher les zones Dynamic Workload Console de chaque type d'action.

Remarque : Utilisateurs PDF, les tables de paramètre ci-dessus constituent un fichier html référencé dans le fichier PDF. Il n'est pas sauvegardé localement avec

le fichier PDF dans le centre de documentation. Vous devez dans un premier temps le consulter dans le centre de documentation avant de le sauvegarder ou de l'imprimer.

Utilisation de la propriété SchedTimeResolutionCriteria de l'action SubmitJob

Utilisez cette propriété pour faire correspondre le travail en question avec une instance spécifique du flot de travaux qui le contient (défini avec la propriété JobStreamName) en fonction de l'heure planifiée du flot de travaux. Les valeurs que vous pouvez définir sont les suivantes :

Précédent
Le travail est soumis à l'instance de flot de travaux précédente la plus proche dans le plan.

Suivant
Le travail est soumis à l'instance de flot de travaux suivante la plus proche dans le plan.

Tout Le travail est soumis à l'instance de flot de travaux précédente ou suivante la plus proche dans le plan.

TWSForZosAction

Ce fournisseur implémente une action unique nommée AddJobStream qui ajoute une occurrence d'application (flot de travaux) au plan actuel sur IBM Tivoli Workload Scheduler for z/OS. Ce fournisseur est à utiliser dans des configurations de planification de bout en bout Tivoli Workload Scheduler.

La description d'application de l'occurrence à ajouter doit exister dans la base de données AD de IBM Tivoli Workload Scheduler for z/OS.

Cliquez ici pour afficher les zones Dynamic Workload Console de AddJobStream.

Remarque : Utilisateurs PDF, la table de paramètres ci-dessus est un fichier html référencé dans le fichier PDF. Il n'est pas sauvegardé localement avec le fichier PDF dans le centre de documentation. Vous devez dans un premier temps le consulter dans le centre de documentation avant de le sauvegarder ou de l'imprimer.

Exemple

Dans cet exemple, une entreprise pharmaceutique utilise la règle ZOSRULE031 pour produire un planning de distribution des marchandises sous le contrôle de département DISTR07. Dès que la liste des marchandises commandées qui sont à livrer dans le mois à venir est prête et placée dans le fichier MONTHLYORDERS.TXT sur l'agent RU192298 d'une filiale, le système centralisé ajoute l'application (flot de travaux) ADFIRST au plan actuel. ADFIRST contient les opérations (travaux) qui produisent un calendrier de livraison optimisé pour le mois prochain.

```
<?xml version="1.0"?>
<eventRuleSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns="http://www.ibm.com/xmlns/prod/tws/1.0/event-management/rules"
      xsi:schemaLocation="http://www.ibm.com/xmlns/prod/tws/1.0/
                          event-management/rules/EventRules.xsd">
   <eventRule name="ZOSRULE031" ruleType="filter" isDraft="no">
      <eventCondition name="fileCrtEvt19" eventProvider="FileMonitor"
      eventType="FileCreated">
      <scope>/PRODORDER/MONTHLYORDERS.TXT ON RU192298</scope>
         <filteringPredicate>
           <attributeFilter name="Param1" operator="ne">
```

```
            <value>/prodorder/monthlyorders.txt</value>
           </attributeFilter>
           <attributeFilter name="SampleInterval" operator="eq">
            <value>60</value>
           </attributeFilter>
           <attributeFilter name="Workstation" operator="eq">
           <value>RU192298</value>
           </attributeFilter>
         </filteringPredicate>
       </eventCondition>
       <action actionProvider="TWSForZosAction" actionType="AddJobStream"
       responseType="onDetection">
         <scope>
           ADD JOBSTREAM ADFIRST[DEADLINE OFFSET: 0001] WITH OWNER DISTR07 IN PLAN
         </scope>
          <parameter name="HoldAll">
             <value>false</value>
          </parameter>
          <parameter name="Priority">
             <value>5</value>
          </parameter>
          <parameter name="JobStreamDeadlineOffset">
             <value>0001</value>
          </parameter>
          <parameter name="JobStreamName">
             <value>ADFIRST</value>
          </parameter>
          <parameter name="OwnerDescription">
             <value>Owner description</value>
          </parameter>
          <parameter name="Owner">
             <value>distr07</value>
          </parameter>
          <parameter name="DependenciesResolution">
             <value>All</value>
          </parameter>
          <parameter name="AuthorityGroup">
             <value>AuthGrpBase</value>
          </parameter>
          <parameter name="Parm_1">
             <value>var1=value1</value>
          </parameter>
          <parameter name="Parm_2">
             <value>var2=value2</value>
          </parameter>
          <parameter name="JCLVariableTable">
             <value>VarTableZos01</value>
          </parameter>
          <parameter name="JobStreamDescription">
             <value>Ce flot de travaux contient des travaux qui traitent
                    des commandes pour le propriétaire DISTR07.</value>
          </parameter>
          <parameter name="Group">
             <value>GroupBase</value>
          </parameter>
       </action>
    </eventRule>
</eventRuleSet>
```

Annexe B. Référence de schéma Job Submission Description Language

Cette section de référence spécifie la sémantique et la structure du langage Job Submission Description Language (JSDL) qui est destiné à être utilisé spécifiquement avec Dynamic Workload Broker. Le schéma JSDL permet de décrire les exigences du travail pour la soumission sur les ressources. Dynamic Workload Broker analyse l'environnement informatique et affecte les meilleures ressources disponibles pour exécuter le travail, en fonction des exigences que vous spécifiez.

Introduction

Job Submission Description Language (JSDL) est un langage permettant de décrire les exigences du travail pour la soumission aux ressources. Il contient un vocabulaire et un schéma XML normatif qui facilite l'expression de ces exigences sous forme d'un ensemble d'éléments XML.

Les fichiers JSDL adhèrent à la syntaxe et à la sémantique XML telles qu'elles sont définies dans le schéma JSDL.

Structure de document Job Submission Description Language

Un fichier JSDL est décrit à l'aide de la syntaxe XML et adhère à la syntaxe et à la sémantique XML. La syntaxe XML est une norme informatique et n'est pas expliquée dans ce manuel. Le fichier JSDL adhère également à des règles de syntaxe JSDL, telles qu'elles sont expliquées dans «Types d'éléments Job Submission Description Language», à la page 728 et dans «Eléments JSDL», à la page 731.

Le fichier JSDL est constitué d'éléments (simples ou complexes) et de types. Les éléments complexes contiennent d'autres éléments tandis que les éléments simples n'en contiennent pas. Une spécification de type réalise un contrôle de syntaxe sur la valeur spécifiée pour l'élément auquel elle se réfère. Par exemple, l'élément **physicalMemory** adhère au type jsdl:NumericRangeType. Le type jsdl:NumericRangeType spécifie que vous pouvez affecter à cet élément soit une valeur numérique spécifique, soit une valeur de plage numérique. Aucun autre type de valeur n'est pris en charge pour l'élément **physicalMemory**.

Le fichier JSDL est organisé en une structure hiérarchique où l'élément **jobDefinition** est l'élément racine. L'élément **jobDefinition** contient tous les éléments qui décrivent le travail et leurs attributs.

La définition du pseudo-schema est du type suivant :

```
< jobDefinition >
  <annotation ... />?
  <category>... />*
  <variables ... />?
  <application ... />
  <resources ... />?
  <relatedResources ... />*
  <optimization ... >?
  <scheduling ...>?
</jobDefinition>
```

Le tableau 111 fournit une vue en tableau du fichier JSDL en indiquant les relations hiérarchiques entre les éléments contenus dans l'élément **jobDefinition**.

Tableau 111. Structure hiérarchique du fichier JSDL

Premier niveau	**Deuxième niveau**	**Troisième niveau**	**Quatrième niveau**
annotation			
category			
variables	stringVariable		
	uintVariable		
	doubleVariable		
application	script		
	arguments	value	
	environment	nom de la variable	
	credential	username	
		groupname	
		password	
	j2ee	invoker	type
		jms	connFactory
			destination
			message
		ejb	jndiHome
		credential	userName
			password
			JAASalias

Tableau 111. Structure hiérarchique du fichier JSDL (suite)

Premier niveau	Deuxième niveau	Troisième niveau	Quatrième niveau
resources	candidateHosts	hostName	
	candidateCPUs	cpu	speed
	physicalMemory		
	virtualMemory		
	candidateOperating Systems	operatingsystems	
	fileSystem		
	logicalResource		
	group		
	properties	and	and
			or
			requirement
		or	and
			or
			requirement
		requirement	and
			or
			requirement
	allocation		
	relationship		
	candidateResources (réservé pour une utilisation interne)	endpointReference (réservé pour une utilisation interne)	
relatedResources	logicalResource		
	group		
	properties	and	and
			or
			requirement
		or	and
			or
			requirement
		requirement	and
			or
			requirement
	allocation		
	relationship		
	candidateResources (réservé pour une utilisation interne)	endpointReference (réservé pour une utilisation interne)	
optimization	objective		
	ewlm		

Tableau 111. Structure hiérarchique du fichier JSDL (suite)

Premier niveau	Deuxième niveau	Troisième niveau	Quatrième niveau
scheduling	maximumResource WaitingTime		
	estimatedDuration		
	priority		
	recoveryActions	action	parameters
			credential
			tpmaddress
			workflow

La syntaxe JSDL utilise les conventions de style BNF pour les éléments et les attributs :

? Indique que l'élément ou l'attribut est facultatif et peut être spécifié une fois.

* Indique que l'élément ou l'attribut est facultatif et peut être spécifié zéro ou plusieurs fois.

+ Indique que l'élément ou l'attribut est obligatoire et peut être spécifié une ou plusieurs fois.

[...] Indique que les éléments et attributs contenus dans les parenthèses forment un groupe.

| Indique que deux ou plusieurs éléments ou attributs s'excluent mutuellement.

Types d'éléments Job Submission Description Language

La spécification JSDL utilise un certain nombre de types de schéma XML standard. Elle utilise également un certain nombre de types spécifiques à la description des exigences de travail.

Ces deux types réalisent une vérification de syntaxe sur la valeur qui peut être affectées à chaque élément dans le fichier JSDL. Par exemple, l'élément **physicalMemory** adhère au type jsdl:NumericRangeType. Le type jsdl:NumericRangeType spécifie que vous pouvez affecter à cet élément soit une valeur numérique spécifique, soit une valeur de plage numérique. Aucun autre type de valeur n'est pris en charge pour l'élément **physicalMemory**.

Types de schéma XML normatifs

La spécification JSDL adopte les types de schéma XML normatifs (xsd) répertoriés ci-dessous. La syntaxe XML est une norme informatique et n'est pas expliquée dans ce manuel.

- xsd:any
- xsd:anyURI
- xsd:boolean
- xsd:double
- xsd:DoubleVariableType
- xsd:duration

- xsd:IDREF
- xsd:NCName
- xsd:PriorityType
- xsd:QName
- xsd:string
- xsd:unsignedInt
- xsd:UnsignedIntVariableType

Types JSDL

Les types suivants sont spécifiques à la syntaxe JSDL :

StringVariableExpressionType
Un type d'expression de variable de chaîne est un type simple dans lequel vous pouvez indiquer une expression de variable qui peut contenir une ou plusieurs références de variable, telles que ${var}, tout caractère et toute chaîne. Voici le schéma de syntaxe pour ce type :

```
<...>
<xsd:simpleType name="StringVariableExpressionType">
  <xsd:union>
   <xsd:simpleType>
    <xsd:restriction base='xsd:string' />
   </xsd:simpleType>
   <xsd:simpleType>
    <xsd:restriction base='xsd:string'>
     <xsd:pattern
      value=".*\t*\r*\n*((\$\{[a-zA-Z_]+
          [0-9a-zA-Z_\.\-]*\})+[^\{]*[.\n]*)+" />
    </xsd:restriction>
   </xsd:simpleType>
  </xsd:union>
 </xsd:simpleType>
</...>
```

DoubleVariableExpressionType
Un type d'expression de variable double est une type simple dans lequel vous pouvez spécifier une expression de variable qui peut contenir une référence de variable, telle que ${var} ou une valeur double. Voici le schéma de syntaxe pour ce type :

```
<...>
 <xsd:simpleType name="DoubleVariableExpressionType">
  <xsd:union>
   <xsd:simpleType>
    <xsd:restriction base='xsd:double' />
   </xsd:simpleType>
   <xsd:simpleType>
    <xsd:restriction base='xsd:string'>
     <xsd:pattern value="[\n\r\t ]*($\{[a-zA-Z_]+
               [0-9a-zA-Z_\.\-]*\})[\n\r\t ]*" />
    </xsd:restriction>
   </xsd:simpleType>
  </xsd:union>
 </xsd:simpleType>

</...>
```

UnsignedIntVariableExpressionType
Un type d'expression de variable non signé est un type simple dans lequel vous pouvez spécifier une expression de variable qui peut contenir une référence de variable, telle que ${var}, ou une valeur entière non signée. Voici le schéma de syntaxe pour ce type :

```
<...>
<xsd:simpleType name="UnsignedIntVariableExpressionType">
  <xsd:union>
   <xsd:simpleType>
    <xsd:restriction base='xsd:unsignedInt' />
   </xsd:simpleType>
   <xsd:simpleType>
    <xsd:restriction base='xsd:string'>
     <xsd:pattern value="[\n\r\t ]*($\{[a-zA-Z_]+
               [0-9a-zA-Z_\.\-]*\})[\n\r\t ]*" />
    </xsd:restriction>
   </xsd:simpleType>
  </xsd:union>
 </xsd:simpleType>

</...>
```

NotEmptyStringVariableExpressionType

Un type d'expression de variable de chaîne est un type simple qui permet la spécification d'une expression de variable pouvant contenir une ou plusieurs références de variable telles que ${var}, et être facultativement associée à tout caractère ou à une chaîne simple. Cette expression de variable ne peut pas être vide. Voici le schéma de syntaxe pour ce type :

```
<xsd:simpleType name="NotEmptyStringVariableExpressionType">
  <xsd:union>
   <xsd:simpleType>
    <xsd:restriction base='xsd:string'>
     <xsd:minLength value="1"/>
    </xsd:restriction>
   </xsd:simpleType>
   <xsd:simpleType>
    <xsd:restriction base='xsd:string'>
     <xsd:pattern
      value=".*\t*\r*\n*((\$\{[a-zA-Z_]+
                    [0-9a-zA-Z_\.\-]*\})+[^\{]*[.\n]*)+" />
    </xsd:restriction>
   </xsd:simpleType>
  </xsd:union>
 </xsd:simpleType>
```

NumericRangeOnlyType

Une valeur de plage numérique est un type complexe qui permet la définition des intervalles et plages supérieurs à, inférieurs à ou contenus dans la valeur spécifiée. Tous les nombres indiqués sont des expressions de variable double. Voici le schéma de syntaxe pour ce type :

```
<...>
 <minimum>jsdl:DoubleVariableExpressionType</minimum> ?
 <maximum> jsdl:DoubleVariableExpressionType</maximum> ?
</...>
```

NumericRangeType

Une valeur de plage numérique est un type complexe qui permet la définition des valeurs ou plages exactes. Tous les nombres indiqués sont des expressions de variable double. Voici le schéma de syntaxe pour ce type :

```
<...>
 <exact>jsdl:DoubleVariableExpressionType</exact> |
 <range>jsdl:NumericRangeOnlyType</range>
</...>
```

StringRangeOnlyType

Une valeur de plage de chaîne est un type complexe qui permet la définition des intervalles et plages supérieurs à, inférieurs à ou contenus

dans la valeur spécifiée. Tous les nombres et chaînes indiqués sont des expressions de variable de chaîne. Voici le schéma de syntaxe pour ce type :

```
<...>
 <minimum>jsdl:StringVariableExpressionType</minimum> ?
 <maximum>jsdl:StringVariableExpressionType</maximum> ?
</...>
```

StringRangeType

Une valeur de plage de chaîne est un type complexe qui permet la définition de valeurs exactes sous forme d'expressions de variable de chaîne ou de plages qui peuvent être appliquées aux types d'entier ou de chaîne. Voici le schéma de syntaxe pour ce type :

```
<...>
 <exact>jsdl:StringVariableExpressionType</exact> |
 <range>jsdl:StringRangeOnlyType</range>
</...>
```

Eléments JSDL

L'élément principal JSDL défini contient la sémantique des éléments qui sont définis par le fichier JSDL.

Le fichier JSDL est constitué d'éléments (simples ou complexes) et de types. Les éléments complexes contiennent d'autres éléments tandis que les éléments simples n'en contiennent pas. Une spécification de type réalise un contrôle de syntaxe sur la valeur spécifiée pour l'élément auquel elle se réfère.

Voici une liste des éléments contenus dans la syntaxe JSDL :

élément jobDefinition

Définition

Cet élément décrit le travail et ses exigences. Il s'agit de l'élément racine du document JSDL. Cet attribut est obligatoire.

Type Le type de cet élément est jsdl:JobDefinitionType. Il peut contenir les éléments suivants :

- annotation
- category
- variables
- application
- resources
- relatedResources
- optimization
- scheduling

Attributs

name Nom du travail indiqué par l'utilisateur. Le type de cet attribut est **xsd:NCName**. Ce nom doit commencer par un caractère alphabétique et peut comporter les symboles suivants : soulignement (_), moins (-), et point (.). Les espaces, les caractères spéciaux et les caractères accentués et les nombres ne sont pas pris en charge. Cet attribut est obligatoire. Le nom que vous définissez pour cette zone identifie de façon unique la définition de travail lorsqu'elle est sauvegardée dans la base de données de référentiel

des travaux. Après avoir sauvegardé la définition de travail dans la base de données, vous pouvez la soumettre à l'aide de Dynamic Workload Console ou de la ligne de commande.

description
Chaîne spécifiant une courte description de la définition de travail. Le type de cet attribut est **xsd:string**. Cet attribut est facultatif.

targetNamespace
Identificateur URI spécifiant l'espace de nom cible de la définition de travail. Le type de cet attribut est **xsd:anyURI**. Cet attribut est obligatoire.

Pseudo-schéma

```
<jobDefinition
 name="xsd:NCName"
 description="xsd:string"?
 xsd:anyAttribute##other>
 <annotation ... />?
 <category>.../>*
 <variables ... />?
 <application ... />
 <resources ... />?
 <relatedResources .../>*
 <optimization ...>?
 <scheduling ...>?
 <xsd:any##other/>*
</jobDefinition>
```

élément annotation

Définition
Cet élément fournit des informations descriptives et lisibles sur la définition des travaux. Cet élément est facultatif et peut être spécifié une fois.

Type Le type de cet élément est xsd:string.

Attributs
Aucun attribut n'est défini.

Pseudo-schéma

```
<annotation
 xsd:anyAttribute##other>
 xsd:string
 <xsd:any##other/>*
</application>
```

élément category

Définition
Cette élément décrit la catégorie de travail qui vous aide à classer le travail par catégorie. Un travail peut avoir plusieurs catégories, par exemple : Education_DB, Financial_Dept, Asset_Management. Cette valeur peut être toute valeur de chaîne. Cet élément est facultatif et peut être spécifié zéro ou plusieurs fois.

Type Le type de cet élément est xsd:string.

Attributs
Aucun attribut n'est défini.

Pseudo-schéma

```
<category>
 xsd:string
</category>
```

élément variables

Définition

Cet élément décrit la liste des variables définies dans le fichier JSDL. Les trois types de variable suivants sont pris en charge :

- Chaîne
- Double
- Entier

La valeur de variable peut être référencée dans d'autres parties du document JSDL en spécifiant : ${variable name} Une variable référencée peut être l'une des variables définies dans le fichier JSDL avec l'élément **variable** ou elle peut être définie lors de la soumission du travail. La substitution peut être effectuée par le serveur Dynamic Workload Broker dans différentes phases du traitement du travail. Dans chaque phase, le serveur Dynamic Workload Broker tente de faire correspondre toutes les références de variable non encore substituées par les variables définies. Cet élément est facultatif et peut être spécifié une fois.

Type Le type de cet élément est jsdl:VariablesType. Il peut contenir les éléments suivants :

- stringVariable
- uintVariable
- doubleVariable

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
<variables
 xsd:anyAttribute##other>
 <stringVariable ...>*
 <uintVariable ...>*
 <doubleVariable ...>*
 <xsd:any##other/>*
</variables>?
```

élément stringVariable

Définition

Cet élément décrit une variable en spécifiant le nom de variable et la valeur de chaîne par défaut affectée. Cet élément est facultatif et peut être spécifié zéro ou plusieurs fois.

Type Le type de cet élément est jsdl:StringVariableType.

Attributs

Les attributs suivants sont définis :

name Cet attribut spécifie le nom de la variable. Le type de cet attribut est xsd:NCName. Cet attribut est obligatoire.

description

Cet attribut spécifie la description de la variable. Le type de cet attribut est xsd:string. Cet attribut est facultatif.

Pseudo-schéma

```
<stringVariable
 name="xsd:NCName"
 description="xsd:string"?
 xsd:anyAttribute##other>
 xsd:string
 <xsd:any##other/>*
</stringVariable>
```

élément doubleVariable

Définition

Cet élément décrit une variable en spécifiant le nom de la variable ainsi que la valeur double par défaut affectée. Cet élément est facultatif et peut être spécifié zéro ou plusieurs fois.

Type Le type de cet élément est xsd:DoubleVariableType.

Attributs

Les attributs suivants sont définis :

name Cet attribut spécifie le nom de la variable. Le type de cet attribut est xsd:NCName. Cet attribut est obligatoire.

description
Cet attribut spécifie la description de la variable. Le type de cet attribut est xsd:string. Cet attribut est facultatif.

Pseudo-schéma

```
<doubleVariable
 name="xsd:NCName"
 description="xsd:string"?
 xsd:anyAttribute##other>
 xsd:double
 <xsd:any##other/>*
</doubleVariable>
```

élément uintVariable

Définition

Cet élément décrit une variable en spécifiant le nom de la variable et la valeur d'entier non signé par défaut affectée. Cet élément est facultatif et peut être spécifié zéro ou plusieurs fois.

Type Le type de cet élément est xsd:UnsignedIntVariableType.

Attributs

Les attributs suivants sont définis :

name Cet attribut spécifie le nom de la variable. Le type de cet attribut est xsd:NCName. Cet attribut est obligatoire.

description
Cet attribut spécifie le nom de la variable. Le type de cet attribut est xsd:string. Cet attribut est facultatif.

Pseudo-schéma

```
<uintVariable
 name="xsd:NCName"
 description="xsd:string"?
 xsd:anyAttribute##other>
 xsd:unsignedInt
 <xsd:any##other/>*
</uintVariable>
```

élément application

Définition

Cet élément décrit l'application qui doit s'exécuter avec les paramètres associés. Cet élément est obligatoire et peut être spécifié une fois.

Type Le type de cet élément est jsdl:ApplicationType.

Attributs

Les attributs suivants sont définis :

name Indique le type de l'application. Les valeurs prises en charge sont les suivantes :

Executable
Un fichier utilisé pour effectuer différentes fonctions ou opérations sur un ordinateur.

J2EE Une application basée sur Java 2 Platform Enterprise Edition (J2EE).

Cet attribut est facultatif et peut être spécifié une fois.

description
Indique le nom de l'application. Le type de cet attribut est xsd:string. Cet attribut est facultatif.

version
Indique la version de l'application. Le type de cet attribut est xsd:string. Cet attribut est facultatif.

Pseudo-schéma

```
<application
 name="xsd:NCName"
 description="xsd:string"?
 version="xsd:string"?
 xsd:anyAttribute##other>
 <xsd:any##other/>*
</application>
```

élément resources

Définition

Cet élément contient les exigences de ressources du travail pour lequel une correspondance doit être trouvée sur l'ordinateur cible afin qu'un travail soit affecté à ce système. Les éléments de l'exigence de ressources contenues sont combinés dans une relation AND. Autrement dit, chaque exigence s'ajoute aux autres pour raffiner l'exigence de ressource correspondante et toutes les exigences doivent être satisfaites pour que l'affectation réussisse. Cet élément est facultatif et peut être spécifié une fois. S'il n'est pas présent, le serveur Dynamic Workload Broker peut choisir n'importe quel ensemble de ressources pour exécuter le travail.

Type Le type de cet élément est jsdl:ResourceType. Les types pris en charge pour cet élément sont répertoriés dans le tableau 112, à la page 767. Il peut contenir les éléments suivants :

- candidateHosts
- candidateCPUs
- physicalMemory
- virtualMemory
- candidateOperatingSystems

- fileSystem
- logicalResource
- group
- properties
- allocation
- relationship
- candidateResources

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
<resources
 xsd:anyAttribute##other>
 <candidateHosts .../>?
 <candidateCPUs .../>?
 <physicalMemory .../>?
 <virtualMemory .../>?
 <candidateOperatingSystems .../>?
 <fileSystem .../>*
 <logicalResource ...>*
 <group ...>*
 <properties ...>
 <allocation ...>*
 <relationship ...>*
 <candidateResources ...>?
 <xsd:any##other>*
</resources>
```

élément relatedResources

Définition

Cet élément est un identificateur unique pour l'exigence de ressource et doit être unique dans le document. Les exigences définies dans cet élément s'appliquent à la fois aux ressources logiques et aux systèmes d'ordinateurs. Il peut y être fait référence par les éléments **relationship**. Les éléments de l'exigence de ressources contenues sont combinés dans une relation AND. Autrement dit, chaque exigence s'ajoute aux autres pour raffiner l'exigence de ressource correspondante et toutes les exigences doivent être satisfaites pour que l'affectation réussisse. Cet élément est facultatif et peut être spécifié zéro ou plusieurs fois.

Type Le type de cet élément est jsdl:RelatedResourceType. Il peut contenir les éléments suivants :

- logicalResource
- group
- properties
- allocation
- relationship
- candidateResources

Attributs

Les attributs suivants sont définis :

id Indique l'ID interne de la ressource à associer à l'élément de ressources. Cet ID est uniquement utilisé pour référence interne dans le fichier JSDL. Le type de cet attribut est xsd:ID. Cet attribut est obligatoire.

type Indique le type de la ressource requise. Les types pris en charge sont ComputerSystem et Logical Resource. Si cet attribut n'est pas présent, le système prend en compte le type de ressource ComputerSystem. Un type de ressource est identifié par un nom unique et décrit les propriétés fournies par chaque instance de ressources. Pour plus d'informations sur les propriétés de ressource disponibles, voir tableau 112, à la page 767. Le type de cet attribut est xsd:NCName. Cet attribut est facultatif.

Pseudo-schéma

```
<relatedResources
 id="xsd:ID"
 type="xsd:NCName"?
 xsd:anyAttribute##other>
 <logicalResource ...>*
 <group ...>*
 <properties ...>
 <allocation ...>*
 <relationship ...>*
 <candidateResources ...>?
 <xsd:any##other>*
</relatedResources>
```

élément candidateHosts

Définition

Cet élément spécifie l'ensemble des hôtes désignés qui doivent être sélectionnés pour l'exécution du travail. Dynamic Workload Broker affecte le travail à l'un des hôtes de cette liste. Les hôtes spécifiés sont dans la condition OR, autrement dit, au moins l'un d'entre eux doit trouver une correspondance de ressource de système d'exploitation contenue dans la ressource cible. Si aucun des hôtes que vous spécifiez n'est disponible lors de la soumission du travail, le travail attend que l'un d'entre eux devienne disponible. Si aucun hôte ne devient disponible avant l'expiration du délai, le travail échoue. Cet attribut est facultatif.

Type Le type de cet élément est jsdl:CandidateHostsRequirementType. Il peut contenir l'élément suivant :

- hostName

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
<candidateHosts
 xsd:anyAttribute##other>
 <hostName>jsdl:StringVariableExpressionType<hostName/>+
 <xsd:any##other>*
</candidateHosts>
```

élément orderedCandidatedWorkstations

Définition

Cet élément indique la liste ordonnée des postes de travail qui sont candidats pour la sélection en fonction des informations spécifiées dans la zone d'exigence de la définition de pool dynamique. Le premier poste de travail qui correspond aux exigences est sélectionné pour le traitement.

Type Le type de cet élément est jsdl:OrderedCandidatedWorkstationsType. Il peut contenir l'élément suivant :

- poste de travail

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
<orderedCandidatedWorkstations
 xsd:anyAttribute##other>
 <workstation>jsdl:StringVariableExpressionType<workstation/>+
 <xsd:any##other>*
</orderedCandidatedWorkstations>
```

élément hostName

Définition

Cet élément indique une expression de variable de chaîne contenant le nom d'un hôte unique qui peut être sélectionné pour l'exécution du travail. Si aucun des hôtes que vous spécifiez n'est disponible lors de la soumission du travail, le travail attend que l'un d'entre eux devienne disponible. Si aucun hôte ne devient disponible avant l'expiration du délai, le travail échoue. Pour spécifier le nom d'hôte, vous pouvez utiliser une expression de variable qui peut contenir une ou plusieurs références de variable, telle que ${var}, tout caractère et toute chaîne. Les caractères génériques sont autorisés. Cet attribut est obligatoire et peut être spécifié une ou plusieurs fois.

Type Le type de cet attribut est jsdl:StringVariableExpressionType.

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
 <hostName>jsdl:StringVariableExpressionType<hostName/>


 <hostName>lab145674.example.com<hostName/>

<hostName>${my_preferred_host}<hostName/>
```

élément candidateCPUs

Définition

Cet élément spécifie l'ensemble des caractéristiques d'unité centrale que doivent satisfaire les hôtes qui peuvent être sélectionnés pour l'exécution du travail. Les combinaisons de caractéristiques spécifiées sont dans la condition OR, autrement dit au moins l'un d'entre elles doit trouver une correspondance dans la ressource cible. Si aucune des caractéristiques d'unité centrale que spécifiez n'est disponible lors de la soumission d'un travail, le travail attend que l'une d'entre elles devienne disponible. Si aucune caractéristique d'unité centrale ne devient disponible avant l'expiration du délai, le travail échoue. Cet élément est facultatif et peut être spécifié zéro ou plusieurs fois.

Type Le type de cet élément est jsdl:CandidateCPUsRequirementType. Il peut contenir l'élément suivant :

- cpu

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
<candidateCPUs
 xsd:anyAttribute##other>
 <cpu ...>*
 <xsd:any##other/>*
</candidateCPUs >?
```

élément cpu

Définition

Cet élément spécifie les caractéristiques d'unité centrale que doivent satisfaire les hôtes qui peuvent être sélectionnés pour l'exécution du travail. Les combinaisons de caractéristiques spécifiées sont dans la condition OR, autrement dit au moins l'un d'entre elles doit trouver une correspondance dans la ressource cible. Au moins l'une des unités centrales avec les caractéristiques indiquées doit être disponible pour l'exécution du travail. Si aucun des unités centrales que vous indiquez n'est disponible lors de la soumission du travail, le travail attend que l'une d'elles devienne disponibles. Si aucune caractéristique d'unité centrale ne devient disponible avant l'expiration du délai, le travail échoue. Cet élément est obligatoire et peut être spécifié une ou plusieurs fois.

Type Le type de cet élément est CPURequirementType.

Attributs

Les attributs suivants sont définis :

architecture

Indique l'architecture d'unité centrale requise pour l'exécution du travail. Cet attribut est facultatif. Les valeurs prises en charge sont les suivantes :

- parisc
- powerpc
- powerpc_64
- s390
- s390x
- sparc
- x86
- x86_64

quantity

Indique le nombre de processeurs qui doivent être disponibles sur l'ordinateur. Définir Quantity sur 0 indique que l'exigence est remplie par des ordinateurs comprenant tout nombre de processeurs. Le type de cet attribut est xsd:unsignedInt. Cet attribut est facultatif.

Pseudo-schéma

```
<cpu>
 architecture="xsd:string"?
 speed="jsdl:NumericRangeType"?
 quantity="xsd:unsignedInt"?
 xsd:anyAttribute##other>
 <xsd:any##other>*
</cpu>
```

élément speed

Définition

Cet élément indique la plage de vitesse d'unité centrale en MHz requise pour exécuter le travail. L'unité de mesure est le mégahertz. Cet élément est facultatif et peut être spécifié une fois.

Type Le type de cet élément est jsdl:NumericRangeType.

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
<speed>
 xsd:anyAttribute##other>
 <jsdl:NumericRangeType>*
<speed>
```

élément physicalMemory

Définition

Cet élément spécifie une plage ou la valeur exacte indiquant la quantité de mémoire physique disponible requise pour le travail. La quantité est exprimée en octets. Cet attribut est facultatif.

Type Le type de cet élément est jsdl:NumericRangeType.

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
<physicalMemory
 xsd:anyAttribute##other>
 jsdl:NumericRangeType
 <xsd:any##other>*
</physicalMemory>
```

élément virtualMemory

Définition

Cet élément spécifie une plage ou la valeur exacte indiquant la quantité de mémoire virtuelle disponible requise pour le travail. La quantité est exprimée en octets. Cet attribut est facultatif.

Type Le type de cet élément est jsdl:NumericRangeType.

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
<virtualMemory
 xsd:anyAttribute##other>
 jsdl:NumericRangeType
 <xsd:any##other>*
</virtualMemory>
```

élément candidateOperatingSystems

Définition

Cet élément spécifie l'ensemble des caractéristiques de système d'exploitation que doivent satisfaire des hôtes qui peuvent être sélectionnés pour l'exécution du travail. Les combinaisons de caractéristiques trouvent des correspondances OR, autrement dit, au moins l'une d'entre elles doit être satisfaite par la ressource de système d'exploitation contenue dans la ressource cible. Au moins un des systèmes d'exploitation répertoriés doit

être disponible pour que le travail s'exécute. Si aucun des systèmes d'exploitation que vous spécifiez n'est disponible lors de la soumission du travail, le travail attend que l'un d'entre eux devienne disponible. Si aucun système d'exploitation ne devient disponible avant l'expiration du délai, le travail échoue. Cet attribut est facultatif.

Type Le type de cet élément est jsdl:OperatingSystemsRequirementType. Il peut contenir l'élément suivant :

- operatingSystem

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
<candidateOperatingSystems
 <xsd:anyAttribute##other>
 <operatingSystem...>*
 <xsd:any##other/>*
</ candidatOperatingSystems>?
```

élément operatingSystem

Définition

Cet élément spécifie les caractéristiques de système d'exploitation qui sont requises pour exécuter le travail. Cet attribut est obligatoire et peut être spécifié une ou plusieurs fois.

Type Le type de cet élément est OperatingSystemRequirementType.

Attributs

Les attributs suivants sont définis :

type Cet attribut définit le nom du système d'exploitation requis pour l'exécution du travail. Le type de cet attribut est xsd:string. Cet attribut est obligatoire. Les valeurs prises en charge sont les suivantes :

- AIX
- Linux
- Windows 2000
- Windows 2003
- Windows XP
- Windows Vista
- HPUX
- Solaris

version

Spécifiez la version du système d'exploitation. Vous pouvez indiquer la version ou sous-version exacte du système d'exploitation, par exemple 5.2 ou 5.2.3.30, ou vous pouvez indiquer une partie de la version, par exemple 5.2.3. Dans ce cas, l'exigence s'applique à tous les systèmes d'exploitation possédant la version 5.2.3 et ses sous-versions, tels que les groupes de correctifs et les niveaux de maintenance. Le type de cet attribut est xsd:string. Cet attribut est facultatif.

Pseudo-schéma

```
<operatingSystem
 type="xsd:string"
 version="xsd:string"?
 xsd:anyAttribute##other>
 <xsd:any##other>*
</operatingSystem>
```

élément fileSystem

Définition

Cet élément décrit l'ensemble des caractéristiques de système de fichiers qui peuvent être sélectionnées pour l'exécution du travail. Chaque ensemble de caractéristiques spécifie l'emplacement dans lequel le système de fichiers est disponible, la quantité d'espace disque requise et le type du système de fichiers. Le système de fichiers peut se trouver dans le même emplacement que la ressource, par exemple sur un disque local, ou peut être distant, par exemple sur un montage NFS. L'exigence est satisfaite si pour une cible données, toutes les caractéristiques du système de fichiers sont satisfaites. Les combinaisons de caractéristiques sont satisfaites par AND, autrement dit, elles doivent toutes être satisfaites par les ressources de système de fichiers contenues dans la ressource cible. Tous les systèmes de fichiers répertoriés doivent être présents pour que le travail s'exécute. Si l'un des systèmes de fichiers que vous indiquez n'est pas disponible lors de la soumission du travail, le travail attend qu'il devienne disponible. S'il ne devient pas disponible avant l'expiration du délai, le travail échoue. Cet élément est facultatif et peut être spécifié zéro ou plusieurs fois.

Type Le type de cet élément est jsdl:FileSystemRequirementType. Il contient l'élément **diskSpace**.

Attributs

Les attributs suivants sont définis :

type Jeton spécifiant le type de système de fichiers de l'élément fileSystem conteneur. Le type de cet attribut est jsdl:FileSystemTypeEnumeration. Cet attribut est facultatif. Les valeurs prises en charge sont les suivantes :

Unknown
Le système de fichiers n'est pas spécifié.

No Root Directory
Désigne un système de fichiers local qui n'est pas le répertoire racine.

Removable Disk
Désigne un système de fichiers monté sur un disque dur amovible.

Local Disk
Désigne un système de fichiers monté sur un disque local.

Remote Drive
Désigne un système de fichiers monté sur une unité distante.

CD-ROM
Désigne un système de fichiers monté sur un lecteur de CD-ROM.

RAM Disk
Désigne un système de fichiers monté sur un disque RAM.

mountPoint
Expression de variable de chaîne spécifiant le mappage local sur lequel le système de fichiers est disponible pour le travail. Le type de cet attribut est jsdl:StringVariableExpressionType. Pour spécifier le point de montage, vous pouvez utiliser une expression de variables qui peut contenir une ou plusieurs références de variables, telles que ${var}, tout caractère et toute chaîne. Les caractères génériques sont autorisés. Cet attribut est facultatif.

Pseudo-schéma

```
<fileSystem
 type="jsdl:FileSystemTypeEnumeration"?
 mountPoint="jsdl:StringVariableExpressionType"?
 xsd:anyAttribute##other>
 <diskSpace>jsdl:NumericRangeType</diskSpace>?
 <xsd:any##other/>*
</fileSystem>
```

élément diskSpace

Définition
Cet élément indique la quantité d'espace disque sur l'élément de système de fichiers conteneur pour l'exécution du travail. La quantité d'espace disque est exprimée en kilooctets. Cet élément est facultatif et peut être spécifié une fois.

Type Le type de cet élément est jsdl:NumericRangeType.

Attributs
Aucun attribut n'est défini.

Pseudo-schéma

```
<fileSystem
 type="jsdl:FileSystemTypeEnumeration"?
 mountPoint="jsdl:StringVariableExpressionType"?
 xsd:anyAttribute##other>
 <diskSpace>jsdl:NumericRangeType</diskSpace>?
 <xsd:any##other/>*
</fileSystem>
```

élément logicalResource

Définition
Cet élément spécifie une ou plusieurs ressources logiques requises pour exécuter le travail. Les combinaisons de caractéristiques sont satisfaites par AND, autrement dit, elles doivent toutes être remplies par les ressources logiques associées à la ressource cible. Toutes les ressources logiques répertoriées doivent être disponibles pour que le travail s'exécute. Si l'une des ressources logiques que vous spécifiez est indisponible lors de la soumission du travail, le travail attend qu'elle devienne disponible. S'il ne devient pas disponible avant l'expiration du délai, le travail échoue. Cet élément est facultatif et peut être spécifié zéro ou plusieurs fois.

Type Le type de cet élément est LogicalResourceRequirementType.

Attributs
Les attributs suivants sont définis :

name Expression de variable de chaîne spécifiant le nom de la ressource logique demandée. Pour spécifier le nom de ressource logique,

vous pouvez utiliser une expression de variable qui peut contenir un ou plusieurs références de variable, telles que ${var}, tout caractère et toute chaîne. Ce nom doit commencer par un caractère alphabétique et peut comporter les symboles suivants : soulignement (_), moins (-), et point (.). Les espaces, les caractères spéciaux et les caractères accentués ne sont pas pris en charge. Cet attribut est facultatif.

subType
Expression de variable de chaîne spécifiant le type de la ressource logique demandée. Pour spécifier le type de ressource logique, vous pouvez utiliser une expression de variable qui peut contenir un ou plusieurs références de variable, telles que ${var}, tout caractère et toute chaîne. Cet attribut est facultatif.

quantity
Valeur entière spécifiant la quantité de ressource logique requise. La quantité spécifiée est affectée exclusivement au travail en cours d'exécution. Pour spécifier la quantité de ressource, vous pouvez utiliser une expression de variable pouvant contenir une référence de variable, telle que ${var} ou une valeur entière non signée. Cet attribut est facultatif.

Pseudo-schéma

```
<logicalResource
 name="jsdl:StringVariableExpressionType"?
 subType="jsdl:StringVariableExpressionType"?
 quantity="jsdl:UnsignedIntVariableExpressionType"?
 xsd:anyAttribute##other>
 <xsd:any##other/>*
</logicalResource>
```

élément group

Définition
Cet élément spécifie que les ressources requises pour l'exécution du travail doivent appartenir au groupe de ressources spécifié. Les groupes sont satisfaits par AND, c'est-à-dire que la ressource cible doit se trouver dans tous les groupes spécifiés. Cet élément est facultatif et peut être spécifié zéro ou plusieurs fois.

Type Le type de cet élément est jsdl:GroupRequirementType.

Attributs
L'attribut suivant est défini :

name Indique le nom du groupe en une expression de variable de chaîne. Pour spécifier le groupe de ressources, vous pouvez utiliser une expression de variable pouvant contenir une ou plusieurs références de variable, telles que ${var}, tout caractère et toute chaîne. Cet attribut est obligatoire et peut être spécifié une fois.

Pseudo-schéma

```
<group
 name="jsdl:StringVariableExpressionType"
 xsd:anyAttribute##other>
 <xsd:any##other/>*
</group>
```

élément properties

Définition

Cet élément spécifie les propriétés de la ressource requise pour l'exécution du travail. L'exigence est exprimée comme un ensemble de conditions sur les propriétés de ressources combinées avec des opérateurs AND/OR. Celles-ci sont basées sur le modèle de ressource qui décrit les ressources disponibles dans l'environnement comme des instances des types de ressources. Un type de ressource est identifié par un nom unique et décrit les propriétés fournies par chaque instance de ressources. Utilisez cet élément pour spécifier les exigences avancées. Pour plus d'informations sur les types et propriétés de ressource disponibles, voir tableau 112, à la page 767. Cet élément est facultatif et peut être spécifié une fois.

Type Le type de cet élément est jsdl:RequirementCompositorType. Il peut contenir les éléments suivants :

- and
- or
- requirement

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
<properties
 xsd:anyAttribute##other>
 <and .../>?
 <or .../>?
 <requirement .../>?
 <xsd:any##other/>*
</properties>
```

élément and

Définition

Cet élément spécifie une condition AND sur les spécifications d'exigence de conteneur. Cet élément est facultatif et peut être spécifié une fois.

Type Le type de cet élément est jsdl:RequirementCompositorType. Il peut contenir les éléments suivants :

- and
- or
- requirement

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
<and
 xsd:anyAttribute##other>
 <and .../>?
 <or .../>?
 <requirement .../>?
 <xsd:any##other/>*
</and>
```

élément or

Définition

Cet élément spécifie une condition OR sur les spécifications d'exigence de conteneur. Cet élément est facultatif et peut être spécifié une fois.

Type Le type de cet élément est jsdl:RequirementCompositorType. Il peut contenir les éléments suivants :

- and
- or
- requirement

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
<or
 xsd:anyAttribute##other>
 <and .../>?
 <or .../>?
 <requirement .../>?
 <xsd:any##other/>*
</or>
```

élément de condition requise

Définition

Cet élément est une valeur de plage spécifiant une exigence sur les fonctions d'une ressource pour l'exécution du travail. Cet élément est facultatif et peut être spécifié une fois.

Type Le type de cet élément est jsdl:RequirementType.

Attributs

L'attribut suivant est défini :

propertyName

Chaîne spécifiant la propriété de ressource à laquelle s'applique l'exigence. Les propriétés disponibles varient selon les ressources que vous avez sélectionnées dans l'élément resources. Pour plus d'informations sur les types et propriétés de ressources, voir tableau 112, à la page 767. Le type de cet attribut est xsd:NCName. Cet attribut est obligatoire.

Pseudo-schéma

```
<requirement
 propertyName="xsd:NCName"
 xsd:anyAttribute##other>
 jsdl.StringRangeType
 <xsd:any##other/>*
</and>
```

élément allocation

Définition

Cet élément spécifie une exigence d'affectation exclusive sur la propriété donnée d'une ressource. Vous pouvez définir une exigence d'affectation sur les attributs de consommable des ressources. Pour plus d'informations sur les attributs de consommables, voir tableau 112, à la page 767. Le travail peut être exécuté sur la ressource si la valeur requise est disponible. Lorsqu'il s'exécute, le travail utilise exclusivement la valeur de propriété de ressource requise. Lorsque le travail se termine, il libère la valeur de propriété. Cet élément est facultatif et peut être spécifié une fois.

Type Le type de cet élément est jsdl:AllocationRequirementType.

Attributs

L'attribut suivant est défini :

propertyName
Chaîne spécifiant la propriété de ressource à laquelle s'applique l'exigence. Le type de cet attribut est xsd:QName. Cet attribut est obligatoire.

Quantity
Spécifie la quantité de la propriété qui doit être affectée exclusivement au travail. Pour spécifier la quantité de la propriété qui doit être affectée, vous pouvez utiliser une expression de variable qui peut contenir une référence de variable, telle que ${var} ou une double valeur.

Pseudo-schéma

```
<allocation
 propertyName="xsd:QName"
 xsd:anyAttribute##other>
 jsdl:DoubleVariableExpressionType
 <xsd:any##other/>*
</and>
```

élément relationship

Définition
Cet élément spécifie l'exigence que la ressource sélectionnée pour exécuter le travail doit avoir une relation avec d'autres ressources correspondant à certains critères supplémentaires. Une relation est une association directe entre une ressource source et une ressource cible. Cet élément est facultatif et peut être spécifié une fois.

Type Le type de cet élément est jsdl:RelationshipRequirementType.

Attributs
Les attributs suivants sont définis :

type Chaîne spécifiant le type de relation requis. Le type de cet attribut est xsd:NCName. Cet attribut est facultatif.

source Chaîne spécifiant l'ID d'un élément **relatedResources**. Si cet attribut est spécifié, l'exigence de relation indique que la ressource doit avoir au moins une relation dirigée vers elle-même depuis une ou plusieurs ressources satisfaites par l'élément **relatedResources**. Si cet attribut n'est pas spécifié, un attribut cible doit être présent. Le type de cet attribut est xsd:IDREF. Cet attribut est facultatif.

target Chaîne spécifiant l'ID d'un élément **relatedResources**. Si cet attribut est spécifié, l'exigence de relation indique que la ressource doit avoir au moins une relation dirigée vers elle-même depuis une ou plusieurs ressources satisfaites par l'élément **relatedResources**. Si cet attribut n'est pas spécifié, un attribut source doit être présent. Le type de cet attribut est xsd:IDREF. Cet attribut est facultatif.

Pseudo-schéma

```
<relationship
 type="xsd:NCName"
 source="xsd:IDREF"?
 target="xsd:IDREF"?
 xsd:anyAttribute##other>
 <xsd:any##other/>*
</relationship>
```

élément candidateResources

Cet élément est réservé pour une utilisation interne.

Définition

Cet élément spécifie l'ensemble des ressources qui peuvent être sélectionnées pour l'exécution du travail. Si cet élément est spécifié, une ou plusieurs ressources de l'ensemble doivent être choisies pour l'exécution du travail. Les ressources sont identifiées au moyen de leur adresse de référence de noeud final (référence de noeud final d'adressage WS) du service de fabrique de travaux gérant la ressource. Les combinaisons d'exigences trouvent des correspondances OR, autrement dit au moins l'une d'entre elles doit être satisfaite par la ressource contenue dans la ressource cible. Au moins une des ressources répertoriées doit être disponible pour que le travail s'exécute. Si aucun des ressources que vous spécifiez n'est disponible lors de la soumission du travail, le travail attend que l'une d'entre elles devienne disponible. Si aucune ressource ne devient disponible avant l'expiration du délai, le travail échoue. Cet élément est facultatif et peut être spécifié une fois.

Type Le type de cet élément est jsdl:CandidateResourcesRequirementType. Il contient l'élément **endpointReference**.

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
<candidateResources
 xsd:anyAttribute##other>
 <endpointReference>wsa:EndpointReferenceType<endpointReference/>+
 <xsd:any##other>*
</candidateResources>
```

élément endpointReference

Cet élément est réservé pour une utilisation interne.

Définition

Cet élément spécifie la référence de noeud final d'adressage de services Web du service de fabrique de travaux gérant la ressource. Cet élément est obligatoire et peut être spécifié une ou plusieurs fois. Cet élément est réservé pour une utilisation interne.

Type Le type de cet élément est wsa:EndpointReferenceType.

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
<candidateResources
 xsd:anyAttribute##other>
 <endpointReference>wsa:EndpointReferenceType<endpointReference/>+
 <xsd:any##other>*
</candidateResources>
```

élément optimization

Définition

Cet élément spécifie les règles d'optimisation à appliquer au travail. Selon que vous sélectionniez l'objectif ou l'élément ewlm, la règle de sélection de ressources change. Cet élément est facultatif et peut être spécifié une fois.

Si aucune valeur n'est spécifiée, le règle d'équilibrage de charge par défaut consistant à égaliser le nombre de travaux qui s'exécutent sur chaque ressource est appliquée.

Type Le type de cet élément est jsdl:OptimizationType. Il ne peut contenir que les éléments suivants :

- objective
- ewlm

Attributs

L'attribut suivant est défini :

name Indique le nom de la règle d'optimisation. Les valeurs prises en charge sont les suivantes :

JPT_JSDLOptimizationPolicyType
Si vous utilisez cette option, vous devez spécifier l'élément **objective**. A l'aide de l'élément **objective**, vous pouvez spécifier que les propriétés de ressources soient maximisées ou réduites. Lorsque vous définissez l'élément **objective**, Dynamic Workload Broker exécute le travail sur la ressource qui correspond à l'exigence d'optimisation. Il s'agit de la valeur par défaut. Voir «Ressources de la définition de travail», à la page 766 pour plus d'informations.

JPT_BestResource
Utilisez cette option afin que le pool de ressources du travail soit constitué uniquement des meilleures ressources parmi le pool de ressources qui répondent à la règle spécifiée. Si vous utilisez cette option, vous devez spécifier l'élément **objective**. A l'aide de l'élément **objective**, vous pouvez spécifier que les propriétés de ressources soient maximisées ou réduites. Lorsque vous définissez l'élément **objective**, Dynamic Workload Broker exécute le travail sur la ressource qui correspond à l'exigence d'optimisation. Il s'agit de la valeur par défaut. Voir «Ressources de la définition de travail», à la page 766 pour plus d'informations.

JPT_EWLM
Si vous utilisez cette option, l'élément **ewlm** est automatiquement inséré.

Pseudo-schéma

```
<optimization
 name="xsd:NCName"
 xsd:anyAttribute##other>
 <objective .../> |
 <ewlm .../>
 <xsd:any##other>*
</optimization>
```

élément objective

Définition

Cet élément spécifie l'objectif à atteindre lorsque vous effectuez l'optimisation d'un travail. Par exemple, si vous sélectionnez l'utilisation d'unité centrale **resourcePropertyName** pour le système informatique **resourceType** avec une réduction **propertyObjective**, Dynamic Workload

Broker tentera d'exécuter le travail sur la ressource où l'utilisation d'unité centrale est la plus faible. Cet élément et l'élément **ewlm** s'excluent mutuellement.

Type Le type de cet élément est PropertyObjectiveType.

Attributs

Les attributs suivants sont définis :

propertyObjective

Chaîne spécifiant le type d'objet. Cet attribut est obligatoire. Les valeurs prises en charge sont les suivantes :

minimize

Les ressources sont affectées au travail avec l'objectif de réduire la valeur de la propriété de type de ressource spécifiée. Par exemple, vous pouvez choisir cet objectif pour attribuer le travail à la ressource dans laquelle l'utilisation de l'unité centrale est la plus faible.

maximize

Les ressources sont affectées au travail avec l'objectif de maximiser la valeur de la propriété de type de ressource spécifiée. Par exemple, vous pouvez choisir cet objectif pour affecter le travail à la ressource dans laquelle la vitesse de traitement est la plus élevée.

minimize.utilization

Les ressources sont affectées au travail avec l'objectif de réduire l'utilisation de la propriété de type de ressource spécifiée. Cet attribut est uniquement disponible pour les propriétés de consommables. Pour une liste de toutes les propriétés de consommable, voir tableau 112, à la page 767. Si vous choisissez de minimiser l'utilisation de la consommation de la propriété, le travail est affecté à une ressource sur laquelle une quantité moindre de la propriété est utilisée.

maximize.utilization

Les ressources sont affectées au travail avec l'objectif de maximiser l'utilisation de la propriété du type de ressource spécifiée. Cet attribut est uniquement disponible pour les propriétés de consommables. Pour une liste de toutes les propriétés de consommable, voir tableau 112, à la page 767. Par exemple, vous aurez peut-être besoin d'effectuer un test de charge sur un poste de travail en créant des travaux dans lesquels la propriété de ressource d'utilisation d'unité centrale est définie par Maximize Utilization. Dans ce cas, tous les travaux présentant ce paramètre seraient affectés au poste de travail sur lequel l'utilisation de l'unité centrale est supérieure, créant une boucle.

resourceType

Chaîne spécifiant le type de la ressource à laquelle la règle s'applique. Si cet élément n'est pas spécifié, le système prend en compte le type de ressource ComputerSystem. Le type de cet attribut est xsd:QName. Cet attribut est facultatif.

resourcePropertyName
Chaîne spécifiant la propriété de ressource à laquelle la règle s'applique. Le type de cet attribut est xsd:QName. Cet attribut est obligatoire.

Remarque : Lorsque vous indiquez une exigence d'optimisation sur une propriété d'un type de ressource, vous devez avoir défini préalablement une exigence sur ce type de ressource. Par exemple, si vous souhaitez optimiser la mémoire physique totale sur un système d'exploitation, vous devez préalablement définir une exigence sur le type de ressource de système d'exploitation. Cette procédure ne s'applique pas au type de ressource Computer System, car celui-ci est le type de ressource par défaut.

Pseudo-schéma

```
<objective
 propertyObjective="minimize" | "maximize"
 resourceType="xsd:QName"?
 resourcePropertyName="xsd:QName"
 xsd:anyAttribute##other>
 <xsd:any##other>*
</objective>
```

élément ewlm

Définition
Cet élément spécifie l'optimisation en fonction du calcul du poids de ressource d'Enterprise Workload Manager. Dynamic Workload Broker exécutera le travail sur les meilleures ressources disponibles telles qu'elle sont indiquées par Enterprise Workload Manager. Cet élément est facultatif et peut être spécifié une fois. Cet élément et l'élément **objective** s'excluent mutuellement.

Type Le type de cet élément est jsdl:PropertyObjectiveType.

Attributs
Aucun attribut n'est défini.

Pseudo-schéma

```
<ewlm>
 xsd:anyAttribute##other>
 <xsd:any##other>*
</ewlm>
```

élément scheduling

Définition
Cet élément spécifie les paramètres de planification à appliquer lors de l'exécution du travail. Cet élément est facultatif et peut être spécifié une fois.

Dans Dynamic Workload Broker, cet élément correspond à la page Planification. Pour plus d'informations sur la page Planification, voir la documentation de l'aide en ligne.

Type Le type de cet élément est jsdl:SchedulingType. Il peut contenir les éléments suivants :

- maximumResourceWaitingTime
- estimatedDuration
- priority

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
<scheduling
 xsd:anyAttribute##other>
 <maximumResourceWaitingTime>xsd:duration<maximumResourceWaitingTime>?
 <estimatedDuration>xsd:duration<estimatedDuration>?
 <priority>xsd:unsignedint<priority>?
 <xsd:any##other>*
</objective>
```

élément maximumResourceWaitingTime

Définition

Cet élément spécifie combien de temps le serveur Dynamic Workload Broker doit attendre depuis la soumission du travail avant de décider qu'aucune ressource ne correspond aux exigences. Cet élément est facultatif et peut être spécifié une fois.

Type Le type de cet élément est xsd:duration.

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
<maximumResourceWaitingTime>?
 xsd:anyAttribute##other>
 <xsd:duration>*
<maximumResourceWaitingTime>
```

élément estimatedDuration

Définition

Cet élément spécifie la durée estimée d'exécution du travail. Celle-ci peut être utilisée par le serveur Dynamic Workload Broker pour planifier l'affectation des ressources. Cet élément est facultatif et peut être spécifié une fois.

Type Le type de cet élément est xsd:duration.

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
<estimatedDuration>?
 <xsd:anyAttribute##other>
 <xsd:duration>*
<estimatedDuration>
```

élément priority

Définition

Cet élément spécifie la priorité du travail sous forme d'un entier compris entre 0 et 100. Plus les valeurs sont hautes, plus la priorité est haute. Cet élément est facultatif et peut être spécifié une fois.

Type Le type de cet élément est xsd:PriorityType. Il peut contenir les éléments suivants :

- maximumResourceWaitingTime
- estimatedDuration
- priority

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
<scheduling
 xsd:anyAttribute##other>
 <maximumResourceWaitingTime>xsd:duration<maximumResourceWaitingTime>?
 <estimatedDuration>xsd:duration<estimatedDuration>?
 <priority>xsd:unsignedint<priority>?
 <xsd:any##other>*
</objective>
```

recoveryActions

Définition

Cet élément décrit la liste des actions de reprise que Dynamic Workload Broker doit effectuer si l'intervalle de temps spécifié dans l'élément maximumResourceWaitingTime arrive à expiration et qu'aucune ressource correspondant aux exigences n'a été trouvée. Cet élément est facultatif et peut être spécifié une fois. Les actions de reprise définies dans cet élément sont effectuées en démarrant un flux de travail Tivoli Provisioning Manager.

Type Le type de cet élément est jsdl:RecoveryActionList. Il peut contenir l'élément action.

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
<recoveryActions
 xsd:anyAttribute##other>
 <action...>+
 <xsd:any##other/>*
</recoveryActions >
```

action

Définition

Cet élément spécifie les actions de reprise à effectuer. Dynamic Workload Broker exécute les actions répertoriées de façon séquentielle en fonction de l'ordre dans lequel elles ont été spécifiées. Toute action de reprise suivante est exécutée uniquement si l'action précédente s'est terminée avec succès. Si l'élément **recoveryActions** est spécifié, au moins un élément **action** doit être spécifié.

Type Le type de cet élément est jsdl:RecoveryActionType.

Attributs

Les attributs suivants sont définis :

name Spécifie le nom de l'action de reprise à effectuer. Cet attribut est facultatif.

additionalTimeOnCompletion

Spécifie l'intervalle de temps pendant lequel Dynamic Workload Broker doit attendre pour que l'action de reprise devienne effective après achèvement. Si cet attribut est spécifié pour une action de reprise, l'action de reprise suivante est effectuée seulement après expiration de l'intervalle de temps spécifié. Si la ressource requise devient disponible avant que l'intervalle expire, Dynamic Workload Broker peut décider d'exécuter le travail avant que l'action se termine.

maximumExecutionTime
Spécifie la période de temps attendue pendant laquelle Dynamic Workload Broker attend que l'action de reprise se termine. Si l'action de reprise n'est pas terminée à l'expiration de ce délai, la procédure de reprise échoue et la séquence de reprise est arrêtée.

Pseudo-schéma

```
< action>
 name="xsd:NCName"
 additionalTimeOnCompletion ="xsd:duration"?
 maximumExecutionTime ="xsd:duration"?
 xsd:anyAttribute##other>
 <xsd:any##other/>*
</ action >
```

élément tpmaction

Définition
Cet élément spécifie les paramètres requis pour exécuter une action de reprise Tivoli Provisioning Manager. Cet attribut est facultatif et peut être spécifié une fois.

Type Le type de cet attribut est jsdltpm:TPMActionType. Il peut contenir les éléments suivants :

- parameters
- credential
- tpmaddress
- workFlow

Attributs
Aucun attribut n'est défini.

Pseudo-schéma

```
<tpmaction
 <parameters... />?
 <credential.../>?
 <tpmaddress.../>?
 workFlow="jsdl:StringVariableExpressionType"
</tpmaction>
```

élément parameters

Définition
Cet élément indique les arguments à utiliser lors de l'exécution du flux de travaux Tivoli Provisioning Manager. Cet élément est facultatif et peut être spécifié une fois.

Type Le type de cet attribut est jsdltpm:ParametersType.

Attributs
Les attributs suivants sont pris en charge :

name Indique le nom de l'attribut à utiliser lors de l'exécution du flux de travail Tivoli Provisioning Manager. Cet attribut est obligatoire et peut être spécifié une ou plusieurs fois.

Pseudo-schéma

```
<parameters
 <parameter... />+
</parameters>
```

élément credential

Définition

Cet élément spécifie les données d'identification requises pour l'exécution du flux de travaux Tivoli Provisioning Manager. Cet élément est facultatif et peut être spécifié une fois.

Type Le type de cet élément est jsdl:CredentialType. Il peut contenir les éléments suivants :

- userName
- password

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
<credential
 <userName> jsdl:NotEmptyStringVariableExpressionType</userName>
 <password> jsdl:StringVariableExpressionType </password>
</credential>
```

élément userName

Définition

Cet élément indique un nom d'utilisateur défini sur le système cible, qui est utilisé pour exécuter le flux de travail Tivoli Provisioning Manager. Cet élément est obligatoire si vous utilisez l'élément credential et peut être spécifié une fois.

Type Le type de cet élément est jsdl:NotEmptyStringVariableExpressionType.

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
<userName>
 xsd:anyAttribute##other>
 <jsdl:NotEmptyStringVariableExpressionType*
<userName>
```

élément password

Définition

Cet élément spécifie le mot de passe de l'utilisateur spécifié utilisé pour exécuter le flux de travaux Tivoli Provisioning Manager sur le système cible. Cet élément est obligatoire si vous utilisez l'élément credential et peut être spécifié une fois.

Type Le type de cet élément est jsdl:StringVariableExpressionType.

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
<password>?
 xsd:anyAttribute##other>
 <jsdl:StringVariableExpressionType
<password>
```

élément tpmaddress

Définition

Cet élément spécifie l'adresse Tivoli Provisioning Manager qui doit être

utilisée pour appeler le service Web Tivoli Provisioning Manager nécessaire à l'exécution du flux de travail. Cet élément est facultatif et peut être spécifié une fois.

Type Le type de cet élément est jsdltpm:TPMAddressType.

Attributs

Les attributs suivants sont définis :

hôte Spécifie le nom d'hôte du serveur Tivoli Provisioning Manager à utiliser lors de l'exécution de l'action de reprise.

port Spécifie le numéro de port du server Tivoli Provisioning Manager à utiliser lors de l'exécution de l'action de reprise.

Pseudo-schéma

```
< tpmaddress
 <host... />
 <port... />
</ tpmaddress >
```

élément workflow

Définition

Cet élément indique le nom du flux de travaux Tivoli Provisioning Manager à exécuter. Pour spécifier le nom de flux de travail, vous pouvez utiliser une expression de variable qui peut contenir une ou plusieurs références de variable, telles que ${var}, tout caractère et toute chaîne. Cet élément est obligatoire.

Type Le type de cet élément est jsdl:StringVariableExpressionType.

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
<workflow>?
 xsd:anyAttribute##other>
 <jsdl:StringVariableExpressionType
<workflow>
```

élément executable

Définition

Cet élément spécifie les paramètres pour l'exécution d'une commande système native, c'est-à-dire des fichiers exécutables et des scripts. Vous pouvez également imbriquer des scripts dans cet élément. Cet élément est obligatoire.

Remarque : Les restrictions suivantes s'appliquent :

- Sur les systèmes Windows, vous pouvez exécuter des scripts contenant des commandes de traitement par lots. Les formats de script pris en charge sont :
  - .cmd
  - .bat
- Sur les systèmes UNIX et Linux, seuls les scripts de shell sont pris en charge. Au début du script de shell, vous devez indiquer l'interpréteur de commandes.
- Sur les systèmes UNIX et Linux, les commandes contenues dans les scripts doivent s'exécuter en avant-plan. Cela signifie que vous ne pouvez pas utiliser le paramètre "&" en association avec la commande.

- Sur aucune des plateformes prises en charge, vous ne pouvez inclure dans les travaux des commandes démarrant un programme au moyen d'une interface graphique.

Type Le type de cet élément est jsdle:ExecutableType. Il peut contenir les éléments suivants :

- script
- arguments
- environment
- credential

Attributs

Les attributs suivants sont définis :

path Expression de variable de chaîne spécifiant le nom de chemin du fichier exécutable à exécuter. Si l'élément **script** n'est pas présent, l'attribut **path** doit être spécifié. Si l'élément **script** est présent, l'attribut **path** ne peut pas être spécifié. Pour indiquer le chemin, vous pouvez utiliser une expression de variable qui peut contenir une ou plusieurs références de variable, telles que ${var}, tout caractère et toute chaîne.

Vous devez indiquer l'extension de fichier. Si vous souhaitez exécuter un fichier exécutable sans spécifier son extension, vous pouvez spécifier le nom du fichier exécutable dans l'élément **script**, de sorte que le fichier soit exécuté dans le shell.

input Expression de variable de chaîne spécifiant l'entrée standard de la commande. Cet attribut est le nom de chemin absolu ou le nom de chemin relatif du répertoire de travail. Pour indiquer le chemin, vous pouvez utiliser une expression de variable qui peut contenir une ou plusieurs références de variable, telles que ${var}, tout caractère et toute chaîne. Cet attribut est facultatif.

output

Expression de variable de chaîne spécifiant la sortie standard de la commande. Cet attribut est le nom de chemin absolu ou le nom de chemin relatif du répertoire de travail. Pour indiquer le chemin, vous pouvez utiliser une expression de variable qui peut contenir une ou plusieurs références de variable, telles que ${var}, tout caractère et toute chaîne. Cet attribut est facultatif.

error Expression de variable de chaîne spécifiant l'erreur standard pour la commande. Cet attribut est le nom de chemin absolu ou le nom de chemin relatif du répertoire de travail. Pour indiquer le chemin, vous pouvez utiliser une expression de variable qui peut contenir une ou plusieurs références de variable, telles que ${var}, tout caractère et toute chaîne. Cet attribut est facultatif.

workingDirectory

Expression de variable de chaîne spécifiant le répertoire de travail requis par le travail à exécuter. Pour spécifier le répertoire, vous pouvez utiliser une expression de variable pouvant contenir une ou plusieurs références de variable, telles que ${var}, tout caractère et toute chaîne. Cet attribut est facultatif. Si vous ne spécifiez pas cet attribut, le travail s'exécute dans les répertoires suivants, selon le système d'exploitation :

Sous UNIX

Les cas suivants s'appliquent :

- Le travail s'exécute dans le répertoire $HOME_DIRECTORY de l'utilisateur qui soumet le travail, s'il existe.
- Si le répertoire n'existe pas, le travail s'exécute sur /root, si l'utilisateur qui soumet le travail possède les droits requis.
- Si l'utilisateur ne possède pas les droits requis, le travail s'exécute dans le répertoire de travail de l'agent Tivoli Workload Scheduler.

Sous Windows

Le travail s'exécute dans le répertoire de travail de l'agent Tivoli Workload Scheduler.

script Spécifie le code de script à exécuter. Pour spécifier des caractères spéciaux requis par les langages de script, le contenu de l'élément script peut être indiqué à l'aide d'une section CDATA.

Pseudo-schéma

```
<executable
 path="jsdl:StringVariableExpressionType"
 input="jsdl:StringVariableExpressionType"?
 output="jsdl:StringVariableExpressionType"?
 error="jsdl:StringVariableExpressionType"?
 workingDirectory="jsdl:StringVariableExpressionType"?
 xsd:anyAttribute##other>
 <script ... />?
 <arguments .../>?
 <environment .../>?
 <credential .../>?
 <xsd:any##other>*
</executable>
```

élément script

Définition

Cet élément spécifie le code de script à exécuter. Cet élément est facultatif et peut être spécifié une fois.

Type Le type de cet élément est xsd:string.

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
<script>?
 xsd:anyAttribute##other>
 <xsd:string
<script>
```

élément arguments

Définition

Cet élément spécifie la liste des arguments sous forme d'expressions de variable qui sont concaténées pour produire la chaîne d'argument à transmettre à la commande. Cet élément est facultatif et peut être spécifié une fois.

Type Le type de cet élément est jsdle:ArgumentsType. Il peut contenir l'élément suivant :

- value

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
<arguments
 xsd:anyAttribute##other>
 <value>jsdl:StringVariableExpressionType</value>+
 <xsd:any##other>*
</arguments>
```

élément value

Définition

Cet élément spécifie la valeur de l'élément **arguments**. Pour spécifier la valeur, vous pouvez utiliser une expression de variable pouvant contenir une ou plusieurs références de variable, telles que ${var}, tout caractère et toute chaîne. Cet élément est obligatoire et peut être spécifié une ou plusieurs fois.

Type Le type de cet élément est jsdl:StringVariableExpressionType.

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
<arguments
 xsd:anyAttribute##other>
 <value>jsdl:StringVariableExpressionType</value>+
 <xsd:any##other>*
</arguments>
```

Remarque : Si vous avez besoin de spécifier qu'une valeur consiste en un espace blanc, vous devez l'entourer de guillemets.

élément environment

Définition

Cet élément indique une expression de variable de chaîne de variables d'environnement qui seront définies pour le travail dans l'environnement en cours d'exécution. Cet élément est facultatif et peut être spécifié une fois.

Type Le type de cet élément est jsdl:EnvironmentType. Il peut contenir l'élément suivant :

- variable

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
environnement <
 xsd:anyAttribute##other>
 <variable name="xsd:string">jsdl:StringVariableExpressionType</variable>+
 <xsd:any##other>*
</environment>
```

Remarque : Si vous avez besoin de spécifier qu'une valeur consiste en un espace blanc, vous devez l'entourer de guillemets.

élément variable

Définition

Cet élément indique une expression de variable de chaîne de variables d'environnement qui seront définies pour le travail dans l'environnement en cours d'exécution. Cet élément est facultatif et peut être spécifié une fois.

Type Le type de cet élément est jsdl:StringVariableExpressionType.

Attributs

Les attributs suivants sont définis :

name Spécifie le nom de la variable.

value Spécifie la valeur de la variable. Pour spécifier la valeur de la variable, vous pouvez utiliser une expression de variable pouvant contenir une ou plusieurs références de variable, telles que ${var}, tout caractère et toute chaîne.

élément credential

Définition

Cet élément spécifie les données d'identification pour l'exécution de la commande. Incluez cet élément lorsque vous voulez spécifier un nom d'utilisateur ou de groupe avec lequel l'exécutable ou le script s'exécute sur le système cible, qui est différent du nom d'utilisateur ou de groupe avec lequel l'agent de charge de travail s'exécute. Cet élément est facultatif et peut être spécifié une fois.

Type Le type de cet élément est jsdle:CredentialType. Il peut contenir les éléments suivants :

- userName
- groupName
- password

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
<credential
 xsd:anyAttribute##other>
 <userName> jsdl:StringVariableExpressionType </userName>
 <groupName> jsdl:StringVariableExpressionType </groupName>
 <password> jsdl:StringVariableExpressionType </password>
 <xsd:any##other>*
</credential>
```

élément userName

Définition

Expression de variable de chaîne qui spécifie le nom d'utilisateur d'un utilisateur défini sur le système cible. La commande s'exécute à l'aide de ce nom d'utilisateur. Cet élément est obligatoire si vous utilisez l'élément credential et peut être spécifié une fois. Il peut être un ID utilisateur UNIX ou Windows. Pour spécifier le nom d'utilisateur, vous pouvez utiliser une expression de variable pouvant contenir une ou plusieurs références telles que ${var}, que vous pouvez associer facultativement à tout caractère ou à une chaîne simple. Si l'application s'exécute sur un système Windows en tant qu'utilisateur de domaine Windows, spécifiez le nom d'utilisateur comme suit :

```
nom_domaine\nom_utilisateur
```

Si l'application s'exécute en tant qu'utilisateur local, vous pouvez utiliser le format suivant :

```
nom_utilisateur
```

Type Le type de cet élément est jsdl:NotEmptyStringVariableExpressionType.

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
<credential
 xsd:anyAttribute##other>
 <userName> jsdl:StringVariableExpressionType </userName>
 <groupName> jsdl:StringVariableExpressionType </groupName>
 <password> jsdl:StringVariableExpressionType </password>
 <xsd:any##other>*
</credential>
```

élément groupName

Définition

Expression de variable de chaîne qui spécifie le nom du groupe auquel appartient l'utilisateur, qui est défini sur le système cible sur lequel la commande s'exécute. Cet élément est facultatif et peut être spécifié une fois. Cet élément est ignoré sur les systèmes cible Windows. Pour spécifier le nom de groupe, vous pouvez utiliser une expression de variable qui peut contenir une ou plusieurs références de variable, telle que ${var}, tout caractère et toute chaîne.

Type Le type de cet élément est jsdl:StringVariableExpressionType.

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
<credential
 xsd:anyAttribute##other>
 <userName> jsdl:StringVariableExpressionType </userName>
 <groupName> jsdl:StringVariableExpressionType </groupName>
 <password> jsdl:StringVariableExpressionType </password>
 <xsd:any##other>*
</credential>
```

élément password

Définition

Expression de variable de chaîne qui définit le mot de passe du nom d'utilisateur spécifié utilisé pour exécuter la commande sur le système cible. Cet élément est facultatif et peut être spécifié une fois. Cet élément est ignoré sur les systèmes cible UNIX. Pour spécifier le mot de passe, vous pouvez utiliser une expression de variable pouvant contenir une ou plusieurs références de variable, telles que ${var}, tout caractère et toute chaîne.

Type Le type de cet élément est jsdl:StringVariableExpressionType.

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
<credential
 xsd:anyAttribute##other>
 <userName> jsdl:StringVariableExpressionType </userName>
 <groupName> jsdl:StringVariableExpressionType </groupName>
 <password> jsdl:StringVariableExpressionType </password>
 <xsd:any##other>*
</credential>
```

élément j2ee

Définition

Cet élément spécifie les informations d'application J2EE nécessaires au travail. Cet élément est facultatif et peut être spécifié une fois. Les opérations J2EE que vous pouvez effectuer varient selon le type de planificateur (direct ou indirect) que vous sélectionnez et selon que vous activez ou non WebSphere Application Server ou la sécurité J2EE.

Type Le type de cet élément est jsdlj:J2EEType. Il peut contenir les éléments suivants :

- invoker
- jsm
- ejb
- credential

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
<j2ee>?
 xsd:anyAttribute##other>
 <jsdlj:J2EEType
<j2ee>
```

élément invoker

Définition

Cet élément spécifie si un auteur d'appel direct ou indirect doit être utilisé pour l'application J2EE. Cet élément est obligatoire et peut être spécifié une fois. Sélectionner un auteur d'appel direct signifie que l'agent Tivoli Workload Scheduler achemine immédiatement le travail aux composants d'instance WebSphere Application Server (EJB ou JMS). Sélectionner un auteur d'appel indirect signifie que l'agent Tivoli Workload Scheduler optimise une infrastructure de planification WebSphere existante déjà configurée sur le WebSphere Application Server cible.

Type Le type de cet élément est jsdlj:InvokerType.

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
<invoker>?
 xsd:anyAttribute##other>
 <jsdlj:InvokerType
<invoker>
```

élément jms

Définition

Cet élément spécifie la file d'attente Java Message System (JMS) cible et le

message à envoyer. Cet élément est facultatif et peut être spécifié une fois. Cet élément et l'élément **ejb** s'excluent mutuellement.

Type Le type de cet élément est jsdlj:JMSActionType.

Attributs
Aucun attribut n'est défini.

Pseudo-schéma
```
<jms>?
 xsd:anyAttribute##other>
 <jsdlj:JMSActionType
<jms>
```

élément ejb

Définition
Cet élément spécifie les caractéristiques du répertoire de base JNDI de l'EJB à appeler. Cet élément et l'élément **jms** s'excluent mutuellement. L'EJB doit être déjà installé dans le système WAS cible et doit implémenter l'interface TaskHandler.

Type Le type de cet élément est jsdlj:EJBActionType. Il peut contenir les éléments suivants :

- jndiHome
- credential

Attributs
Aucun attribut n'est défini.

Pseudo-schéma
```
<ejb>?
 xsd:anyAttribute##other>
 <jsdlj:EJBActionType
<ejb>
```

élément jndiHome

Définition
Cet élément spécifie le répertoire de base de l'interface de programmation Java Naming and Directory Interface (JNDI). Cet élément est obligatoire et peut être spécifié une fois.

Type Le type de cet élément est jsdl:StringVariableExpressionType.

Attributs
Aucun attribut n'est défini.

Pseudo-schéma
```
<jndiHome>?
 xsd:anyAttribute##other>
 <jsdl:StringVariableExpressionType
<jndiHome>
```

élément ejb

Définition
Cet élément spécifie les caractéristiques de l'action JMS.

Type Le type de cet élément est jsdlj:JMSActionType. Il peut contenir les éléments suivants :

- connFactory
- destination

- message
- credential

Attributs
Aucun attribut n'est défini.

Pseudo-schéma
```
<jms>?
 xsd:anyAttribute##other>
 <jsdl:JMSActionType
<jms>
```

élément connFactory

Définition
Cet élément spécifie un objet administré utilisé par un client pour créer une connexion au fournisseur JMS. Cet élément est obligatoire et peut être spécifié une fois.

Type Le type de cet élément est jsdl:StringVariableExpressionType.

Attributs
Aucun attribut n'est défini.

Pseudo-schéma
```
<connFactory>
 xsd:anyAttribute##other>
 <jsdl:StringVariableExpressionType
<connFactory>
```

élément destination

Définition
Cet élément spécifie un objet administré qui encapsule l'identité d'une destination de message, qui est l'emplacement où les messages sont fournis et consommés. Cet élément est obligatoire et peut être spécifié une fois.

Multiplicité

Type Le type de cet élément est jsdl:StringVariableExpressionType.

Attributs
Aucun attribut n'est défini.

Pseudo-schéma

élément message

Définition
Cet élément spécifie un objet qui est envoyé d'une application à une autre. Cet élément est obligatoire et peut être spécifié une fois.

Type Le type de cet élément est jsdl:StringVariableExpressionType.

Attributs
Aucun attribut n'est défini.

Pseudo-schéma
```
<message>
 xsd:anyAttribute##other>
 <jsdl:StringVariableExpressionType
<message>
```

élément credential

Définition

Cet élément spécifie les données d'identification requises pour l'exécution de l'application J2EE. Incluez cet élément lorsque vous voulez spécifier un nom d'utilisateur avec lequel l'application s'exécute sur le système cible, qui est différent du nom d'utilisateur avec lequel l'agent de charge de travail s'exécute. Cet élément est facultatif et peut être spécifié une fois.

Type Le type de cet élément est jsdl:CredentialType. Il peut contenir les éléments suivants :

- userName
- password
- JAASAuthenticationAlias

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
<credential>?
 xsd:anyAttribute##other>
 <jsdl:CredentialType
<credential>
```

élément userName

Définition

Cet élément spécifie le nom d'utilisateur d'un utilisateur défini sur le système cible. L'application J2EE s'exécute à l'aide de ce nom d'utilisateur. Cet élément est obligatoire si vous utilisez l'élément credential et peut être spécifié une fois. Pour indiquer le nom d'utilisateur, vous pouvez utiliser une expression de variable qui contient une ou plusieurs références telles que ${var}, que vous pouvez associer facultativement à tout caractère ou à une chaîne simple. Si vous choisissez un auteur d'appel indirect, utilisez cet élément pour spécifier le nom d'utilisateur requis pour vous connecter au planificateur WebSphere Application Server.

Type Le type de cet élément est jsdl:NotEmptyStringVariableExpressionType.

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
<userName>
 xsd:anyAttribute##other>
 <jsdl:NotEmptyStringVariableExpressionType
<userName>
```

élément password

Définition

Cet élément spécifie le mot de passe du nom d'utilisateur spécifié utilisé pour exécuter l'application J2EE sur le système cible. Cet élément est facultatif et peut être spécifié une fois. Pour spécifier le mot de passe, vous pouvez utiliser une expression de variable pouvant contenir une ou plusieurs références de variable, telles que ${var}, tout caractère et toute chaîne. Si vous choisissez un auteur d'appel indirect, utilisez cet élément pour spécifier le mot de passe requis pour vous connecter au planificateur WebSphere Application Server.

Type Le type de cet élément est jsdl:StringVariableExpressionType.

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
<password>
 xsd:anyAttribute##other>
 <jsdl:StringVariableExpressionType
<password>
```

élément JAASAuthenticationAlias

Définition

Cet élément spécifie l'alias d'authentification JAAS. Cet élément est facultatif et peut être spécifié une fois. Il est obligatoire uniquement lorsque vous utilisez un auteur d'appel indirect. Pour spécifier l'alias, vous pouvez utiliser une expression de variable pouvant contenir une ou plusieurs références de variable telles que ${var}, que vous pouvez associer facultativement à tout caractère ou à une chaîne simple.

Type Le type de cet élément est jsdl:StringVariableExpressionType.

Attributs

Aucun attribut n'est défini.

Pseudo-schéma

```
<JAASAuthenticationAlias>
 xsd:anyAttribute##other>
 <jsdl:NotEmptyStringVariableExpressionType
<JAASAuthenticationAlias>
```

Ressources de la définition de travail

Cette rubrique présente comment les ressources et leurs propriétés sont utilisées dans la définition de travail afin d'identifier les cibles possibles, de réserver les affectations des ressources consommables et d'optimiser l'équilibrage de charge entre les ressources disponibles.

Il est essentiel de comprendre les ressources physiques et logiques ainsi que leurs propriétés pour créer une définition de travail qui cible précisément les ressources adaptées pour l'exécution du travail, détermine l'exigence d'affectation de ressources et contribue à l'équilibrage de charge entre les ressources disponibles. Chaque ressource possède une ou plusieurs propriétés qui lui sont associées. Les propriétés peuvent avoir les caractéristiques suivantes :

Elles sont consommables

Les propriétés des ressources qui sont consommables possèdent une quantité finie qui leur est associée et peut être consommée par les travaux affectés à la ressource. Par exemple, un système informatique possède un nombre fini de processeurs.

Elles peuvent être optimisées

Certaines propriétés peuvent être utilisées pour définir des objectifs d'optimisation, qui déterminent comment la charge doit être équilibrée lorsque les travaux sont affectés à un groupe de ressources. Par exemple, vous pouvez choisir d'affecter un travail aux ressources correspondantes qui ont l'utilisation d'unité centrale la plus faible.

Elles prennent en charge les caractères génériques

Certaines propriétés peuvent être spécifiées dans la définition de travail à l'aide de caractères génériques. Par exemple, une exigence pour une série

spécifique de modèles d'ordinateurs peut être définie en spécifiant le modèle à l'aide de caractères génériques.

Le tableau 112 présente les différents types de ressources qui peuvent être incluses dans une définition de travail ainsi que leurs propriétés disponibles.

Tableau 112. Types de ressource et propriétés

Type de ressource	**Propriétés disponibles**	**Elles sont consommables**	**Elles peuvent être optimisées**	**Elles prennent en charge les caractères génériques**
Système informatique	Utilisation d'unité centrale	Non	Yes	Non
	HostName	Non	Non	Yes
	Fabricant	Non	Non	Oui
	Modèle	Non	Non	Yes
	Nombre de processeurs	Yes	Yes	Non
	Vitesse de traitement	Non	Yes	Non
	Type de processeur	Non	Non	Non
LogicalResource	Nom d'affichage	Non	Non	Yes
	SubType	Non	Non	Yes
	Quantity	Yes	Yes	Non
OperatingSystem	Nom d'affichage	Non	Non	Yes
	Mémoire physique disponible	Non	Yes	Non
	Espace de permutation disponible	Non	Yes	Non
	Mémoire virtuelle disponible	Non	Yes	Non
	Type de système d'exploitation	Non	Non	Non
	OperatingSystem Version	Non	Non	Non
	Mémoire physique totale	Yes	Yes	Non
	Espace de permutation total	Yes	Yes	Non
	Mémoire virtuelle totale	Yes	Yes	Non
FileSystem	Nom d'affichage	Non	Non	Yes
	Racine du système de fichiers	Non	Non	Yes
	Type de système de fichiers	Non	Non	Non
	Mémoire disponible	Non	Yes	Non
	Mémoire totale	Yes	Yes	Non

Tableau 112. Types de ressource et propriétés (suite)

Type de ressource	Propriétés disponibles	Elles sont consommables	Elles peuvent être optimisées	Elles prennent en charge les caractères génériques
Système réseau	Adresse réseau	Non	Non	Non
	Système réseau HostName	Non	Non	Yes

Annexe C. Aide-mémoire pour les commandes

Cette annexe est composée de quatre sections :

Gestion du plan

Cette section décrit les opérations que vous pouvez effectuer sur le plan à l'aide du script **JnextPlan** et de la ligne de commande **planman** :

Tableau 113. Commandes utilisées sur le plan

Syntaxe de la commande ou du script	Action effectuée
JnextPlan [**-from** *mm/jj/[aa]aa[hh[:]mm[**tz** | **timezoned** nom_fh]]*] {**-to** *mm/jj/[aa]aa[hh[:]mm[**tz** | **fuseau horaire** nom_fh]]* | **-for** *[h]hh[:]mm* [**-days** *n*] | **-days** *n*}	Crée ou étend le plan de production.
planman [*paramètres_connexion*] **crt** [**-from** *mm/jj/[aa]aa* [*hh[:]mm* [**tz** | **fuseau horaire** *nom_fh*]]] {**-to** *mm/jj/[aa]aa[hh[:]mm[**tz** | **fuseau horaire** nom_fh]]* | **-for** *[h]hh[:]mm* [**-days** *n*] | **-days** *n*}	Crée un plan de production intermédiaire.
planman [*paramètres_connexion*] **deploy** [**-scratch**]	Déploie toutes les règles ne se trouvant pas à l'état de brouillon.
planman [*paramètres_connexion*] **ext** {**-to** *mm/jj/[aa]aa[hh[:]mm[**tz** | **fuseau horaire** nom_fh]]* | **-for** *[h]hh[:]mm* [**-days** *n*] | **-days** *n*}	Crée un plan intermédiaire pour une extension de plan.
planman [*paramètres_connexion*] **showinfo**	Extrait les informations du plan de production.
planman [*paramètres_connexion*] **crttrial** *file_name* [**-from** *mm/jj/[aa]aa* [*hh[:]mm* [**tz** | **fuseau horaire** *nom_fh*]]] {**-to** *mm/jj/[aa]aa[hh[:]mm[**tz** | **fuseau horaire** nom_fh]]* | **-for** *[h]hh[:]mm* [**-days** *n*] | **-days** *n*}	Crée un plan d'essai.

Tableau 113. Commandes utilisées sur le plan (suite)

Syntaxe de la commande ou du script	Action effectuée
planman [*paramètres_connexion*] **exttrial** *file_name* {**-to** *mm/jj/[aa]aa[hh[:]mm[**tz** | **fuseau horaire** nom_fh]]* | **-for** *[h]hh[:]mm* [**-days** *n*] | **-days** *n*}	Crée un plan d'essai avec une extension du plan de production.
planman [*paramètres_connexion*] **crtfc** *file_name* [**-from** *mm/jj/[aa]aa* [*hhmm* [**tz** | **timezone** *fuseau_horaire*]]] {**-to** *mm/jj/[aa]aa[hh[:]mm[**tz** | **fuseau horaire** nom_fh]]* | **-for** *[h]hh[:]mm* [**-days** *n*] | **-days** *n*}	Crée un plan prévisionnel.
planman [*paramètres_connexion*] **unlock**	Déverrouille le plan de production.
ResetPlan [*paramètres_connexion*] [**-scratch**]	Réinitialise le plan de production.
planman reset -scratch	Supprime le plan de préproduction durant la maintenance du fichier Symphony.
planman [*paramètres_connexion*] **resync**	Réplique les données de plan entre le fichier Symphony et la base de données.
planman [*paramètres_connexion*] **checksync**	Surveille la progression et l'issue du processus de réplication du plan de données dans la base de données.

où les valeurs de *paramètres_connexion* sont les suivantes :

```
[-file file_name]
[-host nom_hôte]
[-port nom_port]
[-protocol nom_protocole][-proxy nom_proxy]
[-proxyport numéro_port_proxy]
[-password mot_de_passe_utilisateur]
[-timeout secondes]
[-username nom_utilisateur]
```

Pour plus d'informations, voir «Création et extension d'un plan de production», à la page 84.

Gestion des objets dans la base de données

La présente section comprend les sous-sections suivantes :

Commandes universelles

Cette section décrit le nom et la syntaxe des commandes universelles exécutées à partir du programme **composer**, ainsi que l'autorisation utilisateur requise, le cas échéant, pour les exécuter.

Tableau 114. Commandes universelles

Commande	Syntaxe	Autorisation utilisateur
continue	**continue**&*argument_de_commande*&*argument_de_commande*	Autorisation d'utilisation de composer
edit	**edit** *file_name*	Autorisation d'utilisation de composer
exit	**exit**	Autorisation d'utilisation de composer
aide	**help** *nom_commande*	Autorisation d'utilisation de composer
redo	**redo** *instructions*	Autorisation d'utilisation de composer
validate	**validate** *nom_fichier* [;syntax]	Autorisation d'utilisation de composer
version	**version**	Autorisation d'utilisation de composer

Objets de planification

La présente section contient la syntaxe de définition de tous les objets de planification.

Dans le tableau qui affiche la liste des commandes que vous pouvez exécuter sur l'objet de planification, la variable *file_name* représente un fichier existant lorsqu'elle est utilisée dans la syntaxe des commandes **add** et **replace**, mais elle représente un fichier non existant lorsqu'elle est utilisée dans la syntaxe de la commande **create/extract**.

Agenda

Syntaxe de définition du fichier :

$calendar
nom_agenda [“*description*”]
date [...]

Domaine

Syntaxe de définition du fichier :

domain *nom_domaine*[**description** *"description"*]
  * **manager** *poste_de_travail*
    [**parent** *nom_domaine* | **ismaster**]
end

Règle d'événement

Syntaxe de définition XML :

- eventRule name=" " ruleType=" " isDraft=" " (1, 1)
  - description (0, 1)
  - timeZone (0, 1)
  - validity from=" " to=" " (0, 1)
  - activeTime start=" " end=" " (0, 1)
  - timeInterval amount=" " unit=" " (0, 1)
  - eventCondition eventProvider=" " eventType=" " (1, n)
    - scope (0, 1)
    - filteringPredicate (0, 1)
      - attributeFilter name=" " operator="eq" (0, n)
        - value (1, n)
      - attributeFilter name=" " operator="ne" (0, n)
        - value (1, n)
      - attributeFilter name=" " operator="le" (0, n)
        - value (1, 1)
      - attributeFilter name=" " operator="ge" (0, n)
        - value (1, 1)
      - attributeFilter name=" " operator="range" (0, 1)
        - value (1, 2)
  - correlationAttributes (0, 1)
    - attribute name=" " (1, n)
  - action actionProvider=" " actionType=" " responseType=" " (0, n)
    - description (0, 1)
    - scope (0, 1)
    - parameter name=" "(1, n)
    - value (1, 1)

Travail

Syntaxe de définition du fichier :

$jobs
[*poste_de_travail***#**]*nom_travail*
  {**scriptname** *file_name* | **docommand** *"commande"* |
task *définition_travail*}
  streamlogon *nom_utilisateur*
  [**description** *"description"*]
  [**tasktype** *type_tâche*]
  [**interactive**][1]
  [**rccondsucc** "*Condition de succès*"]
  [**recovery**
    {**stop** | **continue** | **rerun**}
    [**after** [*poste_de_travail***#**]*nom_travail*]
    [**abendprompt** *"texte"*]]

Remarque :

1. Ce mot clé est uniquement disponible sur les plates-formes Windows.

Flot de travaux

Syntaxe de définition du fichier :

schedule [*poste_de_travail*#]*nom_flot_travaux*
  # *commentaire*
  [**validfrom** *date*]
  [**timezone** | **tz** *fuseau_horaire*]
  [**description** *"texte"*]
  [**draft**]
  [**vartable** *nom_table*]
  [**freedays** *nom_agenda* [**-sa**] [**-su**]]
  [**on** [**runcycle** *nom*]
    [**validfrom** *date*] [**validto** *date*]
    [**description** *"texte"*]
    [**vartable** *nom_table*]
   {*date* | *jour* | *agenda* | *demande* | *"icalendar"* | *groupe_cycle_exécution*} [**,...**]
    [**fdignore** | **fdnext** | **fdprev**]
    [({**at** *heure* [+*n* **day**[**s**]] |
    schedtime *heure* [+*n* **day**[**s**]]}
    [**until** *heure* [+*n* **day**[**s**]] [**onuntil** *action*]]
    [**deadline** *heure* [+*n* **day**[**s**]]])]]
  [**,...**]
  [**except** [**runcycle** *nom*]
      [**validfrom** *date*] [**validto** *date*]
      [**description** *"texte"*]
      {*date* | *jour* | *agenda* | *demande* | *"icalendar"* | *groupe_cycle_exécution*} [**,...**]
      [**fdignore** | **fdnext** | **fdprev**]
      [{(**at** *heure* [+*n* **day**[**s**]])] |
      (**schedtime** *heure* [+*n* **day**[**s**]])}]
  [**,...**]
  [{**at** *heure* [***timezone*** | ***tz*** *fuseau_horaire*] [+*n* **day**[**s**]] |
  schedtime *heure* [***timezone*** | ***tz*** *fuseau_horaire*] [+*n* **day**[**s**]]}]
  [**until** *heure* [**timezone** | **tz** *fuseau_horaire*] [+*n* **day**[**s**]] [**onuntil** *action*]]
  [**deadline** *heure* [**timezone** | **tz** *fuseau_horaire*] [+*n* **day**[**s**]]]
  [**carryforward**]
  [**matching** {**previous** | **sameday** | **relative from** [**+** | **-**] *heure* **to** [**+** | **-**] *heure* |
    from *heure* [+ | -*n* day[s]] **to** *heure* [+ *n* day[s]] [**,...**]}]
  [**follows** {[*agent_réseau*::][*poste_de_travail*#]*nom_flot_travaux*[**.***nom_travail* | **@**] [**previous** |
    sameday | **relative from** [**+** | **-**] *heure* **to** [**+** | **-**] *heure* |
    from *heure* [+ | -*n* day[s]] **to** *heure* [+ | -*n* day[s]]
    ]}] [,...]] [...]
  [**keysched**]
  [**limit** *nombre_max_travaux*]
  [**needs** { [*n*] [*poste_de_travail*#]*nom_ressource* } [,...]] [...]
  [**opens** { [*poste_de_travail*#]"*file_name*" [(*qualifiant*)] [,...] }] [...]
  [**priority** *nombre* | **hi** | **go**]
  [**prompt** {*nom_invite* | "[**:** | **!**]*texte*"} [,...]] [...]
 :
instruction-travail
  # *commentaire*
  [{**at** *heure* [**timezone** | **tz** *fuseau_horaire*] [+*n* **day**[**s**]] |
  schedtime *heure* [**timezone** | **tz** *fuseau_horaire*] [+*n* **day**[**s**]]}][**,...**]
  [**until** *heure* [**timezone** | **tz** *fuseau_horaire*] [+*n* **day**[**s**]] [**onuntil** *action*]
  [**deadline** *heure* [**timezone** | **tz** *fuseau_horaire*] [+*n* **day**[**s**]]] [**,...**]
  [**maxdur** *heure* | *pourcentage* % **onmaxdur** *action*]
  [**mindur** *heure* | *pourcentage* % **onmindur** *action*]

[**every** *fréquence*]
[**follows** {[*agent_réseau*::][*poste_de_travail*#]*nom_flot_travaux*{.*nom_travail* **@**} [**previous**|
sameday|**relative from [+|-]** *heure* **to [+|-]** *heure*|
from *heure* [+|-*n* day[s]] **to** *heure* [+|-*n* day[s]]
]}] [,...]] [...]
[**confirmed**]
[**critical**]
[**keyjob**]
[**needs** { [*n*] [*poste_de_travail*#]*nom_ressource* } [,...]] [...]
[**opens** { [*poste_de_travail*#]"*file_name*" [(*qualifiant*)] [,...] }] [...]
[**priority** *nombre* | **hi** | **go**]
[**prompt** {*nom_invite*|"[**:**|**!**]*texte*"} [,...]] [...]

[*instruction-travail*...]
end

Paramètre

Syntaxe de définition du fichier :

$parm
[*nom_table.*]*nom_variable "valeur_variable"*

Invite

Syntaxe de définition du fichier :

$prompt
nom_invite "[**:** | **!**]*texte*"

Ressource

Syntaxe de définition du fichier :

$resource
poste_de_travail#*nom_ressource unités* [*"description"*]

Groupe de cycle d'exécution

Syntaxe de définition du fichier :

$runcyclegroup
nom_groupe_cycle_exécution [*"description"*]
vartable*nom_table*
[**freedays** *nom_agenda* [**-sa**] [**-su**]]
[**on** [**runcycle** *nom*]
[**validfrom** *date*] [**validto** *date*]
[**description** *"texte"*]
[**vartable** *nom_table*]
{*date*|*jour*|*agenda*|*demande*|*"icalendar"*}
[**,...**]
[**fdignore**|**fdnext**|**fdprev**][**subset** *nom_sous-ensemble* **AND**|**OR**]
[({**at** *heure* [+*n* **day**[**s**]] |
schedtime *heure* [+*n* **day**[**s**]]}
[**until** *heure* [+*n* **day**[**s**]] [**onuntil** *action*]]
[**deadline** *heure* [+*n* **day**[**s**]]])]
[**,...**]
[**except** [**runcycle** *nom*]
[**validfrom** *date*] [**validto** *date*]

[**description** *"texte"*]
{*date* | *jour* | *agenda* | *demande* | *"icalendar"*}
[**,...**]
[**fdignore** | **fdnext** | **fdprev**][**subset** *nom_sous-ensemble* **AND** | **OR**]
[{(**at** *heure* [+*n* **day**[**s**]])] |
(**schedtime** *heure* [+*n* **day**[**s**]])}]
[**,...**]
[{**at** *heure* [***timezone*** | ***tz*** *fuseau_horaire*] [+*n* **day**[**s**]] |
schedtime *heure* [***timezone*** | ***tz*** *fuseau_horaire*] [+*n* **day**[**s**]]}]
[**until** *heure* [**timezone** | **tz** *fuseau_horaire*] [+*n* **day**[**s**]] [**onuntil** *action*]]
[**deadline** *heure* [**timezone** | **tz** *fuseau_horaire*] [+*n* **day**[**s**]]]
end

Table de variables

Syntaxe de définition du fichier :

vartable *nom_table*
[**description** *"description"*]
[**isdefault**]
members
[*nom_variable "valeur_variable"*]
...
[*nom_variable "valeur_variable"*]
end

Pour plus d'informations, voir Chapitre 6, «Personnalisation de votre charge de travail à l'aide des tables de variables», à la page 121.

Poste de travail

Syntaxe de définition du fichier :

cpuname *poste_de_travail* [**description** *"description"*]
[**vartable** *nom_table*]
os *type_sys_expl*
[**node** *nom_hôte*] [**tcpaddr** *port*]
[**secureaddr** *port*][**timezone** | **tz** *fuseau_horaire*]
[**domain** *nom_domaine*]
[**for maestro** [**host** *poste_de_travail_hôte* [**access** *méthode*]]
[**type fta** | **s-agent** | **x-agent** | **manager** | **broker** | **agent** |
pool | **d-pool** | **rem-engine**]
[**ignore**]
[**autolink on** | **off**]
[**behindfirewall on** | **off**]
[**securitylevel enabled** | **on** | **force**]
[**fullstatus on** | **off**]
[**server** *id_serveur*]]
[**protocole** *http* | *htpps*]
[**membres** [*poste_de_travail*] [...]]
[**exigences** *définition_jsdl*]]
end

cpuname *poste_de_travail* [**description** *description*]
[**vartable** *nom_table*]
os *type_sys_expl*
node *nom_hôte* [**tcpaddr** *port*]
[**secureaddr** *port*][**timezone** | **tz** *fuseau_horaire*]

```
[domain nom_domaine]
[for maestro    [host poste_de_travail_hôte [access méthode]]
      [type fta | s-agent | x-agent | manager]
      [ignore]
      [autolink on | off]
      [behindfirewall on | off]
      [securitylevel enabled | on | force]
      [fullstatus on | off]
      [server id_serveur]]
end
```

Classe de poste de travail

Syntaxe de définition du fichier :

```
cpuclass classe_poste_de_travail [description "description"]
                                 [ignore]
   members [poste_de_travail | @] [...]
end
```

Définition d'utilisateur

Syntaxe de définition du fichier :

```
nom_utilisateur[poste_de_travail#][domaine\]nom_utilisateur
      mot_de_passe "mot_de_passe"end
```

Commandes composer

Cette section décrit les opérations que vous pouvez effectuer dans la base de données à l'aide du programme d'interface de ligne de commande **composer** en utilisant la syntaxe suivante :

```
composer [paramètres_connexion] [-defaultws uc_tws]
       ["commande[&[commande]][...]"]
```

où *paramètres_connexion*, s'ils ne sont pas fournis dans les fichiers `localopts` ou `useropts`, sont les suivants :

```
[-file file_name]|
[-host nom_hôte]
[-port nom_port]
[-protocol nom_protocole]
[-proxy nom_proxy]
[-proxyport numéro_port_proxy]
[-password mot_de_passe_utilisateur]
[-timeout secondes]
[-username nom_utilisateur]
```

Pour plus d'informations, voir «Configuration des options pour l'utilisation des interfaces utilisateur», à la page 59.

Ces opérations peuvent uniquement être exécutées à partir de toute ligne de commande de client composer installée.

Dans le tableau 115, à la page 777 qui affiche la liste des commandes que vous pouvez exécuter sur l'objet de planification, la variable *file_name* représente un fichier existant lorsqu'elle est utilisée dans la syntaxe des commandes **add** et **replace**, mais elle représente un fichier non existant lorsqu'elle est utilisée dans la syntaxe de la commande **create/extract**.

Tableau 115. Commandes composer

Commande	Syntaxe	Autorisation utilisateur
add	{**add** | **a**} *file_name* [**;unlock**]	*add* ou *modify*
authenticate	{**authenticate** | **au**} [**username=***nom_utilisateur* **password=***mot_de_passe*]	
continuer	{**continue** | **c**}	
create extract	{**create** | **cr** | **extract** | **ext**} *file_name* **from** {[**calendars** | **calendar** | **cal**=*nom_agenda*] | [**eventrule** | **erule** | **er**=*nom_règle_événement*] | [**parms** | **parm** | **vb**=[*nom_table.*]*nom_variable*] | [**vartable** | **vt**=*nom_table*] | [**prompts** | **prom**=*nom_invite*] | [**resources** | **resource** | **res**=[*poste_de_travail#*]*nom_ressource*] | [**runcyclegroup** | **rcg**=*nom_groupe_cycle_exécution*] | [**cpu**={*nom_poste_travail* | *classe_poste_de_travail* | *nom_domaine*}] | [**workstation** | **ws**=*nom_poste_travail*] | [**workstationclass** | **wscl**=*nom_classe_poste_de_travail*] | [**domain** | **dom**=*nom_domaine*] | [**jobs** | **jobdefinition** | **jd**=[*nom_poste_travail#*]*nom_travail*] | [**sched** | **jobstream** | **js**= [*nom_poste_travail#*]*nom_flot_travaux* *[**valid from** date* | *__valid to__ date* | *__valid in__ date date]* *[**;full**]*] | [**users** | **user**=[*nom_poste_travail#*]*nom_utilisateur* [;password]]} [**;lock**]	*display*
delete	{**delete** | **de**} {[**calendars** | **calendar** | **cal**=*nom_agenda*] | [**domain** | **dom**=*nom_domaine*] | [**eventrule** | **erule** | **er**=*nom_règle_événement*] | [**parms** | **parm** | **vb**=[*nom_table.*]*nom_variable*] | [**prompts** | **prom**=*nom_invite*] | [**resources** | **resource** | **res**=[*poste_de_travail#*]*nom_ressource*] | [**runcyclegroup** | **rcg**=*nom_groupe_cycle_exécution*] | [**vartable** | **vt**=*nom_table*] | [**wat**=*workloadapplicationtemplatename*] [**unité centrale**={*workstationame* [**;force**] | *workstationclassname* [**;force**] | *domainame*}] [**workstation** | **ws**=*workstationame*] [**;force**] | [**workstationclass** | **wscl**=*workstationclassname*] [**;force**] | [**jobs** | **jobdefinition** | **jd**=[*nom_poste_travail#*]*nom_travail*] | [**sched** | **jobstream** | **js**= [*nom_poste_travail#*]*nom_flot_travaux* *[**valid from** date* | *__valid to__ date* | *__valid in__ date date]*] | [**users** | **user**=[*nom_poste_travail#*]*nom_utilisateur*]} [**;noask**]	*supprimer*
display	{**display** | **di**} {[**calendars** | **calendar** | **cal**=*nom_agenda*] | [**eventrule** | **erule** | **er**=*nom_règle_événement*] | [**parms** | **parm** | **vb**=*nom_variable.*]*nom_variable*] | [**vartable** | **vt**=*nom_table*] | [**prompts** | **prom**=*nom_invite*] | [**resources** | **resource** | **res**=[*poste_de_travail#*]*nom_ressource*] | [**runcyclegroup** | **rcg**=*nom_groupe_cycle_exécution*] | [**cpu**={*nom_poste_travail* | *nom_classe_poste_de_travail* | *nom_domaine*}] [**workstation** | **ws**=*nom_poste_travail*] | [**workstationclass** | **wscl**=*nom_classe_poste_de_travail*] | [**domain** | **dom**=*nom_domaine*] | [**jobs** | **jobdefinition** | **jd**=[*nom_poste_travail#*]*nom_travail*] | [**sched** | **jobstream** | **js**= [*nom_poste_travail#*]*nom_flot_travaux* *[**valid from** date* | *__valid to__ date* | *__valid in__ date date]* *[**;full**]*] | [**users** | **user**=[*nom_poste_travail#*]*nom_utilisateur*]} [**;offline**]	*display*
edit	{**edit** | **ed**} *file_name*	
exit	{**exit** | **e**}	

Tableau 115. Commandes composer (suite)

Commande	Syntaxe	Autorisation utilisateur
list print	{**list** | **l**} {[**calendars** | **calendar** | **cal**=*nom_agenda*] | [**eventrule** | **erule** | **er**=*nom_règle_événement*] | [**parms** | **parm** | **vb**=[*nom_table.*]*nom_variable*] | [**vartable** | **vt**=*nom_table*] | [**prompts** | **prom**=*nom_invite*] | [**resources** | **resource** | **res**=[*poste_de_travail#*]*nom_ressource*] | [**runcyclegroup** | **rcg**=*nom_groupe_cycle_exécution*] | [**cpu**={*nom_poste_travail* | *nom_classe_poste_de_travail* | *nom_domaine*}] [**workstation** | **ws**=*nom_poste_travail*] | [**workstationclass** | **wscl**=*nom_classe_poste_de_travail*] | [**domain** | **dom**=*nom_domaine*] | [**jobs** | **jobdefinition** | **jd**=[*nom_poste_travail#*]*nom_travail*] | [**sched** | **jobstream** | **js**= [*nom_poste_travail#*]*nom_flot_travaux* *[**valid from** date* | ***valid to** date* |***valid in** date date]* | [**users** | **user**=[*nom_poste_travail#*]*nom_utilisateur*]} [**;offline**]	*display*
lock	{**lock** | **lo**} {[**calendars** | **calendar** | **cal**=*nom_agenda*] | [**eventrule** | **erule** | **er**=*nom_règle_événement*] [**parms** | **parm** | **vb**=[*nom_table.*]*nom_variable*] | [**vartable** | **vt**=*nom_table*] | [**prompts** | **prom**=*nom_invite*] | [**resources** | **resource** | **res**=[*poste_de_travail#*]*nom_ressource*] | [**runcyclegroup** | **rcg**=*nom_groupe_cycle_exécution*] | [**cpu**={*nom_poste_travail* | *nom_classe_poste_de_travail* | *nom_domaine*}] [**workstation** | **ws**=*nom_poste_travail*] | [**workstationclass** | **wscl**=*nom_classe_poste_de_travail*] | [**domain** | **dom**=*nom_domaine*] | [**jobs** | **jobdefinition** | **jd**=[*nom_poste_travail#*]*nom_travail*] | [**sched** | **jobstream** | **js**= [*nom_poste_travail#*]*nom_flot_travaux* *[**début de validité** date* |***fin de validité** date* |***valid*** ***in** date date]* | [**users** | **user**=[*nom_poste_travail#*]*nom_utilisateur*]}	*modify*
modify	{**modify** | **m**} {[**calendars** | **calendar** | **cal**=*nom_agenda*] | [**eventrule** | **erule** | **er**=*nom_règle_événement*] | [**parms** | **parm** | **vb**=[*nom_table.*]*nom_variable*] | [**vartable** | **vt**=*nom_table*] | [**prompts** | **prom**=*nom_invite*] | [**resources** | **resource** | **res**=[*poste_de_travail#*]*nom_ressource*] | [**runcyclegroup** | **rcg**=*nom_groupe_cycle_exécution*] | [**cpu**={*nom_poste_travail* | *nom_classe_poste_de_travail* | *nom_domaine*}] [**workstation** | **ws**=*nom_poste_travail*] | [**workstationclass** | **wscl**=*nom_classe_poste_de_travail*] | [**domain** | **dom**=*nom_domaine*] | [**jobs** | **jobdefinition** | **jd**=[*nom_poste_travail#*]*nom_travail*] | [**sched** | **jobstream** | **js**= [*nom_poste_travail#*]*nom_flot_travaux* *[**valid from** date* |***valid to** date* |***valid in** date date]* *[**;full**]* | [**users** | **user**=[*nom_poste_travail#*]*nom_utilisateur*]}	*modify* ou *add*
new	**new** [**calendar** | **domain** | **eventrule** | **job** | **jobstream** | **parameter** | **prompt** | **resource** | **runcyclegroup** | **user** | **vartable** | **workstation** | **workstation_class**]	*add* ou *modify*

Tableau 115. Commandes composer (suite)

Commande	Syntaxe	Autorisation utilisateur
rename	{**rename** | **rn**} {**calendars** | **calendar** | **cal** | **parms** | **parm** | **vb** | **vartable** | **vt** | **prompts** | **prom** | **resorces** | **resource** | **res** | **runcyclegroup** | **rcg** | **workstation** | **ws** | **workstationclass** | **wscl** | **domain** | **dom** | **jobs** | **jobdefinition** | **jd** | **jobsched** | **jb** | **eventrule** | **erule** | **er** **sched** | **jobstream** | **js** | **users** | **user** } *ancien_identificateur_objet nouvel_identificateur_objet*	*add* et *delete*
replace	{**replace** | **rep**} *file_name* [**;unlock**]	*modify* ou *add*
unlock	{**unlock** | **u**} {[**calendars** | **calendar** | **cal**=*nom_agenda*] | [**eventrule** | **erule** | **er**=*nom_règle_événement*] | [**parms** | **parm** | **vb**=[*nom_table.*]*nom_variable*] | [**vartable** | **vt**=*nom_table*] | [**prompts** | **prom**=*nom_invite*] | [**resources** | **resource** | **res**=[*poste_de_travail#*]*nom_ressource*] | [**runcyclegroup** | **rcg**=*nom_groupe_cycle_exécution*] | [**cpu**={*nom_poste_travail* | *nom_classe_poste_de_travail* | *nom_domaine*}] [**workstation** | **ws**=*nom_poste_travail*] | [**workstationclass** | **wscl**=*nom_classe_poste_de_travail*] | [**domain** | **dom**=*nom_domaine*] | [**jobs** | **jobdefinition** | **jd**=[*nom_poste_travail#*]*nom_travail*] | [**sched** | **jobstream** | **js**= [*nom_poste_travail#*]*nom_flot_travaux* *[**début de validité** date* | ***fin de validité** date* | ***valid in** date date]*] | [**users** | **user**=[*nom_poste_travail#*]*nom_utilisateur*]} [**;forced**]	*modify* et *unlock*
validate	{**validate** | **val**} *file_name* [**;syntax**]	

Gestion des objets dans le plan

Cette section décrit les opérations que vous pouvez effectuer sur le plan à l'aide du programme d'interface de ligne de commande **conman** en utilisant la syntaxe suivante :

```
conman ["command[&[command]...] [&]"]
```

Commandes conman

La présente section répertorie les commandes que vous pouvez exécuter à partir du programme **conman**.

Voici comment accéder à la ligne de commande **conman** :

```
conman [paramètres_connexion] ["commande[&[commande]...] [&]"]
```

où *paramètres_connexion*, s'ils ne sont pas fournis dans les fichiers `localopts` ou `useropts`, sont les suivants :

```
[-file file_name]
[-host nom_hôte]
[-port nom_port]
[-protocol nom_protocole]
[-proxy nom_proxy]
[-proxyport numéro_port_proxy]
[-password mot_de_passe_utilisateur]
[-timeout secondes]
[-username nom_utilisateur]
```

Pour plus d'informations, voir «Configuration des options pour l'utilisation des interfaces utilisateur», à la page 59.

Voici comment sélectionner des travaux dans les commandes :

```
[poste_de_travail#]
{nom_flot_travaux(hhmm[ date]) travail|numéro_travail}
[{+|~}qualifiant_travail[...]]
```

ou :

```
[poste_de_travail#]
ID_flot_travaux.
travail
[{+|~]qualifiant_travail[...]]
 ;schedid
```

Voici comment sélectionner des flots de travaux dans les commandes :

```
[poste_de_travail#]
nom_flot_travaux(hhmm[date])
[{+|~}nom_flot_travaux[...]]
```

ou :

```
[poste_de_travail#]
ID_flot_travaux
 ;schedid
```

Vous pouvez exécuter ces commandes à partir de différents types de poste de travail. Dans ce tableau :

F représente les gestionnaires de domaine et les agents tolérants aux pannes.

S représente les agents standard.

Vous trouverez le nom et la syntaxe de chaque commande, ainsi que les types de poste de travail à partir desquels vous pouvez l'émettre et, le cas échéant, l'autorisation utilisateur requise.

Tableau 116. Commandes pouvant être exécutées à partir de conman

Commande	Syntaxe	Types de poste de travail	Autorisation utilisateur
adddep job	{**adddep job** | **adj**} *travail_sélectionné* **;***dépendance*[**;**...] [**;noask**]	F	*adddep* - (*à utiliser* lorsque les mots clés prompts et needs sont utilisés)
adddep sched	{**adddep sched** | **ads**} *flot_travaux_sélectionné* **;***dépendance*[**;**...] [**;noask**]	F	*adddep* - (*à utiliser* lorsque les mots clés prompts et needs sont utilisés)
altpass	**altpass** [*poste_de_travail***#**] *nom_utilisateur* [**;"***mot_de_passe***"**]	F	*altpass*
altpri	{**altpri** | **ap**} *travail_sélectionné* | *flot_travaux_sélectionné* [**;***pri*] [**;noask**]	F	*altpri*
bulk_discovery	{**bulk_discovery** | **bulk**}	F	*display*

Tableau 116. Commandes pouvant être exécutées à partir de conman (suite)

Commande	Syntaxe	Types de poste de travail	Autorisation utilisateur
cancel job	{**cancel job** | **cj**} *travail_sélectionné* [**;pend**] [**;noask**]	F	*cancel*
cancel sched	{**cancel sched** | **cs**} *flot_travaux_sélectionné* [**;pend**] [**;noask**]	F	*cancel*
checkhealthstatus	{**checkhealthstatus** | **chs**} [*poste_de_travail*]	M,F,S	
confirm	{**confirm** | **conf**} *travail_sélectionné* **;**{**succ** | **abend**} [**;noask**]	F	*confirm*
console	{**console** | **cons**} [**sess** | **sys**] [**;level=***niveau_message*]	F-S	*console*
continue	{**continue** | **cont**}	F-S	
deldep job	{**deldep job** | **ddj**} *jobselect* **;***dépendance*[**;**...] [**;noask**]	F	*deldep*
deldep sched	{**deldep sched** | **dds**} *jstreamselect* **;***dépendance*[**;**...] [**;noask**]	F	*deldep*
deployconf	{**deployconf** | **deploy**} [*domaine***!**]*poste_de_travail*	F, S	Droit de démarrer (*start*) des actions sur des objets *cpu*
display	{**display file** | **df**} *file_name* [**;offline**] {**display job** | **dj**} *jobselect* [**;offline**] {**display sched** | **ds**} *jstreamselect* [**valid** {**at** *date* | **in** *date date*} [**;offline**]	F-S[1]	*display*
exit	{**exit** | **e**}	F-S	
fence	{**fence** | **f**} *poste_de_travail* **;***pri* [**;noask**]	F	*fence*
help (UNIX uniquement)	{**help** | **h**} {*commande*|*mot_clé*}	F-S	
kill	{**kill** | **k**} *jobselect* [**;noask**]	F	*kill*
limit cpu	{**limit cpu** | **lc**} *poste de travail* **;***limite* [**;noask**]	F	*limit*
limit sched	{**limit sched** | **ls** } *flot_travaux_sélectionné* **;***limite* [**;noask**]	F	*limit*

Tableau 116. Commandes pouvant être exécutées à partir de conman (suite)

Commande	Syntaxe	Types de poste de travail	Autorisation utilisateur
link	{**link** | **lk**} [*domaine***!**]*poste_de_travail* **[;noask]**	F-S	*link*
listsym	{**listsym** | **lis**} **[trial** | **forecast]** **[;offline]**	F	
recall	{**recall** | **rc**} [*poste de travail*] **[;offline]**	F	*display*
redo	{**redo** | **red**}	F-S	
release job	{**release job** | **rj**} *travail_sélectionné* **[;***dépendance***[;...]]** **[;noask]**	F	*release*
release sched	{**release sched** | **rs**} *flot_travaux_sélectionnés* **[;***dépendance***[;...]]** **[;noask]**	F	*release*
reply	{**reply** | **rep**} { *nom_invite* | [*poste_de_travail***#**]*num_message*} ***;reply*** **[;noask]**	F	*reply*
rerun	{**rerun** | **rr**} *travail_sélectionné* **[;from=[***poste_de_travail***#]***travail* **[;at=***heure***]** **[;pri=***pri***]]** **[;step=***étape***]** **[;noask]**	F	*rerun*
resource	{**resource** | **reso**} [*poste_de_travail***#**] *ressource***;***num* **[;noask]**	F	*resource*
setsym	{**setsym** | **set**} **[essai** | **prévision]** [*num_fichier*]	F	
showcpus	{**showcpus** | **sc**} [[*domaine***!**]*poste de travail*] **[;info** | **;link]** **[;offline]**	F-S	*list*[2]
showdomain	{**showdomain** | **showd**} [*domaine*] **[;info]** **[;offline]**	F-S	*list*[2]
showfiles	{**showfiles** | **sf**} [[*num_poste_de_travail***]***fichier*] **[;***état***[;...]]** **[;keys]** **[;offline]** {**showfiles** | **sf**} [[*poste de travail***#]***fichier*] **[;***état***[;...]]** **[;deps[;keys** | **info** | **logon]]** **[;offline]**	F	

Tableau 116. Commandes pouvant être exécutées à partir de conman (suite)

Commande	Syntaxe	Types de poste de travail	Autorisation utilisateur
showjobs	{**showjobs** | **sj**} [*travail_sélectionné*] [**;deps**[**;keys** | **info** | **logon**]] [**;short** | **single**] [**;offline**] [**;showid**] [**;props**] {**showjobs** | **sj**} [*travail_sélectionné* | [**poste_de_travail#**]*jobnumber.hhmm*] [**;stdlist**[**;keys**]] [**;short** | **single**] [**;offline**] [**;showid**] [**;props**]	F	*list*[2]
showprompts	{**showprompts** | **sp**} [*invite_sélectionnée*] [**;keys**] [**;offline**] {**showprompts** | **sp**} [*invite_sélectionnée*] [**;deps**[**;keys** | **info** | **logon**]][**;offline**]	F	*list*[2]
showresources	{**showresources** | **sr**} [[*poste de travail***#**]*nom_ressource*] [**;keys**] [**;offline**] {**showresources** | **sr**} [[*poste de travail***#**]*nom_ressource*] [**;deps**[**;keys** | **info** | **logon**]] [**;offline**]	F	*list*[2]
showschedules	{**showscheds** | **ss**} [*flot_travaux_sélectionné*] [**;keys**] [**;offline**] [**;showid**] {**showscheds** | **ss**} [*flot_travaux_sélectionné*] [**;deps**[**;keys** | **info** | **logon**]] [**;offline**] [**;showid**]	F	*list*[2]
shutdown	{**shutdown** | **shut**} [**;wait**]	F-S	*shutdown*
start	**start**[*domaine***!**]*poste_de_travail* [**;mgr**] [**;noask**] [**;demgr**]	F-S	*start*
startappserver	**startappserver**[*domaine***!**]*poste_de_travail* [**;wait**]	F-S	Droit de démarrer (*start*) des actions sur des objets *cpu*
startevtp	{**starteventprocessor** | **startevtp**} [*domaine***!**]*poste de travail*	M[4]	Droit de démarrer (*start*) des actions sur des objets *cpu*
startmon	{**startmon** | **startm**} [*domaine***!**]*poste de travail* [**;noask**]	F-S	Droit de démarrer (*start*) des actions sur des objets *cpu*

Tableau 116. Commandes pouvant être exécutées à partir de conman (suite)

Commande	Syntaxe	Types de poste de travail	Autorisation utilisateur
status	{**status** | **stat**}	F-S	appserver
stop	**stop** [*domaine***!**]*poste_de_travail* [**;wait**] [**;noask**]	F-S	*stop*
stop ;progressive	**stop ;progressive**		*stop*
stopappserver	{**stopappserver** | **stopapps**} [*domaine***!**]*poste de travail* [**;wait**]	F-S	Droit d'arrêter (*stop*) des actions sur des objets *cpu*
stopevtp	{**stopeventprocessor** | **stopevtp**} [*domaine***!**][*poste de travail*]	M[4]	Droit d'arrêter (*stop*) des actions sur des objets *cpu*
stopmon	{**stopmon** | **stopm**} [*domaine***!**]*poste de travail* [**;wait**] [**;noask**]	F-S	Droit d'arrêter (*stop*) des actions sur des objets *cpu*
submit docommand	{**submit** **docommand** | **sbd**} [*poste de travail*#]**"***cmd***"** [**;alias**[**=***nom*]] [**;into=**[*poste de travail*#] {*ID_flot_travaux***;schedid** |*nom_flot_travaux* ([*hhmm*[*date*]])}] [**;***option_travail*[**;**...]]	F-S	*submit* - (*à utiliser* lorsque les mots clés prompts et needs sont utilisés)
submit file	{**submit** **file** | **sbf**} **"***file_name***"** [**;alias**[**=***nom*]] [**;into=**[*poste_de_travail*#]{*ID_flot_travaux* **;schedid** | *nom_flot_travaux*([*hhmm*[*date*]])}] [**;***option_travail*[**;**...]] [**;noask**]	F-S	*submit* - (*à utiliser* lorsque les mots clés prompts et needs sont utilisés)
submit job	{**submit** **job** | **sbj**} [*poste de travail*#]*nom_travail* [**;alias**[**=***nom*]] [**;into=**[*poste_de_travail*#]{*ID_flot_travaux* **;schedid** | *nom_flot_travaux*([*hhmm*[*date*]])}] [**;***option_travail*[**;**...]] [**;vartable=***nom_table*] [**;noask**]	F-S[3]	*submit* - (*à utiliser* lorsque les mots clés prompts et needs sont utilisés)
submit sched	{**submit** **sched** | **sbs**} [*poste de travail*#]*nom_flot_travaux* [**;alias**[**=***nom*]] [**;***option_flot_travaux*[**;**...]] [**;vartable=***nom_table*] [**;noask**]	F-S[3]	*submit* - (*à utiliser* lorsque les mots clés prompts et needs sont utilisés)
switchevtp	{**switcheventprocessor** | **switchevtp**} *poste de travail*	M[4]	Droit de *démarrer* et *arrêter* des actions sur des objets *cpu*
switchmgr	{**switchmgr** | **switchm**} *domaine***;***nouv_gest*	F	*start stop*
system	[**:** | **!**] *commande système*	F-S	
tellop	{**tellop** | **to**} [*texte*]	F-S	
unlink	**unlink** [*domaine***!**]*poste_de_travail* [**;noask**]	F-S	*unlink*

Tableau 116. Commandes pouvant être exécutées à partir de conman (suite)

Commande	Syntaxe	Types de poste de travail	Autorisation utilisateur
version	{**version** | **v**}	F-S	

où :

(1) Indique que pouvez uniquement afficher des fichiers sur un agent standard.

(2) Vous devez avoir un accès **list** à l'objet qui est affiché si l'option ***enListSecChk*** a été définie sur **yes** sur le gestionnaire de domaine maître lorsque le plan de production a été créé ou étendu.

(3) Indique que vous pouvez utiliser les commandes submit job (**sbj**) et submit sched (**sbs**) sur un agent standard en utilisant les paramètres de connexion ou en spécifiant les paramètres dans le fichier `useropts` lors de l'appel de la ligne de commande **conman**.

(4) Vous pouvez exécuter cette commande sur des gestionnaires de domaine maîtres et maîtres de sauvegarde, ainsi que sur des postes de travail installés comme maîtres de sauvegarde mais utilisés comme instances ordinaires de agent tolérant aux pannes.

Commandes d'utilitaire

Cette section contient la liste de commandes d'utilitaire que vous pouvez exécuter à partir de l'invite de commande du système d'exploitation. Les commandes d'utilitaire sont divisées en trois groupes : celles que vous pouvez exécuter à la fois sur les systèmes d'exploitation UNIX et Windows, celles que vous ne pouvez exécuter que sur UNIX, et celles que vous ne pouvez exécuter que sur Windows.

Commandes d'utilitaire disponibles à la fois pour les systèmes d'exploitation UNIX et Windows

Tableau 117. Commandes d'utilitaire disponibles à la fois pour UNIX et Windows

Commande	Syntaxe
at	**at -V | -U** **at -s** *flot_travaux* | **-q** *file_attente spéc-heure*
batch	**batch -V | -U** **batch** [**-s** *flot_travaux*]
cpuinfo	**cpuinfo -V | -U** **cpuinfo** *poste_de_travail* [*type_info*] [...]
datecalc	**datecalc -V | -U** **datecalc** *date-base* [*décalage*] [**pic** *format*][**freedays** *nom_agenda* [**-sa**] [**-su**]] **datecalc -t** *heure* [*date-base*] [*décalage*] [**pic** *format*] **datecalc** *aaaammjjhhmm* [*décalage*] [**pic** *format*]

Tableau 117. Commandes d'utilitaire disponibles à la fois pour UNIX et Windows (suite)

Commande	Syntaxe
delete	**delete -V | -U** **delete** *file_name*
evtdef	**evtdef -U | -V** **evtdef** [*paramètres de connexion*] **dumpdef** *chemin d'accès au fichier* **evtdef** [*paramètres de connexion*] **loaddef** *chemin d'accès au fichier*
evtsize	**evtsize -V | -U** **evtsize** *file_name taille* **evtsize -compact** *file_name* [*taille*] **evtsize -info** *file_name* **evtsize -show** *file_name* **evtsize -info | -show pobox**
jobinfo	**jobinfo -V | -U** **jobinfo** *option-travail* [...]
jobstdl	**jobstdl -V | -U** **jobstdl** [**-day** *num*] [{**-first** | **-last** | **-num** *n* | **-all**}] **[-twslog]** [{**-name** ["*nom_flot_travaux* [(*hhmm date*),(*ID_flot_travaux*)].] *nom_travail*" | *num_travail* | **-schedid** *ID_flot_travaux.nom_travail*}]
maestro	**maestro** [**-V | -U**]
makecal	**makecal** [**-c** *nom*] **-d** *n* | **-e** | {**-f 1 | 2 | 3 -s** *date*} | **-l** | **-m** | **-p** *n* | {**-r** *n* **-s** *date*} | **-w** *n* [**-i** *n*] [**-x | -z**][**-freedays** *nom_agenda* [**-sa**] [**-su**]]
morestdl	**morestdl -V | -U** **morestdl** [**-day** *num*] [**-first** | **-last** | **-num** *n* | **-all**] **[-twslog]** [{**-name** ["*nom_flot_travaux* [(*hhmm date*),(*ID_flot_travaux*)].] *nom_travail*" | *num_travail* | **-schedid** *ID_flot_travaux.nom_travail*}]
param	**param -u | -V** **param {-c | -ec}** [*fichier.section.*|*fichier.*|*section.*] *variable* [*valeur*] **param** [*fichier.section.*|*fichier.*|*section.*] *variable* **param {-d | -fd}** [*fichier.section.*|*fichier.*|*section.*] *variable*
parms	**parms** {[**-V | -U**] | **-build**} **parms {-replace | -extract}** *file_name* **parms** [**-d**]*nom_paramètre***parms -c** *valeur_nom_paramètre*

Tableau 117. Commandes d'utilitaire disponibles à la fois pour UNIX et Windows (suite)

Commande	Syntaxe
release	**release -V | -U** **release** [**-s**] [*poste_de_travail***#**]*nom_ressource* [*count*]
rmstdlist	**rmstdlist -V | -U** **rmstdlist** [**-p**] [*age*]
sendevent	**sendevent -V | ? | -help | -U | -usage** **sendevent** [**-hostname** *nom_hôte*] [{**-port | -sslport**} *port*] *type_événement* *source* [[*attribute=valeur*]...] Dans les environnements dynamiques : **sendevent** [**-hostname** *nom_hôte*] [**-port** *port*] *type_événement* *source* [[*attribute=valeur*]...]
showexec	**showexec** [**-V | -U | INFO**]
ShutDownLwa	**ShutDownLwa**
StartUp	**StartUp** [**-V | -U**]
StartUpLwa	**StartUpLwa**
tws_inst_pull_info	**tws_inst_pull_info -twsuser** *ID_utilisateur* **-log_dir_base** *chemin* [**-u** | [**-run_db2_module** [y|n] | **-extract_db_defs** [y|n] | **-date** *aaaammjj*]

Commandes d'utilitaire disponibles uniquement pour le système d'exploitation UNIX

Tableau 118. Commandes d'utilitaire disponibles uniquement pour UNIX

Commande	Syntaxe
at	**at -V | -U** **at -s** *flot_travaux* | **-q***file_attentespéc-heure*
batch	**batch -V | -U** **batch** [**-s** *flot_travaux*]
showexec	**showexec** [**-V | -U | -info**]
version	**version -V | -U | -h** **version** [**-a**] [**-f** *fichierv*] [*fichier* [...]]

Commandes d'utilitaire disponibles uniquement pour le système d'exploitation Windows

Tableau 119. Commandes d'utilitaire disponibles uniquement pour Windows

Commande	Syntaxe
listproc **(NON PRIS EN CHARGE)**	**listproc**
killproc **(NON PRIS EN CHARGE)**	**killproc** *pid*
shutdown	**shutdown [-V | -U] [-appsrv]**

Commandes de génération d'états

La présente section contient une liste et la syntaxe des commandes de génération d'états et des programmes d'extraction d'états. Ces commandes sont exécutées à partir de l'invite de commande du système d'exploitation.

Commandes de génération d'états

Tableau 120. Commandes de génération d'états

Nom	Sortie produite	Syntaxe
rep1	Etat 01 - Liste détails travaux	**rep**[*x*] [**-V**|**-U**]
rep2	Etat 02 - Liste des invites	**rep**[*x*] [**-V**|**-U**]
rep3	Etat 03 - Liste des agendas des utilisateurs	**rep**[*x*] [**-V**|**-U**]
rep4a	Etat 04A - Liste des paramètres des utilisateurs	**rep**[*x*] [**-V**|**-U**]
rep4b	Etat 04B - Liste des ressources TWS	**rep**[*x*] [**-V**|**-U**]
rep7	Etat 07 - Liste des historiques des travaux	**rep7 -V**|**-U** **rep7** [**-c** *poste_de_travail*] [**-s** *nom_flot_travaux*] [**-j** *travail*] [**-f** *date* **-t** *date*] [**-l**]
rep8	Etat 08 - Histogramme des travaux	**rep8 -V**|**-U** **rep8** [**-f** *date* **-b** *heure* **-t** *date* **-e** *heure*] [**-i** *fichier*] [**-p**] **rep8** [**-b** *heure* **-e** *heure*] [**-i** *fichier*] [**-p**]
rep11	Etat 11 - Calendrier de production planifiée	**rep11 -V**|**-U** **rep11** [**-m** *mm*[*aa*] [...]] [**-c** *poste_de_travail* [...]] [**-s** *nom_flot_travaux*] [**-o** *sortie*]

Tableau 120. Commandes de génération d'états (suite)

Nom	Sortie produite	Syntaxe
reptr	Etat 09A - Récapitulatif de la production planifiée Etat 09B - Détail de la production planifiée Etat 10A - Récapitulatif de la production réelle Etat 10B - Détails de la production réelle	**reptr** [**-V**|**-U**] **reptr -pre** [**-**{**summary** | **detail**}] [*fichier_sym*] **reptr -post** [**-**{**summary** | **detail**}] [*fichier_journal*]
xref	Etat 12 - Etat de références croisées	**xref** [**-V**|**-U**] **xref** [**-cpu** *poste_de_travail*] [**-s** *nom_flot_travaux*] [**-depends**|**-files**|**-jobs**|**-prompts**|**-resource**|**-schedules**|**-when** [...]]

Programmes d'extraction d'états

Tableau 121. Programmes d'extraction d'états

Programme d'extraction	Utilisé pour générer	Syntaxe
jbxtract	Etat 01 Etat 07	**jbxtract** [**-V** | **-U**] [**-j** *travail*] [**-c** *poste_de_travail*] [**-o** *sortie*]
prxtract	Etat 02	**prxtract** [**-V** | **-U**] [**-o** *sortie*] **prxtract** [**-V** | **-U**] [**-m** *mm*[*aaaa*]] [**-c** *poste_de_travail*] [**-o** *sortie*]
caxtract	Etat 03	**caxtract** [**-V** | **-U**] [**-o** *sortie*]
paxtract	Etat 04A	**paxtract** [**-V** | **-U**] [**-o** *sortie*]
rextract	Etat 04B	**rextract** [**-V** | **-U**] [**-o** *sortie*]
r11xtr	Etat 11	**r11xtr** [**-V** | **-U**] [**-m** *mm*[*aaaa*]] [**-c** *poste_de_travail*] [**-o** *sortie*] [**-s** *nom_flot_travaux*]
xrxtrct	Etat 12	**xrxtrct** [**-V** | **-U**]

Annexe D. Définition et gestion des travaux de branche génériques

Tivoli Workload Scheduler fournit une large gamme de fonctions de planification. Vous pouvez étendre ces fonctions pour des besoins spécifiques en utilisant une solution personnalisée supplémentaire, nommée travail de branche générique.

Pour plus d'informations sur l'utilisation de cette solution, consultez les sections suivantes :

- «Introduction»
- «Exemples de scénarios», à la page 797
- «Utilisation du travail de branche», à la page 836
- «Indication des paramètres de travail de branche», à la page 840
- «Remarques importantes sur le travail de branche», à la page 852

Remarque : Le travail de branche générique n'est pas une composante native de Tivoli Workload Scheduler mais est tout de même pris en charge par le service de support logiciel IBM. Fourni avec des messages en anglais uniquement, il suit des conventions d'attribution de nom qui lui sont propres (pour de plus amples informations, voir «Placement du travail de branche dans le flot de travaux», à la page 839).

Il est vivement recommandé d'utiliser le travail de branche dans votre environnement de test préalablement. La même recommandation s'applique également pour les exemples de scénarios. C'est seulement après avoir produit une exécution sans erreur de travail de branche ou d'exemple de scénario que vous pouvez effectuer le déplacement vers l'environnement de production.

Introduction

Définir un travail de branche générique afin d'évaluer le statut ou la sortie d'un travail particulier et, selon les conditions indiquées, décider quels sont les travaux à exécuter dans le flot de travaux.

Tivoli Workload Scheduler propose des fonctions de génération de rapports, la planification basée sur les événements, et plusieurs autres fonctions. Toutefois, certaines tâches ne peuvent être exécutées qu'avec une programmation supplémentaire, tels que l'évaluation logique d'un flux de processus dans un flot de travaux.

Utilisez un travail de branche générique pour évaluer le statut ou la sortie d'un travail particulier et décidez quels sont les travaux à exécuter dans le flot de travaux. Les conditions déterminant quels sont les travaux à exécuter peuvent être très simples (par exemple, le prédécesseur s'est terminé avec l'état SUCC ou ABEND) ou être le résultat d'une opération booléenne ou arithmétique complexe.

La figure 39, à la page 792 illustre les concepts essentiels d'un travail de branche générique.

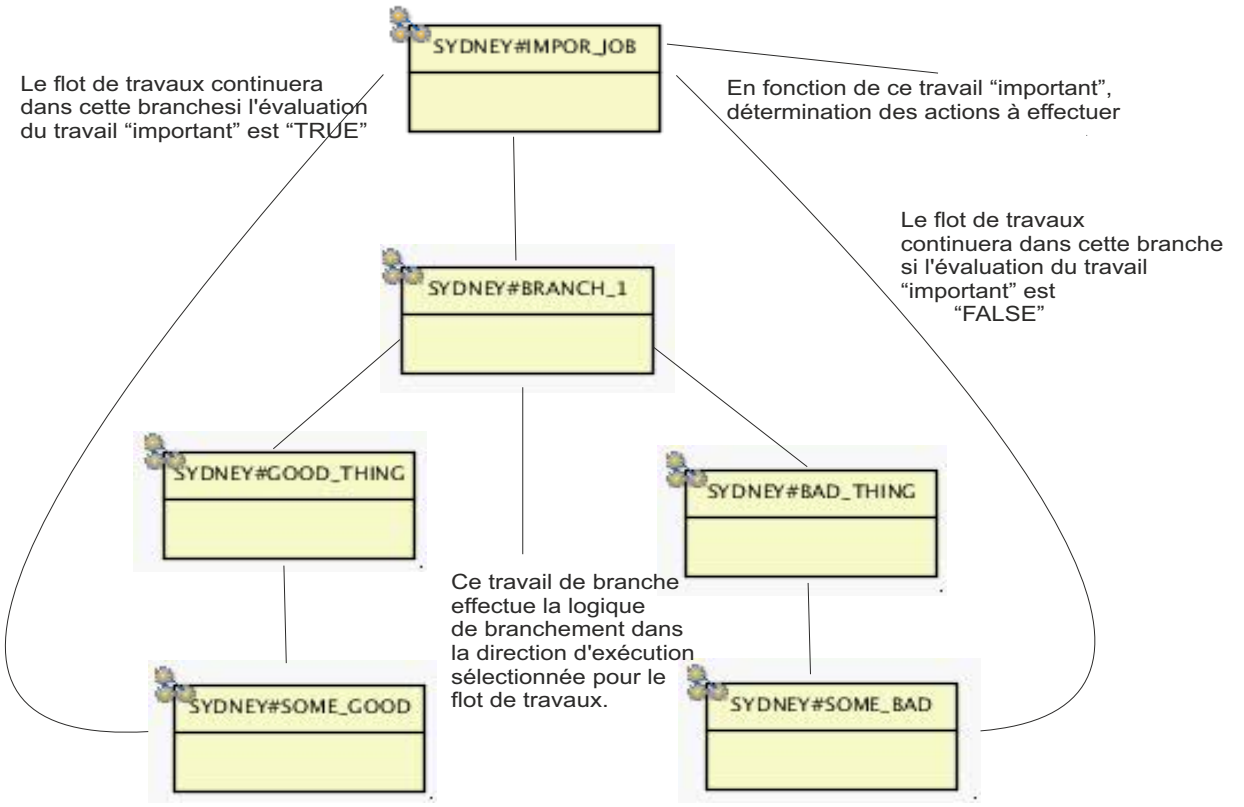


Figure 39. Objectif du travail de branche

La fonction de travail de branche générique n'est pas incluse dans Tivoli Workload Scheduler, mais a été développée par les spécialistes IBM pendant les différentes implémentations de Tivoli Workload Scheduler du client. Le travail de branche générique utilise l'interface ouverte de Tivoli Workload Scheduler.

Vous pouvez implémenter le simple branchement, selon le statut du travail prédécesseur (SUCC ou ABEND), en un temps très court. Toutefois, pour les scénarios plus complexes, par exemple rechercher un modèle et effectuer des comparaisons numériques, vous devez spécifier des paramètres d'entrée.

Terminologie

Les termes suivants permettent de décrire des travaux de branche :

Parent (parfois désigné sous le nom de travail évalué)
Travail dont vous souhaitez évaluer le statut ou d'autres propriétés.

Condition
Condition que vous exécutez sur le travail parent. La condition peut être aussi simple que *look at parent status* (examiner le statut parent) ou être plus complexe.

Enfant correct
Successeur le plus proche du travail de branche qui doit être exécuté si la condition est évaluée comme TRUE (vraie).

Enfant incorrect
Successeur le plus proche du travail de branche qui doit être stoppé si la condition est évaluée comme FALSE (fausse).

Branche
Séquence de travaux qui sont contrôlés à l'aide de la dépendance FOLLOWS.

Branche d'exécution
Branche sélectionnée à exécuter, en fonction du résultat de condition. La branche d'exécution démarre l'enfant correct si la condition est évaluée comme TRUE (vraie) ou avec l'enfant incorrect si la condition est évaluée comme FALSE (fausse).

Branche d'arrêt
Branche sélectionnée à arrêter, en fonction du résultat de condition. La branche d'arrêt s'arrête avec l'enfant incorrect si la condition est évaluée comme TRUE (vraie) ou avec l'enfant correct si la condition est évaluée comme FALSE (fausse).

Paramètre d'entrée
Argument transmis au travail spécifique de branche. Pour certains paramètres, si vous ne précisez aucune valeur, la valeur par défaut est utilisée.

Suffixe du travail de branche
Différencie plusieurs occurrences de travaux de branche dans le même flot de travaux.

La figure 40, à la page 794 illustre les termes suivants, qui sont utilisés pour définir un flot de travaux géré par un ou plusieurs travaux de branche :

- Parent
- Enfant correct et branche correcte
- Enfant incorrect et branche incorrecte
- Emplacement de la condition

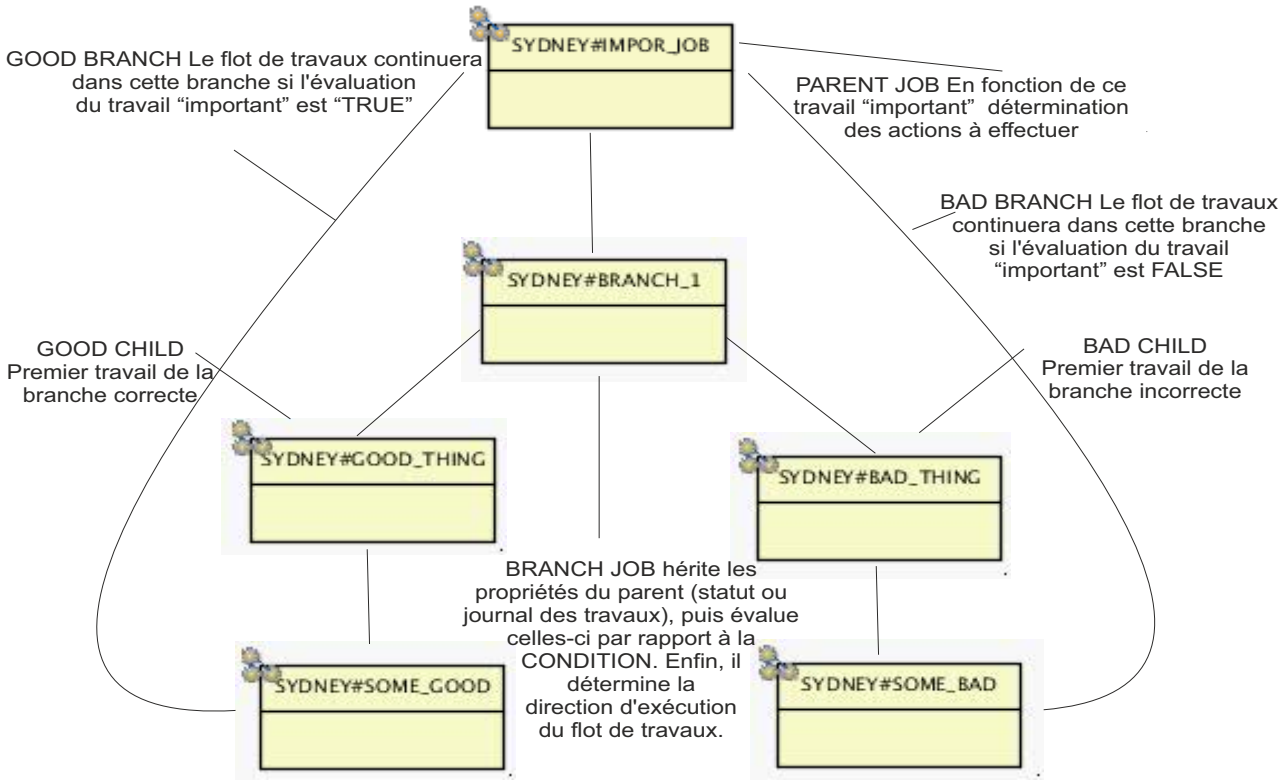


Figure 40. Termes associés à la définition de flot de travaux

La figure 41, à la page 795 illustre les termes supplémentaires qui sont utilisés pour l'exécution du flot de travaux :

- Branche d'exécution
- Branche d'arrêt

Affiche également la différence entre les termes corrects et incorrects, l'exécution et l'arrêt. Les termes sont identiques uniquement lorsque CONDITION=TRUE. Si CONDITION=FALSE, la branche d'exécution correspond à la branche incorrecte et la branche d'arrêt correspond à la branche correcte. Il s'agit du concept de logique d'évaluation de travail de branche.

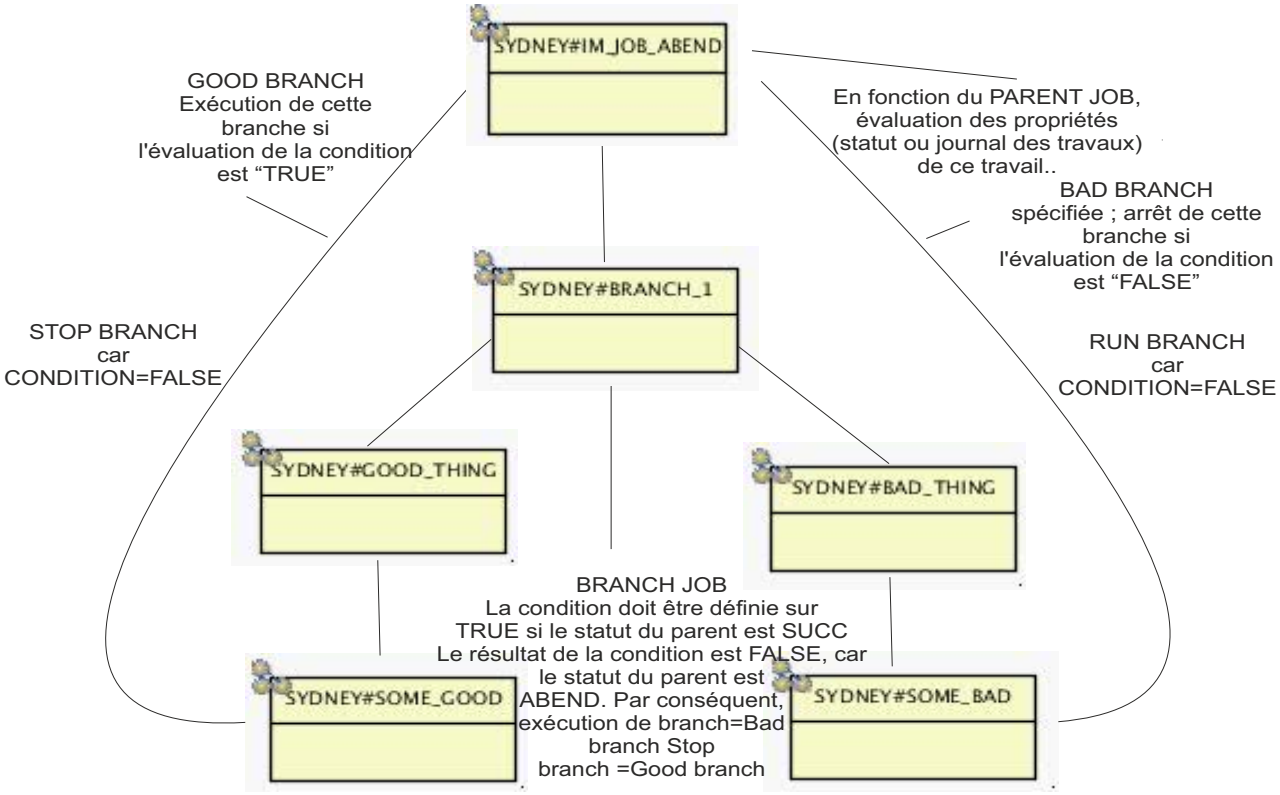


Figure 41. Termes liés à l'exécution de flot de travaux (instance de flot de travaux concrète)

Fonctions de travail de branche

Pour décrire des fonctions de travail de branche générique, le processus de branchement peut être divisé en deux processus :

- **Evaluation** : le travail de branche prend toujours son parent comme entrée. Ensuite, en fonction des paramètres spécifiés, il crée une des sous-conditions suivantes :
  - Vérifie si le parent est terminé avec l'état SUCC ou ABEND.
  - Crée une condition complexe à partir d'une ou de plusieurs sous-conditions parmi les suivantes :
    - Obtient le journal des travaux parent et recherchez une ligne comportant le modèle de texte indiqué (transmis au travail de branche comme paramètre d'entrée).
    - (Facultatif) Recherche un autre modèle sur la même ligne.
    - (Facultatif) Recherche une valeur numérique sur la même ligne. Cette valeur numérique est comparée à un nombre défini à l'aide d'un opérateur arithmétique spécifié (le numéro et l'opérateur arithmétique sont transmis au travail de branche en tant que paramètres d'entrée).

    Ces sous-conditions sont liées à l'aide des opérateurs booléens AND ou OR. La condition complexe est évaluée immédiatement.

  Le résultat de l'évaluation (même si la condition est simple ou complexe) est TRUE ou FALSE.

- **Action** : en fonction du résultat de condition, le travail de branche détermine quelle est la branche à exécuter et quelle est celle à arrêter. Il réalise alors différentes actions sur la branche d'exécution et la branche d'arrêt, comme suit.
  - Les actions possibles sur la branche d'exécution sont :

    DO NOTHING
    La branche d'exécution est exécutée.

    RELEASE
    Si le premier travail de la branche d'exécution est suspendu, il est libéré.
  - Les actions possibles sur la branche d'arrêt sont :

    CANCEL
    Tous les travaux de la branche de travail sont annulés.

    PAUSE
    Le travail de branche est interrompu. Les travaux de la branche d'arrêt ne peuvent pas s'exécuter car leur prédécesseur est mis en attente.
  - De plus, il existe une action spéciale appelée SIGNAL. Cette action écrit uniquement une recommandation pour vous invitez à confirmer le travail de branche dans le journal de travail. Vous créez un travail de branche SIGNAL en définissant l'indicateur `Confirmation requise`.

    Pour plus d'informations sur le travail de branche SIGNAL, voir scénario d'action signal

Vous pouvez combiner les critères d'évaluation et les actions consécutives à votre gré.

Avantages du travail de branche

Les principaux points forts du travail de branche générique sont :

- Un seul travail de branche est défini dans la base de données Tivoli Workload Scheduler.

  Le travail de branche générique est représenté par une définition de travail qui pointe vers un script de shell. Le script ne prend aucun argument de ligne de commande qui pointe vers le parent et les enfants du travail de branche. Les informations sur le parent (prédécesseur) et les enfants (successeurs) sont évalués automatiquement.
- Il n'est pas nécessaire d'indiquer des paramètres d'entrée si vous utilisez la plupart des scénarios communs de travail de branche.

  Vous n'avez pas à spécifier de paramètres lors de l'évaluation de l'état du résultat du travail parent (SUCC ou ABEND). Vous insérez le travail de branche dans le flot de travaux, vous placez les dépendances de prédécesseur/successeur et attribuez aux travaux enfants les noms spécifiques qui identifient l'enfant correct et l'enfant incorrect.
- Le travail de branche renvoie un journal de travail structuré qui contient des informations détaillées sur l'environnement du travail de branche, les paramètres d'entrée, la condition évaluée, les actions exécutées. En lisant le journal de travail, vous pouvez facilement contrôler les activités que le travail de branche a réalisées.
- Le travail de branche utilise la représentation des objets Tivoli Workload Scheduler dans le plan en cours. Cela signifie que tous les objets du plan (par

exemple, les travaux et flots de travaux) sont référencé à l'aide du mot clé `schedid`. Assurez-vous que toutes les actions lancées sur les objets du pointent vers un objet unique.

Cette fonction est optimisée dans toutes les occurrences de flot de travaux dans le plan courant :

- Toute occurrence du flot de travaux soumise sans spécifier d'alias.
- Toute occurrence du flot de travaux soumise en spécifiant l'alias.
- Le travail de branche générique s'exécute sur les deux gestionnaires de domaines maître UNIX et Windows. Etant donné que le travail de branche est consigné dans le script de shell (par conséquent il s'exécute en mode natif sous UNIX), sur les systèmes d'exploitation Windows vous devez utiliser un interpréteur de shell UNIX.
- Le travail de branche générique s'exécute même si le flot de travaux est défini sur la classe de poste de travail. Le travail de branche lui-même doit être défini sur le gestionnaire de domaine maître.

Exemples de scénarios

Les exemples de scénarios décrivent les différents travaux de branche, les concepts principaux et l'utilisation des travaux de branche.

Le nom des vues possède la structure suivante :

- Utilisation du scénario.
- Définition d'exemple de flot de travaux.
- Journal du travail de branche générique.
- Paramètres requis pour le branchement.
- Positionnement du travail dans le flot de travaux et modification des noms de travaux enfant, si nécessaire.

Un travail générique de branche basé sur le type de condition diffère d'un travail générique de branche basé sur le type d'action, comme suit :

Condition
Indique les critères afin de déterminer la branche d'exécution (travaux qui doivent continuer) et la branche d'arrêt (travaux qui doivent arrêter).

Action
Indique l'action à effectuer sur la branche d'exécution et la branche d'arrêt. Vous pouvez effectuer deux types d'action sur la branche d'arrêt et deux types d'actions sur la branche d'exécution. De plus, il existe une action spéciale de *signalement*.

Scénarios basés sur le type de condition

Utilisez un travail générique de branche en fonction du type de condition pour indiquer les critères qui déterminent la branche d'exécution et la branche d'arrêt.

Pour les scénarios basés sur le type d'action, voir la rubrique consacrée à ces scénarios.

Scénario de branche simple

Utilisez le branchement simple pour évaluer le statut d'un travail.

Utilisation de branche simple

Selon le statut renvoyé par le travail que vous évaluez, une branche spécifique s'exécute : si le travail se termine à l'état SUCC, la branche correcte s'exécute ; si le travail se termine de façon anormale, la branche incorrecte s'exécute.

Branche simple se terminant avec l'état SUCC

La figure 42 affiche la définition de flot de travaux.

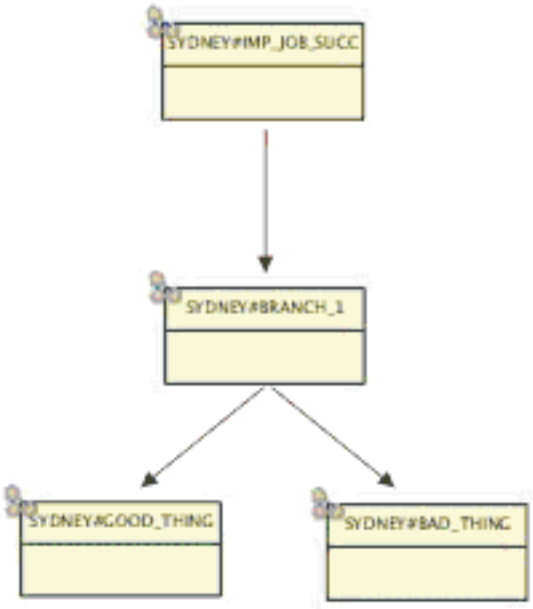


Figure 42. Définition de branche simple (SUCC)

Si le travail parent se termine à l'état SUCC, l'enfant incorrect est annulé et l'enfant correct s'exécute. La figure 43 affiche l'état final du flot de travaux dans le plan courant.

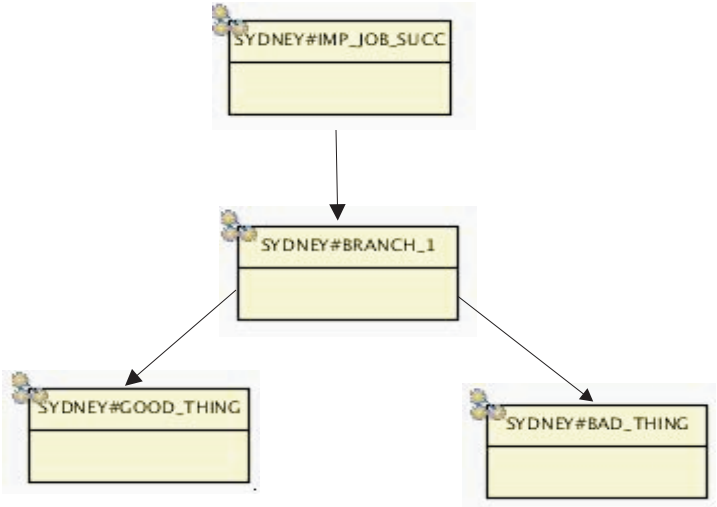


Figure 43. Statut final de branche simple (SUCC)

Fin de branche simple à l'état ABEND

La figure 44 affiche la définition de flot de travaux.

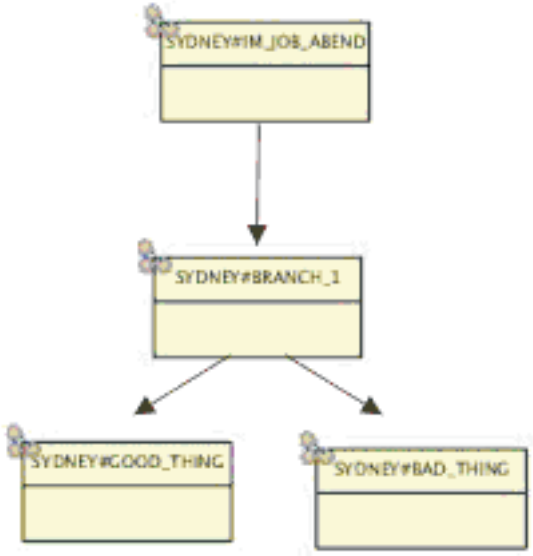


Figure 44. Définition de branche simple (ABEND)

Le travail évalué (c'est-à-dire, le parent) doit avoir l'option de reprise définie sur `Continuer`, sinon le travail arrêté de manière anormale ne libère pas le travail de branche de la dépendance FOLLOWS et le travail se termine à l 'état STUCK. Cela concerne *tous* les travaux parents pour lesquels vous évaluez l'état du résultat, car vous ne devez pas oublier que n'importe quel travail peut se terminer de façon anormale.

L'exemple suivant affiche le journal de l'instance de travail de branche générique :

```
============ START of branch job ===================
============== Job environment ====================
MASTER_PLATFORM=UNIX
STREAM_NAME=SIMPLE_BRANCH_B
PARAMETER_PREFIX=
JOB_NAME=BRANCH_1
BRANCH_SUFFIX=1
PARENT=IMPORTANT_JOB_ABEND
================================================
============= Input parameters =============
INPUT_SWITCH=PARENT_SUCCESS
ACTION_SWITCH=CANCEL
CONDITION_COUNT=0
================================================
============= MAIN DECISION MAKING =================
Evaluation dependent on PARENT_SUCCESS
FALSE: Searched for SUCC parent.
Status of PARENT JOB(IMPORTANT_JOB_ABEND) is ABEND.
================================================
============= Action on STOP Branch ===============
Performing action CANCEL on job SYDNEY#0AAAAAAAAAAAAEC4.G_DO_THE_GOOD_THING
%cj SYDNEY#0AAAAAAAAAAAAEC4.G_DO_THE_GOOD_THING;schedid;noask
Command forwarded to batchman for SYDNEY#GBJ_LONG_ABEND[(2159 12/16/07),
   (0AAAAAAAAAAAAEC4)].G_DO_THE_GOOD_THING
Performing action CANCEL on job SYDNEY#0AAAAAAAAAAAAEC4.SOME_GOOD_1
%cj SYDNEY#0AAAAAAAAAAAAEC4.SOME_GOOD_1;schedid;noask
Command forwarded to batchman for SYDNEY#GBJ_LONG_ABEND[(2159 12/16/07),
   (0AAAAAAAAAAAAEC4)].SOME_GOOD_1
```

```
Performing action CANCEL on job SYDNEY#0AAAAAAAAAAAAEC4.SOME_GOOD_2
%cj SYDNEY#0AAAAAAAAAAAAEC4.SOME_GOOD_2;schedid;noask
Command forwarded to batchman for SYDNEY#GBJ_LONG_ABEND[(2159 12/16/07),
   (0AAAAAAAAAAAAEC4)].SOME_GOOD_2
=================================================
============= Action on RUN Branch ================
Performing action RELEASE on job SYDNEY#0AAAAAAAAAAAAEC4.B_DO_THE_BAD_THING
Releasing of job SYDNEY#0AAAAAAAAAAAAEC4.B_DO_THE_BAD_THING is NOT NECESSARY,
   because priority=10
=========================================================
============= Statistics of branch job BRANCH_1 =============
FALSE: Searched for SUCC parent. Status of PARENT JOB(IMPORTANT_JOB_ABEND) is ABEND.
For action CANCEL-RUN_BRANCH=B_DO_THE_BAD_THING and STOP_BRANCH=G_DO_THE_GOOD_THING
BRANCH selected to STOP: G_DO_THE_GOOD_THING
BRANCH selected to CONTINUE: B_DO_THE_BAD_THING
CANCELED_JOBS: G_DO_THE_GOOD_THING, SOME_GOOD_1, SOME_GOOD_2
PAUSED_JOB:
RELEASED_JOB:
============ END of branch job BRANCH_1 ==========
```

Paramètres d'entrée obligatoires

Le branchement basé sur l'évaluation du statut du travail parent est l'utilisation la plus courante du travail générique de branche ; il ne nécessite *aucun* paramètre d'entrée.

Placement du travail de branche dans le flot de travaux

Insérez le travail de branche générique dans le flot de travaux après le travail parent et renommez l'enfant correct avec le préfixe "G_" et l'enfant incorrect avec le préfixe "B_".

Il est également recommandé de renommer le travail de branche avec un suffixe constitué du caractère de soulignement et d'une valeur numérique. Un nom classique pour le premier travail de branche dans un flot de travaux est BRANCH_1.

Scénario de branche longue

le branchement long est l'utilisation récursive du branchement simple.

Utilisation de branche longue

Le but principal du scénario de branche longue est de montrer que le travail de branche générique peut annuler *tous* les travaux dans la branche d'arrêt, même s'il existe une arborescence entière de travaux d'annulation.

Cette fonction est nécessaire car, dans Tivoli Workload Scheduler, si un travail est annulé, tous les successeurs du travail sont libérés de la dépendance de prédécesseur/successeur, ce qui implique que les travaux qui sont *dépendants* du travail *annulé* démarrent immédiatement. Puisque ce comportement peut être non souhaité dans certains cas, le travail de branche générique annule le premier travail d'arrêt et *tous* ses successeurs.

Remarque : Du point de vue de programmation, le travail générique de branche utilise des appels de fonction récursifs pour parcourir tous les successeurs du premier enfant du travail de branche générique.

Fin de branche longue à l'état SUCC

La figure 45 présente le flot de travaux contenant la structure complexe des successeurs possibles, dans la branche correcte ou incorrecte.

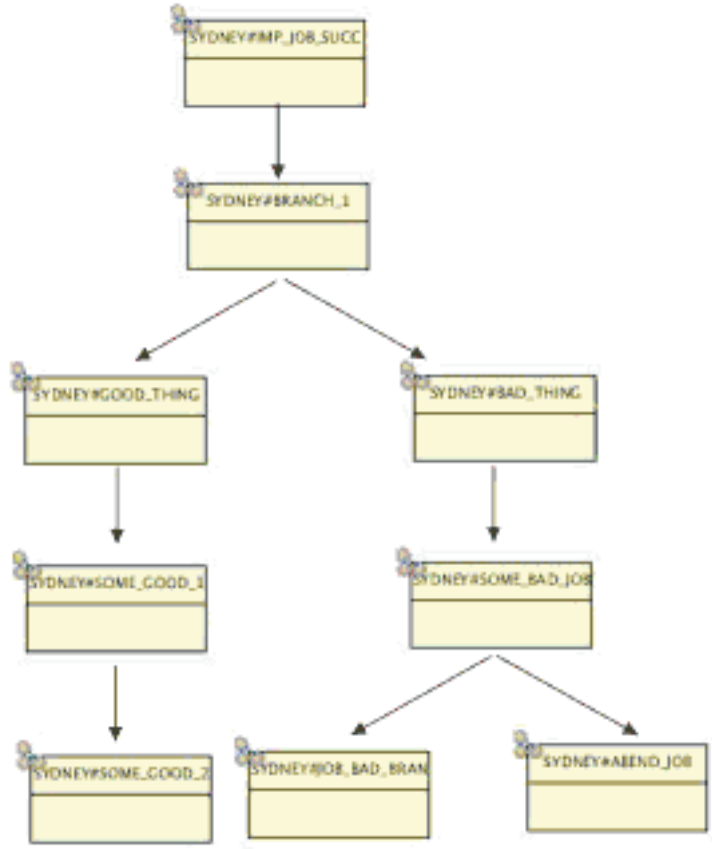


Figure 45. Définition de branche longue (SUCC)

Si le parent s'est terminé à l'état SUCC, le journal de travail montre la sortie suivante :

```
============ START of branch job BRANCH_1 ==========
====================== Job environment =============
MASTER_PLATFORM=UNIX
STREAM_NAME=GBJ_LONG_SUCC
STREAM_CPU=SYDNEY
BRANCH_JOB_NAME=BRANCH_1
PARENT=IMPORTANT_JOB_SUCC
============================================================
============= Input parameters =============
CONDITION_SWITCH=PARENT_SUCCESS
ACTION_SWITCH=CANCEL
============================================================
============= MAIN DECISION MAKING =============
Evaluation dependent on PARENT_SUCCESS
TRUE: Searched for SUCC parent. Status of PARENT JOB(IMPORTANT_JOB_SUCC) is SUCC
============================================================
============= Action on STOP Branch =============
Performing action CANCEL on job SYDNEY#0AAAAAAAAAAAAEC3.B_DO_THE_BAD_THING
%cj SYDNEY#0AAAAAAAAAAAAEC3.B_DO_THE_BAD_THING;schedid;noask
Command forwarded to batchman for SYDNEY#GBJ_LONG_SUCC[(2154 12/16/07),
   (0AAAAAAAAAAAAEC3)].B_DO_THE_BAD_THING
Performing action CANCEL on job SYDNEY#0AAAAAAAAAAAAEC3.SOME_BAD_JOB
%cj SYDNEY#0AAAAAAAAAAAAEC3.SOME_BAD_JOB;schedid;noask
Command forwarded to batchman for SYDNEY#GBJ_LONG_SUCC[(2154 12/16/07),
```

```
   (0AAAAAAAAAAAAEC3)].SOME_BAD_JOB
Performing action CANCEL on job SYDNEY#0AAAAAAAAAAAAEC3.ABEND_JOB
%cj SYDNEY#0AAAAAAAAAAAAEC3.ABEND_JOB;schedid;noask
Command forwarded to batchman for SYDNEY#GBJ_LONG_SUCC[(2154 12/16/07),
   (0AAAAAAAAAAAAEC3)].ABEND_JOB
Performing action CANCEL on job SYDNEY#0AAAAAAAAAAAAEC3.ANOTHER_JOB_IN_BAD_BRANCH
%cj SYDNEY#0AAAAAAAAAAAAEC3.ANOTHER_JOB_IN_BAD_BRANCH;schedid;noask
Command forwarded to batchman for SYDNEY#GBJ_LONG_SUCC[(2154 12/16/07),
   (0AAAAAAAAAAAAEC3)].ANOTHER_JOB_IN_BAD_BRANCH
============================================================
============= Action on RUN Branch =============
Performing action RELEASE on job SYDNEY#0AAAAAAAAAAAAEC3.G_DO_THE_GOOD_THING
Releasing of job SYDNEY#0AAAAAAAAAAAAEC3.G_DO_THE_GOOD_THING is NOT NECESSARY,
   because priority=10
============================================================
============= Statistics of branch job BRANCH_1 =============
TRUE: Searched for SUCC parent. Status of PARENT JOB(IMPORTANT_JOB_SUCC) is SUCC.
For action CANCEL-RUN_BRANCH=G_DO_THE_GOOD_THING and STOP_BRANCH=B_DO_THE_BAD_THING
BRANCH selected to STOP: B_DO_THE_BAD_THING
BRANCH selected to CONTINUE: G_DO_THE_GOOD_THING
CANCELED_JOBS: B_DO_THE_BAD_THING, SOME_BAD_JOB, ABEND_JOB, ANOTHER_JOB_IN_BAD_BRANCH
PAUSED_JOB:
RELEASED_JOB:
============ END of branch job BRANCH_1 ==========
```

Fin de branche longue à l'état ABEND

La figure 46, à la page 803 montre comment le flot de travaux est exécuté si le travail évalué s'est arrêté de manière anormale.

Remarque : Pour libérer les successeurs de la dépendance de prédécesseur/successeur, le travail évalué (travail parent) doit avoir l'option de reprise définie sur `Continue`. Vous pouvez définir ce paramètre uniquement dans la définition de travail, non dans la définition de flot de travaux.

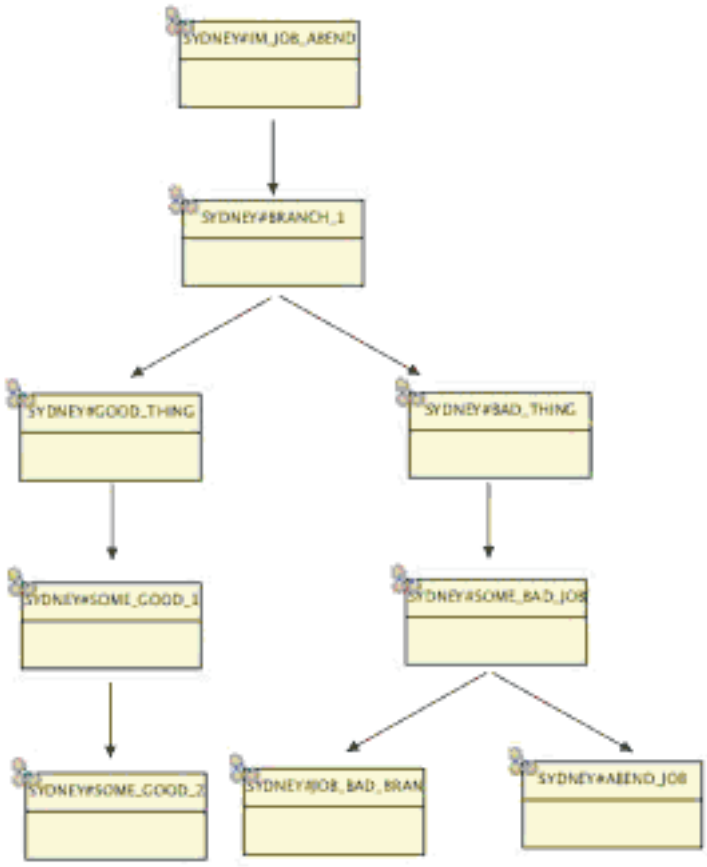


Figure 46. Statut final de branche longue(ABEND)

Le journal de travail affiche la sortie de l'instance de travail de branche générique :

```
============ START of branch job BRANCH_1 ==========
====================== Job environment =============
MASTER_PLATFORM=UNIX
STREAM_NAME=GBJ_LONG_ABEND
STREAM_CPU=SYDNEY
BRANCH_JOB_NAME=BRANCH_1
PARENT=IMPORTANT_JOB_ABEND
============================================================
============= Input parameters =============
CONDITION_SWITCH=PARENT_SUCCESS
ACTION_SWITCH=CANCEL
============================================================
============= MAIN DECISION MAKING =============
Evaluation dependent on PARENT_SUCCESS
FALSE: Searched for SUCC parent.
Status of PARENT JOB(IMPORTANT_JOB_ABEND) is ABEND.
============================================================
============= Action on STOP Branch =============
Performing action CANCEL on job SYDNEY#0AAAAAAAAAAAAEC4.G_DO_THE_GOOD_THING
%cj SYDNEY#0AAAAAAAAAAAAEC4.G_DO_THE_GOOD_THING;schedid;noask
Command forwarded to batchman for SYDNEY#GBJ_LONG_ABEND[(2159 12/16/07),
   (0AAAAAAAAAAAAEC4)].G_DO_THE_GOOD_THING
Performing action CANCEL on job SYDNEY#0AAAAAAAAAAAAEC4.SOME_GOOD_1
%cj SYDNEY#0AAAAAAAAAAAAEC4.SOME_GOOD_1;schedid;noask
Command forwarded to batchman for SYDNEY#GBJ_LONG_ABEND[(2159 12/16/07),
   (0AAAAAAAAAAAAEC4)].SOME_GOOD_1
Performing action CANCEL on job SYDNEY#0AAAAAAAAAAAAEC4.SOME_GOOD_2
%cj SYDNEY#0AAAAAAAAAAAAEC4.SOME_GOOD_2;schedid;noask
Command forwarded to batchman for SYDNEY#GBJ_LONG_ABEND[(2159 12/16/07),
   (0AAAAAAAAAAAAEC4)].SOME_GOOD_2
```

```
============================================================
============= Action on RUN Branch =============
Performing action RELEASE on job SYDNEY#0AAAAAAAAAAAAEC4.B_DO_THE_BAD_THING
Releasing of job SYDNEY#0AAAAAAAAAAAAEC4.B_DO_THE_BAD_THING is NOT NECESSARY,
   because priority=10
============================================================
============= Statistics of branch job BRANCH_1 =============
FALSE: Searched for SUCC parent. Status of PARENT JOB(IMPORTANT_JOB_ABEND) is ABEND.
For action CANCEL-RUN_BRANCH=B_DO_THE_BAD_THING and STOP_BRANCH=G_DO_THE_GOOD_THING
BRANCH selected to STOP: G_DO_THE_GOOD_THING
BRANCH selected to CONTINUE: B_DO_THE_BAD_THING
CANCELED_JOBS: G_DO_THE_GOOD_THING, SOME_GOOD_1, SOME_GOOD_2
PAUSED_JOB:
RELEASED_JOB:
============ END of branch job BRANCH_1 ==========
```

Paramètres d'entrée obligatoires

Ce type de branchement ne nécessite aucun paramètre d'entrée.

Placement du travail de branche dans le flot de travaux

Insérez le travail de branche générique dans le flot de travaux après le travail parent et renommez l'enfant correct avec le préfixe "G_" et l'enfant incorrect avec le préfixe "B_".

Il est également recommandé de renommer le travail de branche avec un suffixe constitué du caractère de soulignement et d'une valeur numérique. Un nom classique pour le premier travail de branche dans un flot de travaux est BRANCH_1.

Branches multiples

Utilisez la fonction de branche multiple pour disposer de travaux de branche différents dans un même flot de travaux.

Utilisation de branches multiples

Le branchement multiple permet de disposer de plusieurs travaux de branche dans un même flot de travaux. La figure 47, à la page 805 présente un exemple d'un flot de travaux géré par les travaux de branches multiples.

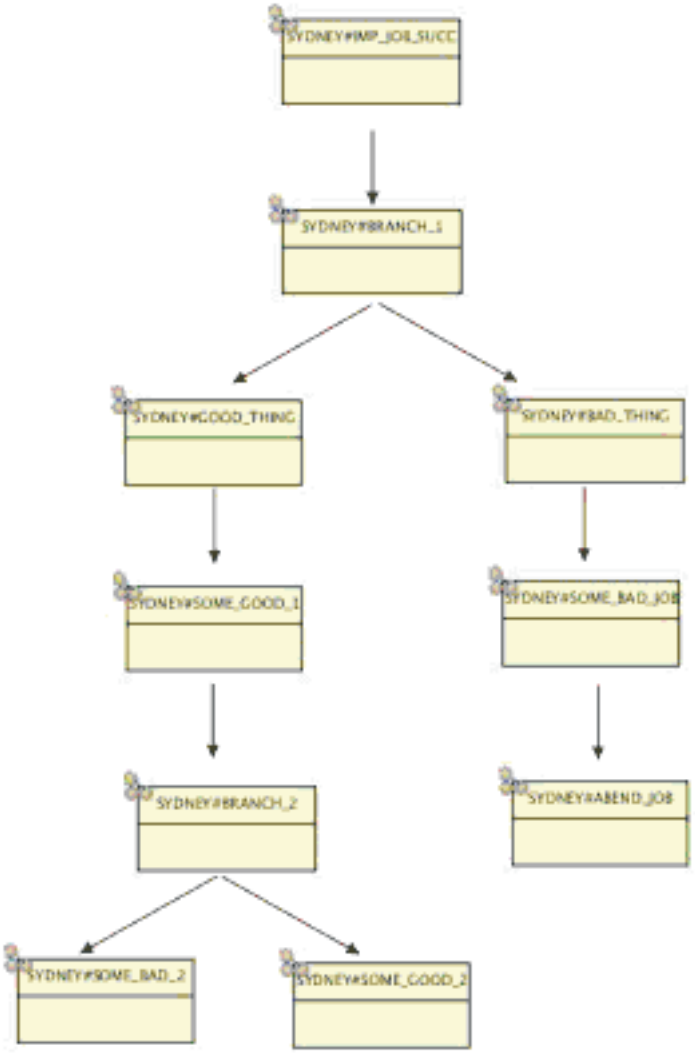


Figure 47. Travaux de branches multiples dans un même flot de travaux

Paramètres d'entrée obligatoires

Il peut s'avérer nécessaire de définir les paramètres d'entrée en fonction des travaux de branche utilisés. Vous n'êtes pas obligé d'indiquer des paramètres pour les travaux basés sur les scénarios suivants :

- Branche simple
- Branche longue

Indiquez les paramètres d'entrée pour les travaux basés sur les scénarios suivants :

- Branche complexe - modèle
- Branche complexe - modèle dans la ligne de modèle
- Branche complexe - comparaison d'une valeur numérique
- Scénario complexe - conditions multiples
- Scénario d'actions de pause/libération
- Scénario d'action de signal

Placement des travaux de branche dans le flot de travaux

Pour chaque travail de branche, insérez le travail de branche générique dans le flot de travaux après le travail parent et renommez l'enfant correct avec le préfixe "G_" et l'enfant incorrect avec le préfixe "B_". Distinguez les travaux de branche différents par *suffixe de branche*, qui doit comprendre un *caractère de soulignement* et un *numéro*.

La meilleure pratique consiste à créer le suffixe de branche à l'aide de numéros dans l'ordre croissant, de sorte que les travaux de branche multiples dans un flot de travaux sont nommés, par exemple, BRANCH_1, BRANCH_2, et ainsi de suite.

Fin anormale du parent

Le scénario de fin anormale du parent représente le cas inversé du scénario de branche simple.

Utilisation de fin anormale du parent

Le scénario de fin anormale du parent décrit les exigences fonctionnelles suivantes :

- Obtention du statut du travail parent.
- Si le statut du travail parent est SUCC, il est considéré comme INCORRECT.
- Si le statut du parent est ABEND, il est considéré comme CORRECT.

Cette section décrit uniquement le cas où le travail parent se termine à l'état SUCC.

Fin de fin anormale du parent à l'état SUCC

La figure 48, à la page 807 présente le flot de travaux avec la fonction PARENT_ABEND.

Le flot de travaux ressemble à la branche simple, la seule différence est que vous avez spécifié le paramètre CONDITION_SWITCH=PARENT_ABEND pour le travail de branche.

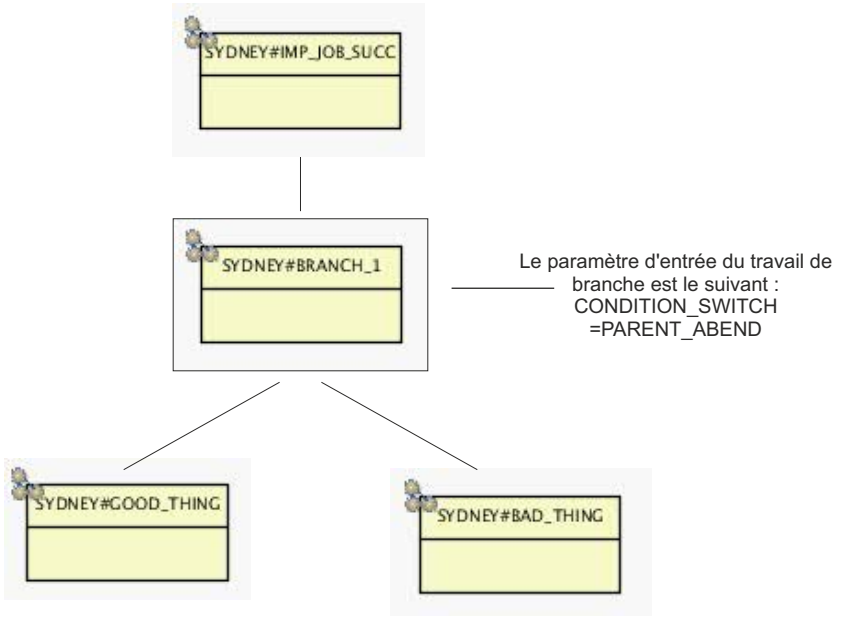


Figure 48. Définition de fin anormale de parent (SUCC)

Le journal de travail affiche la sortie de l'instance de travail de branche générique :

```
============ START of branch job BRANCH_1 ==========
================ Job environment ====================
MASTER_PLATFORM=UNIX
STREAM_NAME=GBJ_PARENT_ABEND
STREAM_CPU=SYDNEY
BRANCH_JOB_NAME=BRANCH_1
PARENT=IMPORTANT_JOB_SUCC
============================================================
============= Input parameters =============
CONDITION_SWITCH=PARENT_ABEND
ACTION_SWITCH=CANCEL
============================================================
============= MAIN DECISION MAKING =============
Evaluation dependent on PARENT_ABEND
FALSE: Searched for ABEND parent.
Status of PARENT JOB(IMPORTANT_JOB_SUCC) is SUCC.
============================================================
============= Action on STOP Branch =============
Performing action CANCEL on job SYDNEY#0AAAAAAAAAAAAEDA.G_DO_THE_GOOD_THING
%cj SYDNEY#0AAAAAAAAAAAAEDA.G_DO_THE_GOOD_THING;schedid;noask
Command forwarded to batchman for SYDNEY#GBJ_PARENT_ABEND[(2314 12/16/07),
   (0AAAAAAAAAAAAEDA)].G_DO_THE_GOOD_THING
============================================================
============= Action on RUN Branch =============
Performing action RELEASE on job SYDNEY#0AAAAAAAAAAAAEDA.B_DO_THE_BAD_THING
Releasing of job SYDNEY#0AAAAAAAAAAAAEDA.B_DO_THE_BAD_THING is NOT NECESSARY,
because priority=10
============================================================
============= Statistics of branch job BRANCH_1 =============
FALSE: Searched for ABEND parent. Status of PARENT JOB(IMPORTANT_JOB_SUCC) is SUCC
For action CANCEL - RUN_BRANCH=B_DO_THE_BAD_THING and
   STOP_BRANCH=G_DO_THE_GOOD_THING
BRANCH selected to STOP: G_DO_THE_GOOD_THING
BRANCH selected to CONTINUE: B_DO_THE_BAD_THING
CANCELED_JOBS: G_DO_THE_GOOD_THING
PAUSED_JOB:
RELEASED_JOB:
============ END of branch job BRANCH_1 ==========
```

Paramètres d'entrée obligatoires

Le tableau 122 affiche les paramètres requis pour le scénario de branche négatif.

Tableau 122. Paramètres d'entrée pour le scénario de travail de branche négatif

Nom du paramètre	Valeur du paramètre
CONDITION_SWITCH	PARENT_ABEND

L'exemple suivant illustre la définition de paramètre. Le texte est saisi dans la zone de commentaires de la définition du flot de travaux.

```
BRANCH_1-BEGIN
CONDITION_SWITCH=PARENT_ABEND
BRANCH_1-END
```

Pour plus d'informations sur la définition des paramètres de travail de branche, voir «Indication des paramètres de travail de branche», à la page 840.

Placement du travail de branche dans le flot de travaux

Insérez le travail de branche générique dans le flot de travaux après le travail parent et renommez l'enfant correct avec le préfixe "G_" et l'enfant incorrect avec le préfixe "B_".

Il est également recommandé de renommer le travail de branche avec un suffixe constitué du caractère de soulignement et d'une valeur numérique. Un nom classique pour le premier travail de branche dans un flot de travaux est BRANCH_1.

Scénarios de branche complexe

les scénarios de branche complexe affichent la flexibilité du travail de branche générique.

Les scénarios précédents ont été basés sur l'évaluation du *statut* du journal parent. Les scénarios de travail de branche complexe sont basés sur l'évaluation du *journal de travail* du travail parent. Pour obtenir une liste des conditions complexes, voir terminologie.

Une condition complexe peut comprendre plusieurs sous-conditions. Par exemple :

1. sous-condition 1
   - Rechercher le modèle de texte (indiqué comme paramètre) dans le journal du travail parent.
   - Isoler la valeur numérique dans la ligne sur laquelle le modèle est trouvé.
   - Comparer la valeur numérique avec un nombre spécifique. Pour cette comparaison, utiliser l'opérateur arithmétique indiqué (indiqué comme paramètre). Si la comparaison arithmétique aboutit, elle renvoie TRUE.

   Par exemple, recherchez la ligne contenant le modèle `Disque disponible`. Si vous la trouvez, recherchez un nombre dans la ligne. Si ce nombre est supérieur à 90, renvoyez TRUE.
2. sous-condition 2
   a. Rechercher un autre modèle de texte dans le journal du travail parent.
   b. Si le modèle est détecté, annuler le résultat et renvoyer FALSE (par exemple, renvoyer FALSE si le journal du travail parent contient le modèle `Erreur`).

3. Connecter les résultats de la première et de la deuxième sous-condition avec un opérateur booléen (l'opérateur booléen est indiqué comme paramètre, les valeurs possibles sont AND et OR).
4. Evaluer le résultat final de toute la condition (TRUE ou FALSE).

La condition définie peut être très flexible et peut couvrir de nombreuses situations standard.

Les sections suivantes décrivent plusieurs scénarios de branche complexe pour expliquer comment :

- Rechercher un modèle spécifique de texte dans le journal du travail parent.
- Rechercher un modèle spécifique de texte dans le journal du travail parent. Rechercher un modèle de texte supplémentaire dans la même ligne si le modèle est trouvé.
- Rechercher un modèle spécifique de texte dans le journal du travail parent. Rechercher une valeur numérique dans la même ligne si le modèle est trouvé. Si la valeur est trouvée, la comparer avec le numéro indiqué à l'aide de l'opérateur arithmétique fourni.
- Combiner plusieurs conditions en une en utilisant les opérateurs booléens AND et OR.

Branche complexe - modèle

Utilisez le scénario de modèle de branche complexe pour rechercher un modèle de texte spécifique dans le journal de travail parent.

Utilisation du scénario de modèle

Vous pouvez rechercher n'importe quelle chaîne, par exemple `terminé correctement` ou `TOUTES LES unités de bande montées`. Généralement le texte doit représenter le message positif inclus dans le journal du travail parent.

Remarque : La recherche de modèle recherche généralement le message positif du journal de travail parent. Dans certains cas, vous pouvez être amené à implémenter la logique inversée, par exemple rechercher le modèle `Erreur`, `sans succès` ou `Espace insuffisant`, qui représentent les messages négatifs du journal. Pour utiliser cette méthode, voir «Branche complexe - modèle refusé», à la page 811.

A l'aide de la branche de modèle, si le modèle de texte est détecté, alors CONDITION=TRUE sinon CONDITION=FALSE.

La figure 49, à la page 810 affiche la définition du flot de travaux pour le scénario de branche de modèle. Le flot de travaux est identique au simple scénario de branche, mais les paramètres supplémentaires sont définis :

- CONDITION_SWITCH=COMPLEX
- PATTERN_1=completed successfully

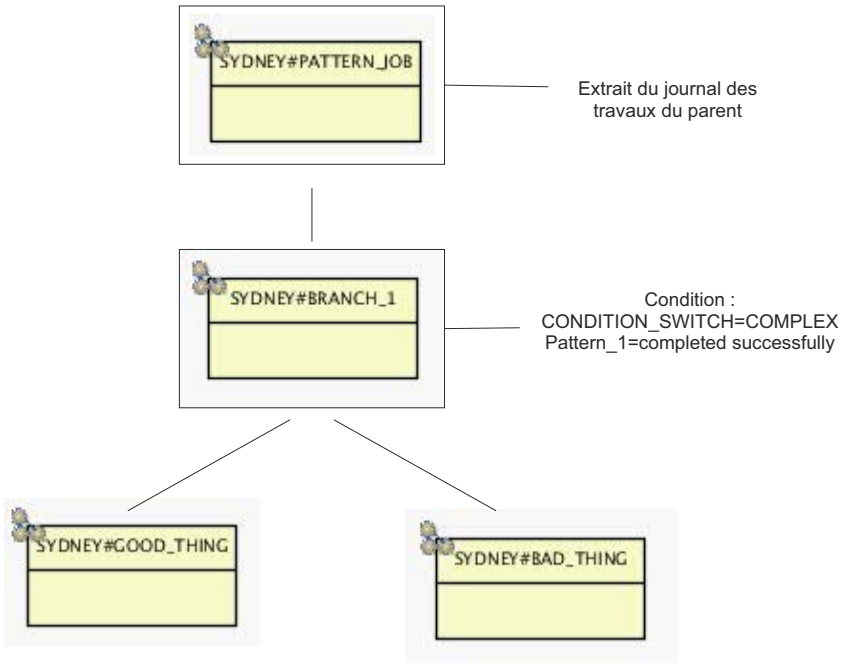


Figure 49. Scénario de modèle - définition

Le journal des travaux du scénario de modèle affiche la sortie de l'instance de travail de branche générique :

```
============ START of branch job BRANCH_1 ==========
================== Job environment =================
MASTER_PLATFORM=UNIX
STREAM_NAME=GBJ_PATTERN
STREAM_CPU=SYDNEY
BRANCH_JOB_NAME=BRANCH_1
PARENT=PATTERN_JOB
=========================================================
============= Input parameters =============
CONDITION_SWITCH=COMPLEX
ACTION_SWITCH=CANCEL
CONDITION_COUNT=1
PATTERN[1]=completed successfully
IS_CASE_SENSITIVE[1]=YES
IS_REGULAR_EXPRESSION[1]=NO
NEGATE_CONDITION_RESULT[1]=NO
=========================================================
============= MAIN DECISION MAKING =============
COMPLEX condition evaluation
----------ATOMIC CONDITION 1------------
Searching for "completed successfully" in JOBLOG of PATTERN_JOB
Pattern FOUND, performing further tests.
No additional value defined for specified pattern.
Condition evaluated as TRUE.
ATOMIC CONDITION RESULT [1]= TRUE
-------------------------------------
-------- COMPLEX CONDITION ----------
[ TRUE ]
CONDITION_RESULT=TRUE
TRUE: The result of complex condition is TRUE.
=============================================================
============= Action on STOP Branch =============
Performing action CANCEL on job SYDNEY#0AAAAAAAAAAAAED4.B_DO_THE_BAD_THING
%cj SYDNEY#0AAAAAAAAAAAAED4.B_DO_THE_BAD_THING;schedid;noask
Command forwarded to batchman for SYDNEY#GBJ_PATTERN[(1733 12/17/07),
   (0AAAAAAAAAAAAED4)].B_DO_THE_BAD_THING
=========================================================
============= Action on RUN Branch =============
```

```
Performing action RELEASE on job SYDNEY#0AAAAAAAAAAAED4.G_DO_THE_GOOD_THING
Releasing of job SYDNEY#0AAAAAAAAAAAED4.G_DO_THE_GOOD_THING is NOT NECESSARY,
   because priority=10
============================================================
============= Statistics of branch job BRANCH_1 =============
TRUE: The result of complex condition is TRUE.
For action CANCEL - RUN_BRANCH=G_DO_THE_GOOD_THING and
   STOP_BRANCH=B_DO_THE_BAD_THING
BRANCH selected to STOP: B_DO_THE_BAD_THING
BRANCH selected to CONTINUE: G_DO_THE_GOOD_THING
CANCELED_JOBS: B_DO_THE_BAD_THING
PAUSED_JOB:
RELEASED_JOB:
============ END of branch job BRANCH_1 ==========
```

Paramètres d'entrée obligatoires

Le tableau 123 affiche les paramètres requis pour le scénario de modèle.

Tableau 123. Paramètres d'entrée pour le scénario de travail de modèle

Nom du paramètre	Valeur du paramètre
CONDITION_SWITCH	COMPLEXE
PATTERN_1	Completed successfully (Terminé correctement)

L'exemple suivant illustre la définition de paramètre. Le texte est saisi dans la zone de commentaires de la définition du flot de travaux.

```
BRANCH_1-BEGIN
CONDITION_SWITCH=COMPLEX
PATTERN_1=completed successfully
BRANCH_1-END
```

Pour une description sur la spécification des paramètres du travail de branche, voir Gestion des paramètres de travail de branche.

Placement du travail de branche dans le flot de travaux

Insérez le travail de branche générique dans le flot de travaux après le travail parent et renommez l'enfant correct avec le préfixe "G_" et l'enfant incorrect avec le préfixe "B_".

Il est également recommandé de renommer le travail de branche avec un suffixe constitué du caractère de soulignement et d'une valeur numérique. Un nom classique pour le premier travail de branche dans un flot de travaux est BRANCH_1.

Branche complexe - modèle refusé

Il s'agit du cas inversé du scénario de branche de modèle.

Utilisation du scénario de modèle refusé

Avec le scénario de branche de modèle vous trouvez le journal de travail parent d'un modèle considéré comme un message positif (par exemple, `terminé correctement`). Si, au contraire, vous souhaitez rechercher un message négatif (par exemple, `Error`), utilisez le travail de branche générique pour refuser chaque

sous-condition définie de la condition complexe. Dans ce scénario, la recherche du modèle a pour conséquence d'activer CONDITION=FALSE. Si le modèle est introuvable, alors CONDITION=TRUE.

La figure 50 affiche la définition du flot de travaux pour le scénario de branche de modèle, avec les paramètres suivants :

- CONDITION_SWITCH=COMPLEX
- PATTERN_1=Error
- NEGATE CONDITION_RESULT_1=YES

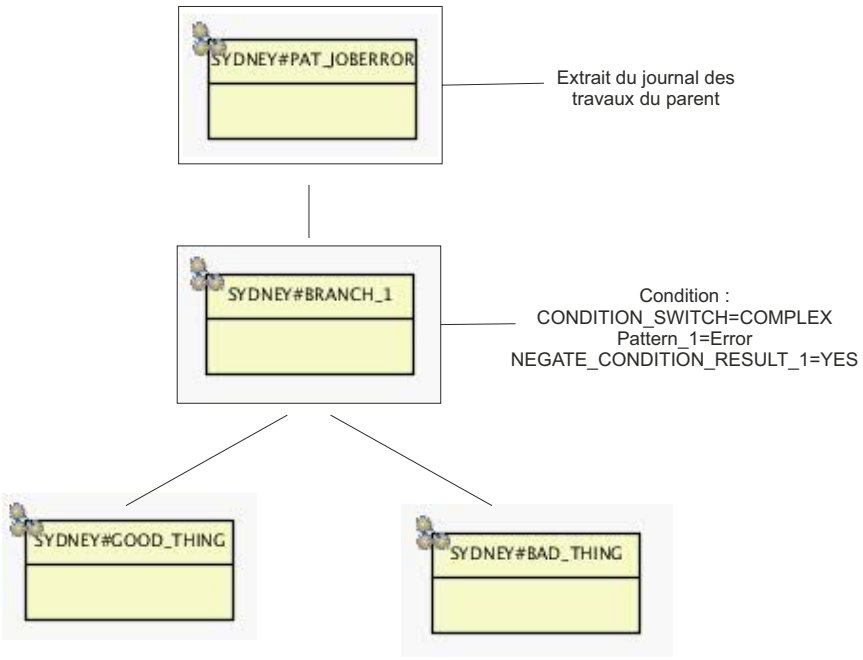


Figure 50. Définition du scénario de modèle annulé

Le journal de travail affiche la sortie de l'instance de travail de branche générique :

```
============ START of branch job BRANCH_1 ==========
=================== Job environment ================
MASTER_PLATFORM=UNIX
STREAM_NAME=GBJ_PATTERN_NEG
STREAM_CPU=SYDNEY
BRANCH_JOB_NAME=BRANCH_1
PARENT=PATTERN_JOB_ERROR
============================================================
============= Input parameters =============
CONDITION_SWITCH=COMPLEX
ACTION_SWITCH=CANCEL
CONDITION_COUNT=1
PATTERN[1]=Error
IS_CASE_SENSITIVE[1]=YES
IS_REGULAR_EXPRESSION[1]=NO
NEGATE_CONDITION_RESULT[1]=YES
============================================================
============= MAIN DECISION MAKING =============
COMPLEX condition evaluation
----------ATOMIC CONDITION 1------------
Searching for "Error" in JOBLOG of PATTERN_JOB_ERROR
Pattern FOUND, performing further tests.
No additional value defined for specified pattern.
Condition evaluated as TRUE.
==NEGATED== ATOMIC CONDITION RESULT [1]= FALSE
```

```
-----------------------------------
-------- COMPLEX CONDITION ----------
[ FALSE ]
CONDITION_RESULT=FALSE
FALSE: The result of complex condition is FALSE.
============================================================
============= Action on STOP Branch =============
Performing action CANCEL on job SYDNEY#0AAAAAAAAAAAAED3.G_DO_THE_GOOD_THING
%cj SYDNEY#0AAAAAAAAAAAAED3.G_DO_THE_GOOD_THING;schedid;noask
Command forwarded to batchman for SYDNEY#GBJ_PATTERN_NEG[(1730 12/17/07),
   (0AAAAAAAAAAAAED3)].G_DO_THE_GOOD_THING
============================================================
============= Action on RUN Branch =============
Performing action RELEASE on job SYDNEY#0AAAAAAAAAAAAED3.B_DO_THE_BAD_THING
Releasing of job SYDNEY#0AAAAAAAAAAAAED3.B_DO_THE_BAD_THING is NOT NECESSARY,
   because priority=10
============================================================
============= Statistics of branch job BRANCH_1 =============
FALSE: The result of complex condition is FALSE.
For action CANCEL - RUN_BRANCH=B_DO_THE_BAD_THING and
   STOP_BRANCH=G_DO_THE_GOOD_THING
BRANCH selected to STOP: G_DO_THE_GOOD_THING
BRANCH selected to CONTINUE: B_DO_THE_BAD_THING
CANCELED_JOBS: G_DO_THE_GOOD_THING
PAUSED_JOB:
RELEASED_JOB:
============ END of branch job BRANCH_1 ==========
```

Paramètres d'entrée obligatoires

Le tableau 124 affiche les paramètres requis pour le scénario de branche refusé.

Tableau 124. Paramètres d'entrée pour le scénario de travail de modèle refusé

Nom du paramètre	Valeur du paramètre
CONDITION_SWITCH	COMPLEXE
PATTERN_1	Error
NEGATE_CONDITION_RESULT_1	OUI

L'exemple suivant illustre la définition de paramètre. Le texte est saisi dans la zone de commentaires de la définition du flot de travaux.

```
BRANCH_1-BEGIN
CONDITION_SWITCH=COMPLEX
PATTERN_1=Error
NEGATE_CONDITION_1=YES
BRANCH_1-END
```

Pour une description sur la spécification les paramètres du travail de branche, voir «Indication des paramètres de travail de branche», à la page 840.

Placement du travail de branche dans le flot de travaux

Insérez le travail de branche générique dans le flot de travaux après le travail parent et renommez l'enfant correct avec le préfixe "G_" et l'enfant incorrect avec le préfixe "B_".

Il est également recommandé de renommer le travail de branche avec un suffixe constitué du caractère de soulignement et d'une valeur numérique. Un nom classique pour le premier travail de branche dans un flot de travaux est BRANCH_1.

Branche complexe - modèle dans la ligne de modèle

Ce scénario étend la fonction de la recherche de modèle.

Modèle dans l'utilisation de scénario de ligne de modèle

L'objet de ce scénario est de :

1. Obtenir le journal de travail parent.
2. Dans le journal de travail, identifier la ligne contenant un modèle spécifique.
3. Si la ligne est détectée, rechercher un autre modèle dans la ligne.
4. Si le second modèle est trouvé, renvoyer la condition comme TRUE.

L'exemple suivant extrait d'un journal de travail parent affiche ce à quoi ressemble un scénario d'utilisation standard :

```
Sending STOP signal to component XYZ: SUCCESS
Stopping component XYZ: SUCCESS
```

Selon cet exemple, une recherche simple pour le modèle de texte `SUCCESS` n'indiquerait pas l'arrêt effectif du composant. Il y a plusieurs occurrences du modèle `SUCCESS` et l'utilisation de la simple recherche de modèle ne vous permet pas de déterminer le résultat correct.

L'autre exemple proposé concerne un extrait du journal des travaux :

```
About to stop component XYZ...
Sending STOP signal to component XYZ: SUCCESS
Stopping component XYZ: FAILED
```

Suivant cet exemple, si vous avez utilisé la recherche simple de modèle, le résultat est CONDITION=TRUE, car le modèle `SUCCESS` est détecté. Mais dans ce cas, le journal du travail parent n'est pas évalué correctement.

Pour évaluer correctement le journal du travail parent, vous devez utiliser le modèle dans le scénario de ligne de modèle, comme suit :

1. Recherchez le modèle `Composant d'arrêt`.
2. Si la ligne est détectée, recherchez le modèle `SUCCESS`.
3. Si les deux recherches sont réussies, alors renvoyez la condition comme VRAIE.

La figure 51, à la page 815 affiche la définition du scénario de branche de modèle.

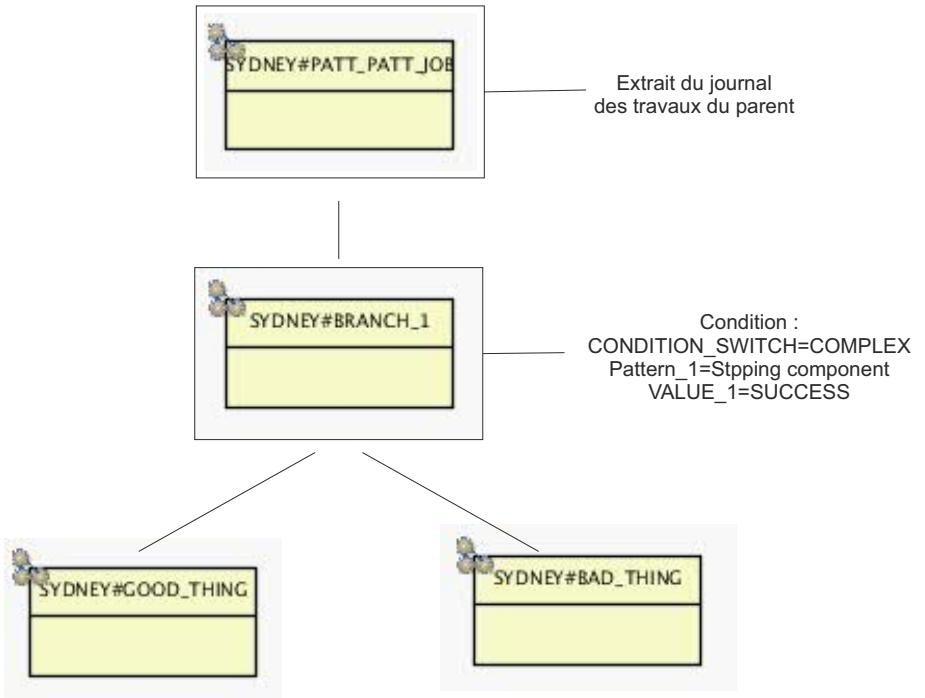


Figure 51. Modèle dans la définition de ligne de modèle

Le journal de travail affiche la sortie de l'instance de travail de branche générique :

```
============ START of branch job BRANCH_1 ==========
================== Job environment =================
MASTER_PLATFORM=UNIX
STREAM_NAME=GBJ_PATTERN_PATT
STREAM_CPU=SYDNEY
BRANCH_JOB_NAME=BRANCH_1
PARENT=PATTERN_PATTERN_JOB
============================================================
============= Input parameters =============
CONDITION_SWITCH=COMPLEX
ACTION_SWITCH=CANCEL
CONDITION_COUNT=1
PATTERN[1]=Stopping component
VALUE[1]=SUCCESS
IS_CASE_SENSITIVE[1]=YES
IS_REGULAR_EXPRESSION[1]=NO
NEGATE_CONDITION_RESULT[1]=NO
============================================================
============= MAIN DECISION MAKING =============
COMPLEX condition evaluation
----------ATOMIC CONDITION 1------------
Searching for "Stopping component" in JOBLOG of PATTERN_PATTERN_JOB
Pattern FOUND, performing further tests.
Searching for STRING=SUCCESS within the
   row "Stopping component XYZ: FAILED".
String "SUCCESS" NOT found within the
   row "Stopping component XYZ: FAILED".
ATOMIC CONDITION RESULT [1]= FALSE
------------------------------------
-------- COMPLEX CONDITION ----------
[ FALSE ]
CONDITION_RESULT=FALSE
FALSE: The result of complex condition is FALSE.
============================================================
============= Action on STOP Branch =============
Performing action CANCEL on job SYDNEY#0AAAAAAAAAAAAED2.G_DO_THE_GOOD_THING
%cj SYDNEY#0AAAAAAAAAAAAED2.G_DO_THE_GOOD_THING;schedid;noask
```

```
Command forwarded to batchman for SYDNEY#GBJ_PATTERN_PATT[(1634 12/17/07),
   (0AAAAAAAAAAAAED2)].G_DO_THE_GOOD_THING
============================================================
============= Action on RUN Branch =============
Performing action RELEASE on job SYDNEY#0AAAAAAAAAAAAED2.B_DO_THE_BAD_THING
Releasing of job SYDNEY#0AAAAAAAAAAAAED2.B_DO_THE_BAD_THING is NOT NECESSARY,
   because priority=10
============================================================
============= Statistics of branch job BRANCH_1 =============
FALSE: The result of complex condition is FALSE.
For action CANCEL - RUN_BRANCH=B_DO_THE_BAD_THING and
   STOP_BRANCH=G_DO_THE_GOOD_THING
BRANCH selected to STOP: G_DO_THE_GOOD_THING
BRANCH selected to CONTINUE: B_DO_THE_BAD_THING
CANCELED_JOBS: G_DO_THE_GOOD_THING
PAUSED_JOB:
RELEASED_JOB:
============ END of branch job BRANCH_1 ==========
```

Paramètres d'entrée obligatoires

Le tableau 125 affiche les paramètres requis pour le modèle dans le scénario de ligne de modèle.

Tableau 125. Paramètres d'entrée pour le modèle dans le scénario de ligne de modèle

Nom du paramètre	Valeur du paramètre
CONDITION_SWITCH	COMPLEXE
PATTERN_1	Arrêt du composant
VALUE_1	SUCCESS

La définition de paramètre ressemble à celle de l'exemple suivant. Le texte est saisi dans la zone de commentaires de la définition du flot de travaux.

```
BRANCH_1-BEGIN
CONDITION_SWITCH=COMPLEX
PATTERN_1=Arrêt du composant
VALUE_1=SUCCESS
BRANCH_1-END
```

Pour plus d'informations sur la définition des paramètres de travail de branche, voir «Indication des paramètres de travail de branche», à la page 840.

Placement du travail de branche dans le flot de travaux

Insérez le travail de branche générique dans le flot de travaux juste après le travail parent et renommez l'enfant correct avec le préfixe "G_" et l'enfant incorrect avec le préfixe "B_".

Il est également recommandé de renommer le travail de branche avec un suffixe constitué du caractère de soulignement et d'une valeur numérique. Un nom classique pour le premier travail de branche dans un flot de travaux est BRANCH_1.

Modèle dans la ligne de modèle - annulé

Dans le scénario de modèle annulé, vous annulez le résultat d'une recherche de modèle simple, dans le modèle annulé au sein du scénario de ligne de modèle, vous utilisez l'approche *annulée* pour *tout* scénario complexe. C'est le cas inversé du modèle dans le scénario de ligne de modèle.

Modèle annulé dans l'utilisation de scénario de ligne de modèle

Si vous effectuez une simple recherche de modèle, par exemple une recherche de l'occurrence `ERROR` dans le journal de travail parent, ou une condition complexe, vous pouvez utiliser l'approche qui annule le résultat.

Choisissez la façon d'analyser la syntaxe du journal de travail parent. En fonction de votre compréhension du contenu du journal des travaux, indiquez si la sortie des lignes contient un message positif ou négatif. A partir de ces connaissances, choisissez d'annuler le résultat de la sous-condition particulière ou de le conserver en l'état.

La logique d'évaluation du modèle annulé dans le scénario de ligne de modèle est la suivante :

1. Recherche de la ligne de modèle.
2. Recherche de l'identificateur principal, tel que `Backup on Primary device`.
3. Si la ligne est détectée, recherchez-y le message négatif `ERROR`.
4. Annulez le résultat.

Cette approche couvre le scénario d'utilisation des sorties telles que `SUCCESS`, `OK` et `COMPLETED` comme messages positifs, et des sorties telles que `FAILED` et `ERROR` comme messages négatifs.

La figure 52 affiche la définition du scénario de branche de modèle.

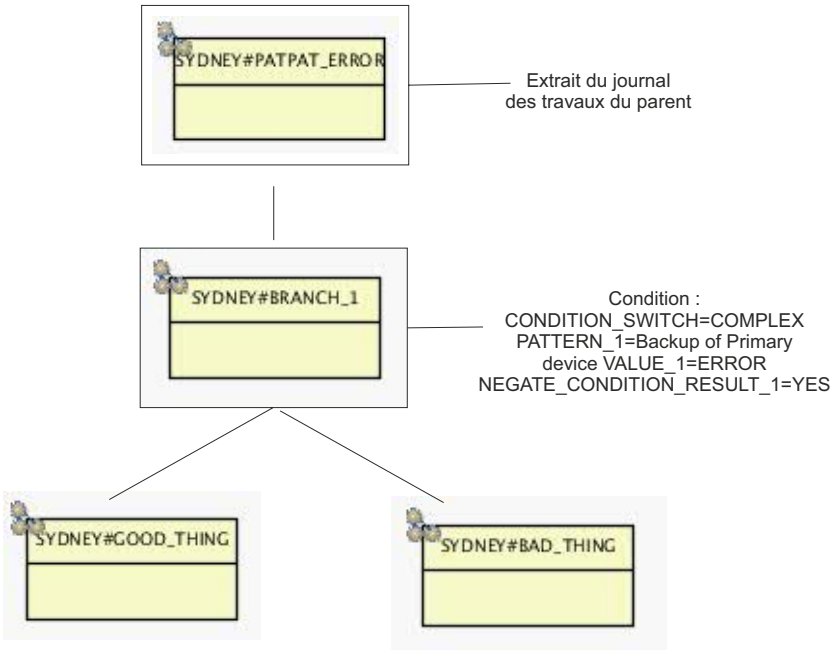


Figure 52. Modèle dans la définition annulée de ligne de modèle

Remarque : Dans *tout* scénario où vous utilisez une condition complexe, vous pouvez également annuler le résultat de *chaque sous-condition particulière*. Vous pouvez également définir des sous-conditions atomiques multiples et annuler uniquement certains d'eux. Pour plus d'informations, voir «Scénario complexe - conditions multiples», à la page 822.

Le journal de travail affiche la sortie de l'instance de travail de branche générique :

```
============ START of branch job BRANCH_1 ==========
================== Job environment =================
MASTER_PLATFORM=UNIX
STREAM_NAME=GBJ_PAT_PAT_NEG
STREAM_CPU=SYDNEY
BRANCH_JOB_NAME=BRANCH_1
PARENT=PATTERN_PATTERN_ERROR_JOB
============================================================
============= Input parameters =============
CONDITION_SWITCH=COMPLEX
ACTION_SWITCH=CANCEL
CONDITION_COUNT=1
PATTERN[1]=Backup of Primary device
VALUE[1]=ERROR
IS_CASE_SENSITIVE[1]=YES
IS_REGULAR_EXPRESSION[1]=NO
NEGATE_CONDITION_RESULT[1]=YES
============================================================
============= MAIN DECISION MAKING =============
COMPLEX condition evaluation
----------ATOMIC CONDITION 1------------
Searching for "Backup of Primary device" in JOBLOG of
   PATTERN_PATTERN_ERROR_JOB
Pattern FOUND, performing further tests.
Searching for STRING=ERROR within the
   row "Backup of Primary device: ERROR".
String "ERROR" found within the
   row "Backup of Primary device: ERROR".
==NEGATED== ATOMIC CONDITION RESULT [1]= FALSE
------------------------------------
-------- COMPLEX CONDITION ----------
[ FALSE ]
CONDITION_RESULT=FALSE
FALSE: The result of complex condition is FALSE.
============================================================
============= Action on STOP Branch =============
Performing action CANCEL on job SYDNEY#0AAAAAAAAAAAAEEA.G_DO_THE_GOOD_THING
%cj SYDNEY#0AAAAAAAAAAAAEEA.G_DO_THE_GOOD_THING;schedid;noask
Command forwarded to batchman for SYDNEY#GBJ_PAT_PAT_NEG[(1941 12/17/07),
   (0AAAAAAAAAAAAEEA)].G_DO_THE_GOOD_THING
============================================================
============= Action on RUN Branch =============
Performing action RELEASE on job SYDNEY#0AAAAAAAAAAAAEEA.B_DO_THE_BAD_THING
Releasing of job SYDNEY#0AAAAAAAAAAAAEEA.B_DO_THE_BAD_THING is NOT NECESSARY,
   because priority=10
============================================================
============= Statistics of branch job BRANCH_1 =============
FALSE: The result of complex condition is FALSE.
For action CANCEL - RUN_BRANCH=B_DO_THE_BAD_THING and
   STOP_BRANCH=G_DO_THE_GOOD_THING
BRANCH selected to STOP: G_DO_THE_GOOD_THING
BRANCH selected to CONTINUE: B_DO_THE_BAD_THING
CANCELED_JOBS: G_DO_THE_GOOD_THING
PAUSED_JOB:
RELEASED_JOB:
============ END of branch job BRANCH_1 ==========
```

Paramètres d'entrée obligatoires

Le tableau 126, à la page 819 affiche les paramètres requis pour le modèle annulé dans le scénario de ligne de modèle.

Tableau 126. Paramètres d'entrée pour le modèle annulé dans le scénario de ligne de modèle

Nom du paramètre	Valeur du paramètre
CONDITION_SWITCH	COMPLEXE
PATTERN_1	Sauvegarde d'unité principale
VALUE_1	ERROR
NEGATE_CONDITION_RESULT_1	OUI

La définition de paramètre ressemble à celle de l'exemple suivant. Le texte est saisi dans la zone de commentaires de la définition du flot de travaux.

```
BRANCH_1-BEGIN
CONDITION_SWITCH=COMPLEX
PATTERN_1=Backup of Primary device
VALUE_1=ERROR
NEGATE_CONDITION_RESULT_1=YES
BRANCH_1-END
```

Pour une description sur la manière de spécifier les paramètres pour un travail de branche, voir «Indication des paramètres de travail de branche», à la page 840.

Placement du travail de branche dans le flot de travaux

Insérez le travail de branche générique dans le flot de travaux après le travail parent et renommez l'enfant correct avec le préfixe "G_" et l'enfant incorrect avec le préfixe "B_".

Il est également recommandé de renommer le travail de branche avec un suffixe constitué du caractère de soulignement et d'une valeur numérique. Un nom classique pour le premier travail de branche dans un flot de travaux est BRANCH_1.

Branche complexe - comparaison d'une valeur numérique

Pour associer la recherche de modèle avec la comparaison d'une valeur numérique.

Modèle dans l'utilisation de branche de ligne de modèle

Ce scénario étend la fonction du scénario de modèle décrit dans Branche complexe - modèle. L'objet de ce scénario est de :

1. Obtenir le journal de travail parent.
2. Dans le journal de travail, identifier la ligne contenant un modèle spécifique.
3. Si la ligne est détectée, rechercher une valeur numérique qu'elle contient.
4. Comparer le numéro avec le numéro qui est indiqué comme paramètre d'entrée, en utilisant l'opérateur arithmétique qui est également indiqué comme paramètre.

L'extrait suivant du journal de travail parent présente un scénario d'utilisation standard :

```
Checking free space...
Free space on volume ABC is 50 %.
```

Vous souhaitez exécuter l'évaluation suivante :

1. Rechercher le modèle `Espace disponible sur le volume`.
2. Si le texte est détecté, essayez de récupérer une valeur numérique.

3. Comparer la valeur numérique comme suit :

```
If (numeric_value > 30)
Then CONDITION=TRUE
Else CONDITION=FALSE
```

Le nombre 30 et l'opérateur > sont des paramètres transmis au travail de branche. Vous pouvez utiliser tous les opérateurs arithmétiques (par exemple, <, <=, >). Pour une description sur la spécification les paramètres du travail de branche, voir «Indication des paramètres de travail de branche», à la page 840.

La figure 53 affiche la définition du scénario de comparaison d'une valeur numérique. Vous pouvez également inverser la logique d'évaluation, bien que généralement cela ne soit pas nécessaire car vous pouvez atteindre le résultat de condition annulé à l'aide de l'opérateur opposé.

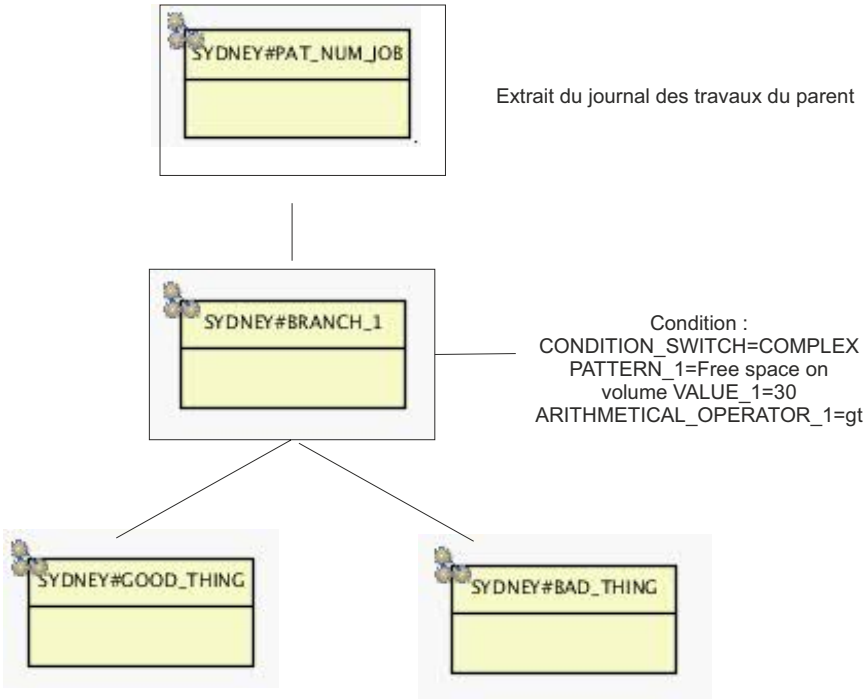


Figure 53. Définition de branche de comparaison numérique

Le journal de travail affiche la sortie de l'instance de travail de branche générique :

```
============ START of branch job BRANCH_1 ==========
================ Job environment ==================
MASTER_PLATFORM=UNIX
STREAM_NAME=GBJ_PATTERN_NUM
STREAM_CPU=SYDNEY
BRANCH_JOB_NAME=BRANCH_1
PARENT=PATTERN_NUMBER_JOB
============================================================
============= Input parameters =============
CONDITION_SWITCH=COMPLEX
ACTION_SWITCH=CANCEL
CONDITION_COUNT=1
PATTERN[1]=Free space on volume
IS_CASE_SENSITIVE[1]=YES
IS_REGULAR_EXPRESSION[1]=NO
VALUE[1]=30
ARITHMETICAL_OPERATOR[1]=-gt
NEGATE_CONDITION_RESULT[1]=NO
============================================================
```

```
============= MAIN DECISION MAKING =============
COMPLEX condition evaluation
----------ATOMIC CONDITION 1------------
Searching for "Free space on volume" in JOBLOG of PATTERN_NUMBER_JOB
Pattern FOUND, performing further tests.
Searching for NUMBER withing row...
Number found=50. Evaluating arithmetical expression [ 50 -gt 30 ]
Arithmetical expression [ 50 -gt 30 ] evaluated as TRUE.
ATOMIC CONDITION RESULT [1]= TRUE
-----------------------------------
-------- COMPLEX CONDITION ----------
[ TRUE ]
CONDITION_RESULT=TRUE
TRUE: The result of complex condition is TRUE.
============================================================
============= Action on STOP Branch =============
Performing action CANCEL on job SYDNEY#0AAAAAAAAAAAAEET.B_DO_THE_BAD_THING
%cj SYDNEY#0AAAAAAAAAAAAEET.B_DO_THE_BAD_THING;schedid;noask
Command forwarded to batchman for SYDNEY#GBJ_PATTERN_NUM[(2113 12/17/07),
(0AAAAAAAAAAAAEET)].B_DO_THE_BAD_THING
============================================================
============= Action on RUN Branch =============
Performing action RELEASE on job SYDNEY#0AAAAAAAAAAAAEET.G_DO_THE_GOOD_THING
Releasing of job SYDNEY#0AAAAAAAAAAAAEET.G_DO_THE_GOOD_THING is NOT NECESSARY,
because priority=10
============================================================
============= Statistics of branch job BRANCH_1 =============
TRUE: The result of complex condition is TRUE.
For action CANCEL - RUN_BRANCH=G_DO_THE_GOOD_THING and
STOP_BRANCH=B_DO_THE_BAD_THING
BRANCH selected to STOP: B_DO_THE_BAD_THING
BRANCH selected to CONTINUE: G_DO_THE_GOOD_THING
CANCELED_JOBS: B_DO_THE_BAD_THING
PAUSED_JOB:
RELEASED_JOB:
============ END of branch job BRANCH_1 ==========
```

Paramètres d'entrée obligatoires

Le tableau 127 affiche les paramètres requis pour le modèle dans le scénario de ligne de modèle.

Tableau 127. Paramètres d'entrée pour le scénario de comparaison numérique

Nom du paramètre	Valeur du paramètre
CONDITION_SWITCH	COMPLEXE
PATTERN_1	Free space on volume
VALUE_1	30
ARITHMETICAL_OPERATOR	-gt

Il est important de comprendre l'ordre utilisé pour transmettre les nombres à l'expression arithmétique :

number_from_the_joblog compared_against *number_supplied_as_parameter*

compared_against est l'opérateur arithmétique spécifié comme paramètre. Si vous n'indiquez pas d'opérateur, la valeur par défaut -eq (égal) est utilisée.

L'exemple suivant illustre la définition de paramètre. Le texte est saisi dans la zone de commentaires de la définition du flot de travaux.

```
BRANCH_1-BEGIN
CONDITION_SWITCH=COMPLEX
PATTERN_1=Free space on volume
VALUE_1=30
ARITHMETICAL_OPERATOR_1=-gt
BRANCH_1-END
```

Pour une description sur la manière de spécifier les paramètres pour un travail de branche, voir «Indication des paramètres de travail de branche», à la page 840.

Placement du travail de branche dans le flot de travaux

Insérez le travail de branche générique dans le flot de travaux après le travail parent et renommez l'enfant correct avec le préfixe "G_" et l'enfant incorrect avec le préfixe "B_".

De plus, il est recommandé de renommer le travail de branche avec un suffixe constitué du caractère de soulignement et d'une valeur numérique. Un nom classique pour le premier travail de branche dans un flot de travaux est BRANCH_1.

Scénario complexe - conditions multiples

Utiliser le scénario de conditions multiples pour définir plusieurs sous-conditions en même temps.

Utilisation de condition complexe

Cette section décrit comment insérer tous les éléments atomiques d'une condition complexe. Le travail de branche obtient le journal de travail parent et exécute une condition complexe sur cette ressource :

- Le journal de travail *ne doit pas* inclure le modèle `Error`.

et

- L'une des conditions atomiques suivantes doit être satisfaite :
  - L'espace disponible sur l'unité principale est connu, et sa valeur est supérieure à 50.
  - L'espace disponible sur l'unité secondaire est connu, et sa valeur est supérieure à 60.

La figure 54, à la page 823 affiche la définition du scénario de branche complexe.

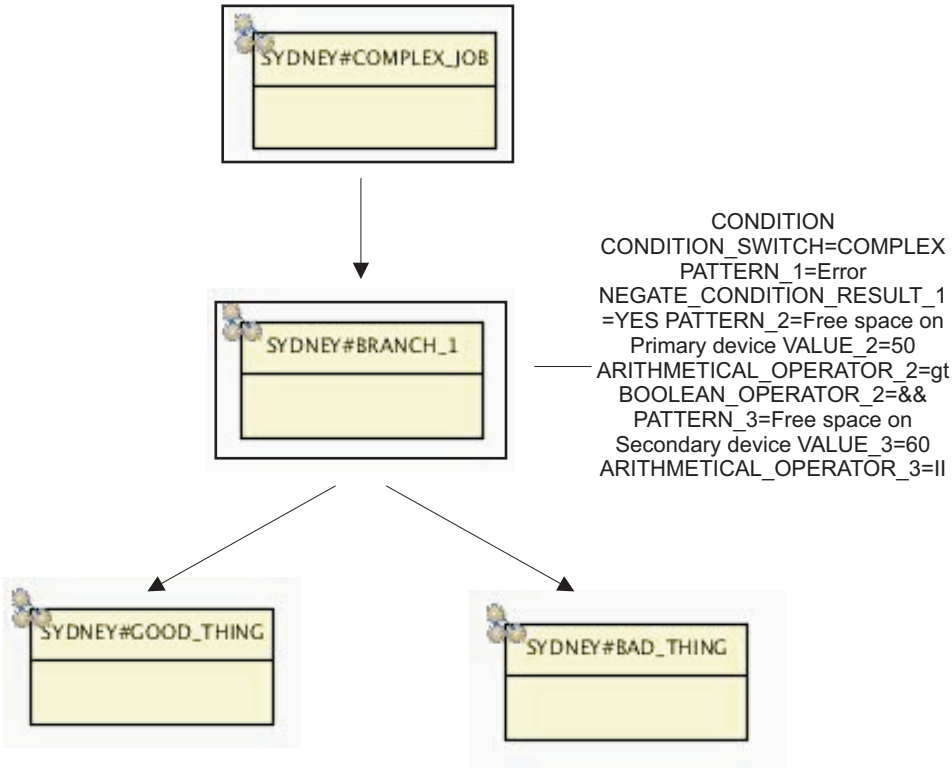


Figure 54. Définition de condition complexe

Le journal de travail affiche la sortie de l'instance de travail de branche générique :

```
============ START of branch job BRANCH_1 ==========
================ Job environment ===================
MASTER_PLATFORM=UNIX
STREAM_NAME=GBJ_COMPLEX
STREAM_CPU=SYDNEY
BRANCH_JOB_NAME=BRANCH_1
PARENT=COMPLEX_JOB
============================================================
============= Input parameters =============
CONDITION_SWITCH=COMPLEX
ACTION_SWITCH=CANCEL
CONDITION_COUNT=3
PATTERN[1]=Error
IS_CASE_SENSITIVE[1]=YES
IS_REGULAR_EXPRESSION[1]=NO
NEGATE_CONDITION_RESULT[1]=YES
PATTERN[2]=Free space on Primary device
IS_CASE_SENSITIVE[1]=YES
IS_REGULAR_EXPRESSION[1]=NO
VALUE[2]=50
ARITHMETICAL_OPERATOR[2]=-gt
BOOLEAN OPERATOR[2]=&&
NEGATE_CONDITION_RESULT[2]=NO
PATTERN[3]=Free space on Secondary device
IS_CASE_SENSITIVE[1]=YES
IS_REGULAR_EXPRESSION[1]=NO
VALUE[3]=60
ARITHMETICAL_OPERATOR[3]=-gt
BOOLEAN OPERATOR[3]=||
NEGATE_CONDITION_RESULT[3]=NO
============================================================
============= MAIN DECISION MAKING =============
COMPLEX condition evaluation
----------ATOMIC CONDITION 1------------
Searching for "Error" in JOBLOG of COMPLEX_JOB
```

```
Pattern NOT FOUND, condition evaluated as FALSE.
==NEGATED== ATOMIC CONDITION RESULT [1]= TRUE
----------ATOMIC CONDITION 2------------
Searching for "Free space on Primary device" in JOBLOG of COMPLEX_JOB
Pattern FOUND, performing further tests.
Searching for NUMBER withing row...
Number found=55. Evaluating arithmetical expression [ 55 -gt 50 ]
Arithmetical expression [ 55 -gt 50 ] evaluated as TRUE.
ATOMIC CONDITION RESULT [2]= TRUE
----------ATOMIC CONDITION 3------------
Searching for "Free space on Secondary device" in JOBLOG of COMPLEX_JOB
Pattern FOUND, performing further tests.
Searching for NUMBER withing row...
Number found=10. Evaluating arithmetical expression [ 10 -gt 60 ]
Arithmetical expression [ 10 -gt 60 ] evaluated as FALSE.
ATOMIC CONDITION RESULT [3]= FALSE
-----------------------------------
-------- COMPLEX CONDITION ----------
[ TRUE ] && [ TRUE ] || [ FALSE ]
CONDITION_RESULT=TRUE
TRUE: The result of complex condition is TRUE.
=========================================================
============= Action on STOP Branch =============
Performing action CANCEL on job SYDNEY#0AAAAAAAAAAAAEEW.B_DO_THE_BAD_THING
%cj SYDNEY#0AAAAAAAAAAAAEEW.B_DO_THE_BAD_THING;schedid;noask
Command forwarded to batchman for SYDNEY#GBJ_COMPLEX[(2219 12/17/07)
(0AAAAAAAAAAAAEEW)].B_DO_THE_BAD_THING
=========================================================
============= Action on RUN Branch =============
Performing action RELEASE on job SYDNEY#0AAAAAAAAAAAAEEW.G_DO_THE_GOOD_THING
Releasing of job SYDNEY#0AAAAAAAAAAAAEEW.G_DO_THE_GOOD_THING is NOT NECESSARY,
because priority=10
=========================================================
============= Statistics of branch job BRANCH_1 =============
TRUE: The result of complex condition is TRUE.
For action CANCEL - RUN_BRANCH=G_DO_THE_GOOD_THING and
STOP_BRANCH=B_DO_THE_BAD_THING
BRANCH selected to STOP: B_DO_THE_BAD_THING
BRANCH selected to CONTINUE: G_DO_THE_GOOD_THING
CANCELED_JOBS: B_DO_THE_BAD_THING
PAUSED_JOB:
RELEASED_JOB:
============ END of branch job BRANCH_1 ==========
```

Paramètres d'entrée obligatoires

Le tableau 128 affiche les paramètres requis pour le scénario de branche complexe.

Tableau 128. Paramètres d'entrée pour le scénario de condition complexe

Nom du paramètre	Valeur du paramètre
CONDITION_SWITCH	COMPLEXE
PATTERN_1	Error
NEGATE_CONDITION_RESULT_1	OUI
PATTERN_2	Free space on Primary device
VALUE_2	50
ARITHMETICAL_OPERATOR_2	-gt
BOOLEAN OPERATOR_2	&&
PATTERN_3	Free space on Secondary device
VALUE_3	60

Tableau 128. Paramètres d'entrée pour le scénario de condition complexe (suite)

Nom du paramètre	Valeur du paramètre
ARITHMETICAL_OPERATOR_3	-gt
BOOLEAN OPERATOR_3	||

L'exemple suivant illustre la définition de paramètre. Le texte est saisi dans la zone de commentaires de la définition du flot de travaux.

```
BRANCH_1-BEGIN
CONDITION_SWITCH=COMPLEX
PATTERN_1=Error
NEGATE_CONDITION_RESULT_1=YES
PATTERN_2=Free space on Primary device
VALUE_2=50
ARITHMETICAL_OPERATOR_2=-gt
BOOLEAN_OPERATOR_2=&&
PATTERN_3=Free space on Secondary device
VALUE_3=60
ARITHMETICAL_OPERATOR_3=-gt
BOOLEAN_OPERATOR_3=||
BRANCH_1-END
```

Pour une description sur la manière de spécifier les paramètres pour un travail de branche, voir «Indication des paramètres de travail de branche», à la page 840.

Placement du travail de branche dans le flot de travaux

Insérez le travail de branche générique dans le flot de travaux après le travail parent et renommez l'enfant correct avec le préfixe "G_" et l'enfant incorrect avec le préfixe "B_".

Il est également recommandé de renommer le travail de branche avec un suffixe constitué du caractère de soulignement et d'une valeur numérique. Un nom typique du premier travail de branche dans un flot de travaux est BRANCH_1.

Paramètres de chaîne supplémentaires

Le travail générique de branche exécute la recherche de modèle à l'aide de la commande `grep` qui accepte plusieurs paramètres d'entrée. Pour affiner la recherche de modèle pour le travail générique de branche, vous pouvez utiliser `IS_CASE_SENSITIVE_i` et `IS_REGULAR_EXPRESSION_i`.

Utilisez les paramètres comme suit :

IS_CASE_SENSITIVE_i_
: Pour activer ou désactiver la distinction entre majuscules et minuscules dans une recherche. La valeur par défaut est YES.

IS_REGULAR_EXPRESSION_i_
: Pour activer ou désactiver une recherche en fonction d'expressions régulières. Par défaut, la valeur est NO.

Remarque : Le suffixe _i est l'index de la sous-condition particulière.

Pour une description sur la manière de spécifier les paramètres pour un travail de branche, voir «Indication des paramètres de travail de branche», à la page 840.

Scénarios basés sur le type d'action

Utilisez un travail générique de branche en fonction du type d'action pour indiquer les critères qui déterminent la branche d'exécution et la branche d'arrêt.

Vous pouvez effectuer les actions suivantes :

- Branche d'arrêt
  - CANCEL : la branche d'arrêt est annulée. Il s'agit de l'action la plus fréquente sur la branche d'arrêt.
  - PAUSE : le premier travail de la branche d'arrêt est mis en pause (HOLD). Pour plus de détails, voir Scénario d'actions de pause/libération
- Branche d'exécution
  - Aucune action : la branche d'exécution s'exécute. Il s'agit de l'action la plus fréquente sur la branche d'exécution.
  - Libération : si le premier travail de la branche d'exécution mis en pause (HOLD), augmenter sa priorité (RELEASE). Pour plus de détails, voir Scénario d'actions de pause/libération
- Action spéciale
  - SIGNAL : cette action n'a pas d'effet sur les branches. Elle recommande la confirmation de l'opérateur Tivoli Workload Scheduler à appliquer. Pour plus de détails, voir Scénario d'action de signal.

Scénario d'actions de pause et de libération

Utilisez ce travail de branche pour gérer un flux de travaux qui est sensible à des résultats de travaux importants, par exemple, pour déterminer quand l'action réalisée par un travail important n'a pas abouti.

Utilisation d'actions de pause et de libération

Utilisez le scénario de pause/libération lorsque, même si le travail de branche a identifié un statut d'erreur, vous ne souhaitez pas annuler les branches. Au lieu d'opter pour l'annulation automatique, vous définissez une séquence d'actions correctives pour l'exécution du travail. Si les actions réussissent, le travail continue comme si l'erreur ne s'était pas produite.

Le flux de processus est similaire à celui présenté dans l'exemple suivant :

1. Vous avez un travail important suivi du premier travail de branche.
2. Le travail de branche est suivi par une bonne branche et un mauvaise branche : la bonne branche (nommé OKbranch) comprend les travaux à exécuter si tout réussit et la mauvaise branche (nommé Correctivebranch) comprend une séquence de tâches destinée à exécuter des actions correctives.
3. Le premier travail de branche évalue la condition exécutée sur le travail parent (votre *travail important*) : si CONDITION=TRUE, tout réussit et la branche d'arrêt est annulée. Tous les travaux de la branche corrective sont également annulés, car aucune action corrective n'est nécessaire.

   Si CONDITION=FALSE, la branche d'arrêt *n'est pas* annulée mais *mise en pause*, ce qui signifie que l'enfant correct (premier travail de la branche d'arrêt) est mis en pause. Dans ce cas, OKbranch est suspendu.
4. Lorsque l'OKbranch est mis en pause, la branche corrective lance les actions correctives.
5. A l'issue de la séquence d'actions correctives, le deuxième travail de branche (placé dans la branche corrective) est soumis pour évaluer le résultat des actions correctives.

Si les corrections réussissent, l'OKbranch est libéré ; si les corrections échouent, l'OKbranch est annulé et le flot de travaux continue d'exécuter la branche incorrecte du deuxième travail de branche.

Généralement, la branche incorrecte du deuxième travail de branche contient un seul travail ABEND (un travail qui exécute une commande *exit 1*). Nous vous conseillons d'exécuter la branche incorrecte de la branche corrective avec le travail ABEND, car cela garantit que le flot de travaux entier s'arrête de manière anormale (les travaux abandonnés précédents possèdent Recovery Option=CONTINUE, ce qui signifie qu'ils n'ont pas propagé l'état ABEND au statut du flot de travaux final).

Remarque : Les deux travaux de branche doivent indiquer le même enfant correct. C'est absolument crucial pour le fonctionnement du processus.

La figure 55, à la page 828 affiche la définition du flot de travaux pour le scénario pause/libération. Le paramètre ACTION_SWITCH=PAUSE est défini pour le premier travail de branche. Aucun paramètre n'est défini pour la deuxième travail de branche.

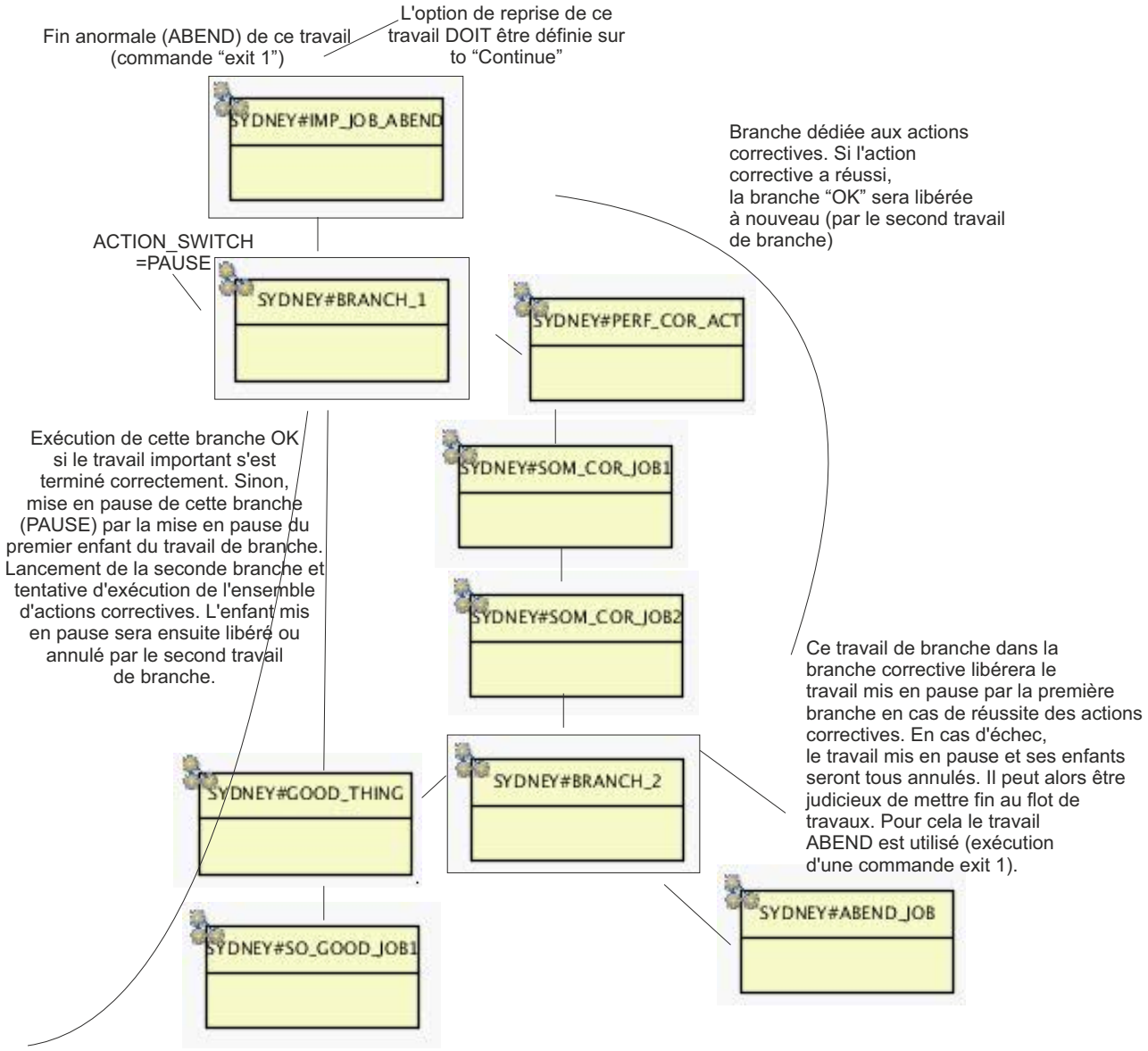


Figure 55. Définition d'actions de pause et de libération

Le journal de travail affiche la sortie de l'instance du premier travail de branche :

```
============ START of branch job BRANCH_1 ==========
================= Job environment ==================
MASTER_PLATFORM=UNIX
STREAM_NAME=GBJ_PAUSE
STREAM_CPU=SYDNEY
BRANCH_JOB_NAME=BRANCH_1
PARENT=IMPORTANT_JOB_ABEND
============================================================
============= Input parameters =============
CONDITION_SWITCH=PARENT_SUCCESS
ACTION_SWITCH=PAUSE
============================================================
============= MAIN DECISION MAKING =============
Evaluation dependent on PARENT_SUCCESS
FALSE: Searched for SUCC parent.
Status of PARENT JOB(IMPORTANT_JOB_ABEND) is ABEND.
============================================================
============= Action on STOP Branch =============
```

```
Performing action PAUSE on job SYDNEY#0AAAAAAAAAAAAEEX.G_DO_THE_GOOD_THING
%altpri SYDNEY#0AAAAAAAAAAAAEEX.G_DO_THE_GOOD_THING;schedid;0;noask
Command forwarded to batchman for SYDNEY#GBJ_PAUSE[(2300 12/17/07),
  (0AAAAAAAAAAAAEEX)].G_DO_THE_GOOD_THING
============================================================
============= Action on RUN Branch =============
Performing action RELEASE on job
SYDNEY#0AAAAAAAAAAAAEEX.B_PERFORM_CORRECTIVE_ACTIONS
Releasing of job SYDNEY#0AAAAAAAAAAAAEEX.B_PERFORM_CORRECTIVE_ACTIONS
is NOT NECESSARY,
because priority=10
============================================================
============= Statistics of branch job BRANCH_1 =============
FALSE: Searched for SUCC parent.
Status of PARENT JOB(IMPORTANT_JOB_ABEND) is ABEND.
For action PAUSE - RUN_BRANCH=B_PERFORM_CORRECTIVE_ACTIONS
and STOP_BRANCH=G_DO_THE_GOOD_THING
BRANCH selected to STOP: G_DO_THE_GOOD_THING
BRANCH selected to CONTINUE: B_PERFORM_CORRECTIVE_ACTIONS
CANCELED_JOBS:
PAUSED_JOB: G_DO_THE_GOOD_THING
RELEASED_JOB:
============ END of branch job BRANCH_1 ==========
```

Le journal de travail affiche la sortie de l'instance du second travail de branche :

```
============ START of branch job BRANCH_2 ==========
============================================================
============= Job environment =============
MASTER_PLATFORM=UNIX
STREAM_NAME=GBJ_PAUSE
STREAM_CPU=SYDNEY
BRANCH_JOB_NAME=BRANCH_2
PARENT=SOME_CORRECTIVE_JOB_2
============================================================
============= Input parameters =============
CONDITION_SWITCH=PARENT_SUCCESS
ACTION_SWITCH=CANCEL
============================================================
============= MAIN DECISION MAKING =============
Evaluation dependent on PARENT_SUCCESS
TRUE: Searched for SUCC parent. Status of
PARENT JOB(SOME_CORRECTIVE_JOB_2) is SUCC.
============================================================
============= Action on STOP Branch =============
Performing action CANCEL on job SYDNEY#0AAAAAAAAAAAAEEX.B_ABEND_JOB
%cj SYDNEY#0AAAAAAAAAAAAEEX.B_ABEND_JOB;schedid;noask
Command forwarded to batchman for SYDNEY#GBJ_PAUSE[(2300 12/17/07),
   (0AAAAAAAAAAAAEEX)].B_ABEND_JOB
============================================================
============= Action on RUN Branch =============
Performing action RELEASE on job
SYDNEY#0AAAAAAAAAAAAEEX.G_DO_THE_GOOD_THING
Releasing SYDNEY#0AAAAAAAAAAAAEEX.G_DO_THE_GOOD_THING, because priority=0
%altpri SYDNEY#0AAAAAAAAAAAAEEX.G_DO_THE_GOOD_THING;schedid;10;noask
Command forwarded to batchman for SYDNEY#GBJ_PAUSE[(2300 12/17/07),
   (0AAAAAAAAAAAAEEX)].G_DO_THE_GOOD_THING
============================================================
============= Statistics of branch job BRANCH_2 =============
TRUE: Searched for SUCC parent. Status of
PARENT JOB(SOME_CORRECTIVE_JOB_2) is SUCC.
For action CANCEL - RUN_BRANCH=G_DO_THE_GOOD_THING and
STOP_BRANCH=B_ABEND_JOB
BRANCH selected to STOP: B_ABEND_JOB
BRANCH selected to CONTINUE: G_DO_THE_GOOD_THING
CANCELED_JOBS: B_ABEND_JOB
```

```
PAUSED_JOB:
RELEASED_JOB:G_DO_THE_GOOD_THING
============ END of branch job BRANCH_2 ==========
```

Paramètres d'entrée obligatoires

Le tableau 129 affiche les paramètres requis pour le premier travail de branche du scénario de pause/libération. Le second travail de branche ne gère pas de paramètre.

Tableau 129. Paramètres d'entrée pour le scénario de pause/libération

Nom du paramètre	Valeur du paramètre
ACTION_SWITCH	PAUSE

Remarque :

- Le parameter ACTION_SWITCH=PAUSE est requis uniquement pour le premier travail de branche dans le flot de travaux. Le second travail de branche doit avoir la valeur ACTION SWITCH=CANCEL (par défaut).
- Les deux travaux de branche doivent indiquer le même enfant correct.
- Chaque travail de branche doit avoir un suffixe différent. Par exemple, dans ce scénario simple de pause et de libération, les deux noms de travail de branche sont utilisés : BRANCH_1 et BRANCH_2.

La définition de paramètre ressemble à celle de l'exemple suivant. Le texte est saisi dans la zone de commentaires de la définition du flot de travaux.

```
BRANCH_1-BEGIN
ACTION_SWITCH=PAUSE
BRANCH_1-END
```

Pour une description sur la manière de spécifier les paramètres pour un travail de branche, voir «Indication des paramètres de travail de branche», à la page 840.

Placement du travail de branche dans le flot de travaux

Insérez le travail de branche générique dans le flot de travaux après le travail parent et renommez l'enfant correct avec le préfixe "G_" et l'enfant incorrect avec le préfixe "B_".

Le premier travail de branche détermine l'OKbranch (suivi de l'enfant correct) et le CorrectiveBranch (suivi de l'enfant incorrect). Le travail représentant l'enfant correct doit posséder le préfixe « G_ », tandis que le travail représentant l'enfant incorrect doit avoir le préfixe « B_ ».

Les deux travaux de branche doivent indiquer le même enfant correct. Cela signifie que le bon enfant du premier travail de branche doit être identique au bon enfant du deuxième travail de branche.

La meilleure pratique est de définir le travail ABEND comme l'enfant incorrect du second travail de branche. Le travail ABEND appelle simplement la commande système *exit 1*, qui a pour effet de mettre fin au travail de façon anormale.

Le fait d'avoir un travail ABEND dans la branche incorrecte assure la propagation de l'état ABEND s'effectue également au niveau du flot de travaux. Aucun travail terminé de façon anormale précédemment ne propagerait l'état ABEND au niveau du flot de travaux si l'option ABEND au niveau de le flot de travaux si l'option de

reprise est réglée sur Continuer. Pour permettre l'exécution du travail de branche, vous devez régler l'option de reprise sur Continuer pour tous les parents du travail de branche.

Scénario d'actions de pause et de libération multiples

Utilisez le scénario d'actions de pause et de libération multiples pour exécuter une séquence d'actions correctives et quitter la branche corrective lorsque l'une d'elles réussit. Lorsqu'une correction aboutit, le travail de branche annule les corrections restantes et libère l'OKbranch.

Utilisation d'actions de pause et de libération multiples

La figure 56, à la page 832 affiche la définition de flot de travaux.

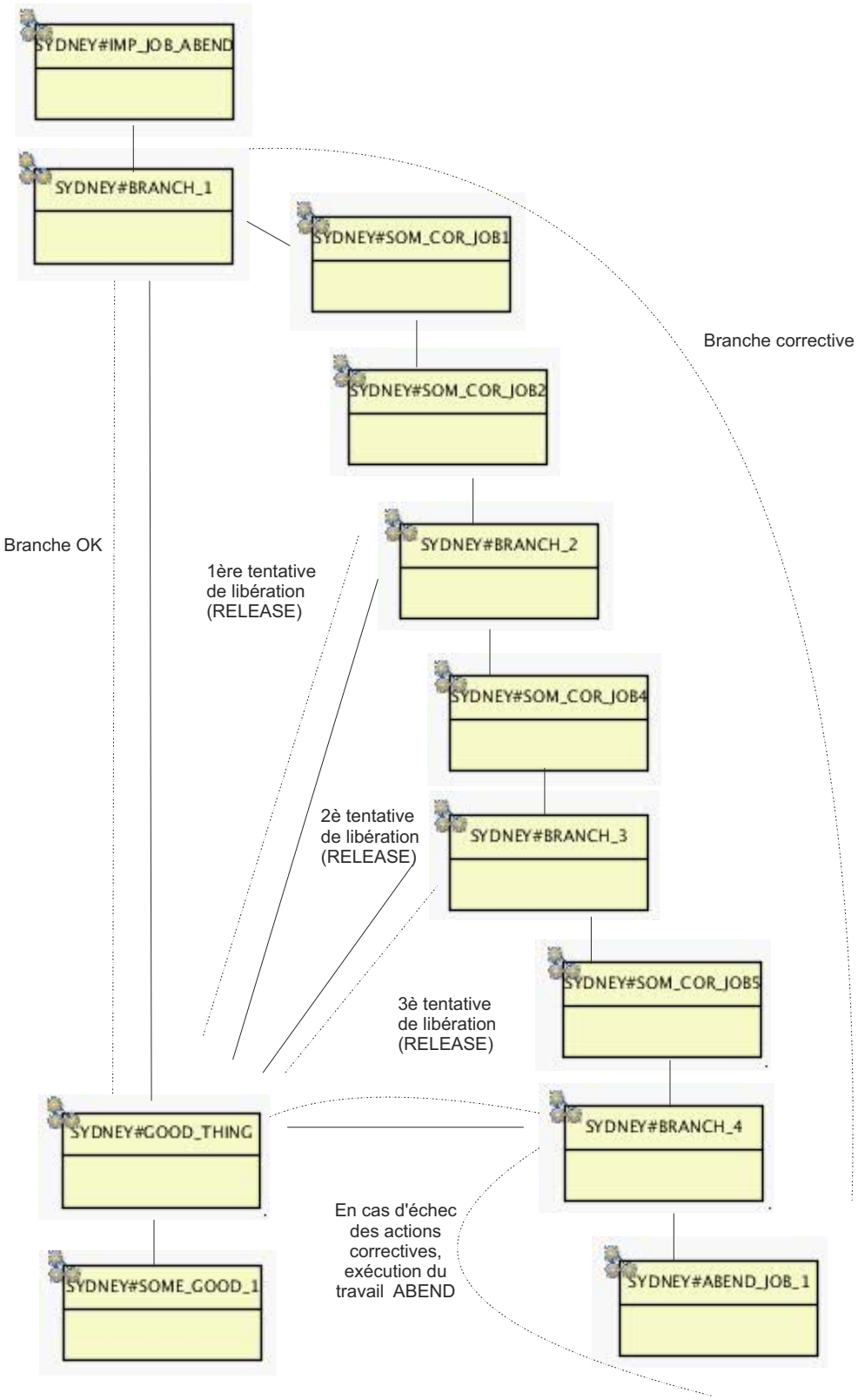


Figure 56. Définition de scénario d'actions de pause et de libération multiples

Vous devez définir le paramètre ACTION_SWITCH=PAUSE pour *tous* les travaux de branche à l'exception du dernier. Par conséquent, dans ce scénario, les travaux de branche BRANCH_1, BRANCH_2 et BRANCH_3 doivent avoir

ACTION_SWITCH=PAUSE. Si vous ne définissez pas ce paramètre, l'enfant approprié est annulé par un travail médian de branche.

Tous les travaux de branche doivent pointer sur le même enfant approprié.

Vérifiez que chaque travail de branche a un suffixe différent. Par exemple, dans ce scénario, les noms BRANCH_1, BRANCH_2, BRANCH_3 et BRANCH_4 sont utilisés.

La meilleure pratique est de définir le travail ABEND comme l'enfant incorrect du dernier travail de branche. Le travail ABEND appelle la commande système `exit 1`. Cela provoque la fin anormale du travail (TO END ABNORMALLY) et son statut est répercuté au niveau de le flot de travaux.

Scénario d'action de signal

Utilisez ce scénario pour disposer d'un travail de signal qui traite et stocke les informations utiles à la prise de décision dans le journal du travail.

Utilisation de scénario d'action de signal

D'un point de vue logique, le travail de signal et le travail de branche sont différents dans leur étape dernière (c'est-à-dire l'action exécutée). Bien que le travail de branche annule toujours, mette en pause ou libère ses travaux enfant, le travail de signal enregistre uniquement une recommandation pour vous permettre de prendre une décision. Au lieu de bloquer le processus, le scénario de signal sélectionne la branche d'exécution et permet au flot de travaux de continuer. Ce scénario étend l'approche déjà disponible avec les invites de Tivoli Workload Scheduler ; il représente la combinaison d'invites avec les droits d'accès du travail de branche.

Dans ce scénario, il existe deux travaux dans l'ordre séquentiel suivant :

1. Travail de signal
2. Travail de branche

Pour le travail de branche :

1. Indiquez le paramètre ACTION_SWITCH=SIGNAL et définissez l'option de reprise sur CONTINUE. Dans la définition du flot de travaux, définissez l'indicateur `Confirmation requise`.

   Le travail de signal exécute la logique d'évaluation, ce qui signifie qu'il évalue la condition sur les propriétés du parent, mais n'annule pas et ne met pas en pause *aucun* de ses travaux enfant.

   L'indicateur Confirmation requise provoque l'arrêt du flot de travaux par le travail de signal. Une fois terminé, le travail de signal reste à l'état PEND. Le journal des travaux du signal présente l'évaluation complète des conditions, y compris la recommandation de confirmation. Cela signifie que le travail de signal évalue la condition et écrit la recommandation (Confirm SUCC ou Confirm ABEND) dans le journal de travail.
2. Consultez le journal du travail du signal et choisissez de confirmer le travail avec un statut SUCC ou ABEND.

   Le travail de branche suivant démarre uniquement lorsque vous confirmez le travail de signal. Le travail de branche évalue le statut défini pour le travail de signal et détermine la branche d'exécution et la branche d'arrêt.

Cette dernière étape représente le branchement simple, qui est décrit dans «Scénario de branche simple», à la page 797.

La figure 57 affiche la définition du flot de travaux pour le scénario de signal.

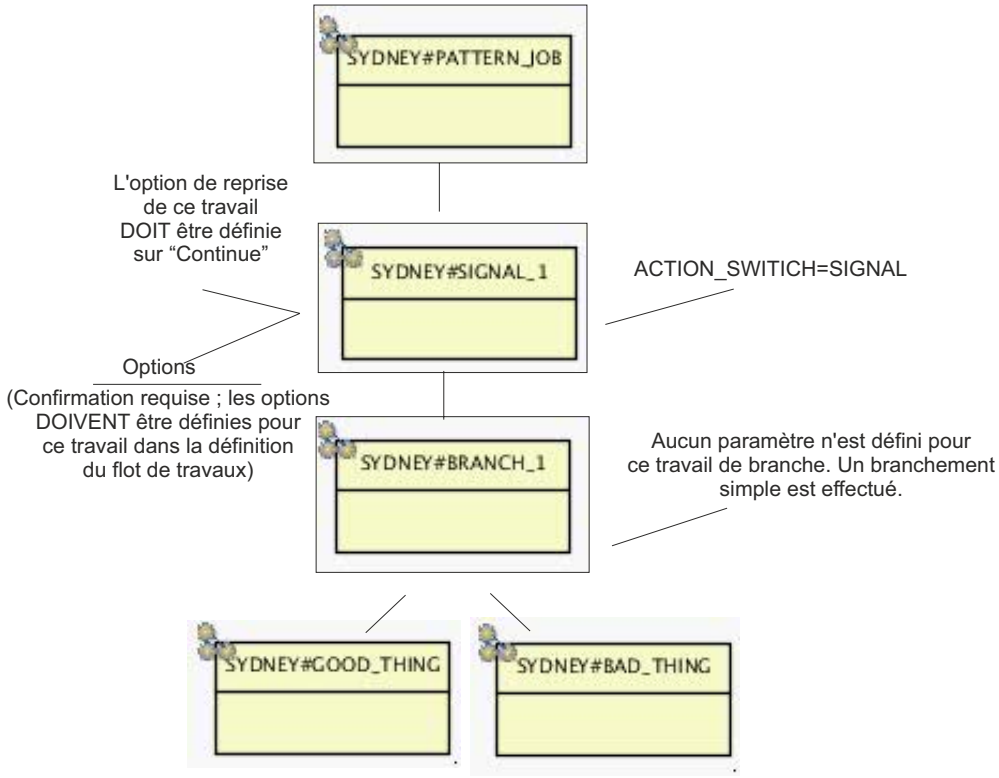


Figure 57. Définition d'action de signal

Le journal du premier travail de branche indique que le travail est mis en attente (HELD), car il est à l'état PEND. Ce statut nécessite votre confirmation, sinon les successeurs du travail ne s'exécutent pas.

```
============ START of branch job SIGNAL_1 ==========
================= Job environment ==================
MASTER_PLATFORM=UNIX
STREAM_NAME=GBJ_SIGNAL
STREAM_CPU=SYDNEY
BRANCH_JOB_NAME=SIGNAL_1
PARENT=PATTERN_JOB
============================================================
============= Input parameters =============
CONDITION_SWITCH=COMPLEX
ACTION_SWITCH=SIGNAL
CONDITION_COUNT=1
PATTERN[1]=completed successfully
IS_CASE_SENSITIVE[1]=YES
IS_REGULAR_EXPRESSION[1]=NO
NEGATE_CONDITION_RESULT[1]=NO
============================================================
============= MAIN DECISION MAKING =============
COMPLEX condition evaluation
----------ATOMIC CONDITION 1------------
Searching for "completed successfully" in JOBLOG of PATTERN_JOB
Pattern FOUND, performing further tests.
No additional value defined for specified pattern. Condition evaluated as TRUE.
ATOMIC CONDITION RESULT [1]= TRUE
-------------------------------------
```

```
-------- COMPLEX CONDITION ----------
[ TRUE ]
CONDITION_RESULT=TRUE
TRUE: The result of complex condition is TRUE.
============================================================
============= Statistics of branch job SIGNAL_1 =============
******Recommended confirmation for this job is SUCC.******
============ END of branch job SIGNAL_1 ==========
```

The log of the second branch job shows the processing of the following branch job instance.

```
============ START of branch job BRANCH_1 ==========
============================================================
============= Job environment =============
MASTER_PLATFORM=UNIX
STREAM_NAME=GBJ_SIGNAL
STREAM_CPU=SYDNEY
BRANCH_JOB_NAME=BRANCH_1
PARENT=SIGNAL_1
============================================================
============= Input parameters =============
CONDITION_SWITCH=PARENT_SUCCESS
ACTION_SWITCH=CANCEL
============================================================
============= MAIN DECISION MAKING =============
Evaluation dependent on PARENT_SUCCESS
TRUE: Searched for SUCC parent. Status of PARENT JOB(SIGNAL_1) is SUCC.
============================================================
============= Action on STOP Branch =============
Performing action CANCEL on job SYDNEY#0AAAAAAAAAAAAEI6.B_DO_THE_BAD_THING
%cj SYDNEY#0AAAAAAAAAAAAEI6.B_DO_THE_BAD_THING;schedid;noask
Command forwarded to batchman for SYDNEY#GBJ_SIGNAL[(0012 12/21/07),
   (0AAAAAAAAAAAAEI6)].B_DO_THE_BAD_THING
============================================================
============= Action on RUN Branch =============
Performing action RELEASE on job SYDNEY#0AAAAAAAAAAAAEI6.G_DO_THE_GOOD_THING
Releasing of job SYDNEY#0AAAAAAAAAAAAEI6.G_DO_THE_GOOD_THING is NOT NECESSARY,
   because priority=10
============================================================
============= Statistics of branch job BRANCH_1 =============
TRUE: Searched for SUCC parent. Status of PARENT JOB(SIGNAL_1) is SUCC.
For action CANCEL - RUN_BRANCH=G_DO_THE_GOOD_THING and
   STOP_BRANCH=B_DO_THE_BAD_THING
BRANCH selected to STOP: B_DO_THE_BAD_THING
BRANCH selected to CONTINUE: G_DO_THE_GOOD_THING
CANCELED_JOBS: B_DO_THE_BAD_THING
PAUSED_JOB:
RELEASED_JOB:
============ END of branch job BRANCH_1 ==========
```

Paramètres d'entrée obligatoires

Le tableau 130 affiche les paramètres requis pour le travail de signal. Le travail de branche suivant ne gère pas de paramètre.

Tableau 130. Paramètres d'entrée pour le scénario d'action de signal

Nom du paramètre	Valeur du paramètre
ACTION_SWITCH	SIGNAL

La définition de paramètre ressemble à celle de l'exemple suivant. Le texte est saisi dans la zone de commentaires de la définition du flot de travaux.

Remarque : Cette définition indique également certains paramètres qui sont requis *uniquement* lorsque vous utilisez le travail de signal pour une recherche de modèle dans le journal de travail parent, ce qui constitue le principal avantage du scénario de signal. Vous l'utilisez pour exécuter l'analyse syntaxique complexe des résultats du travail, enregistrer l'heure et prendre une décision en fonction des informations collectées.

```
SIGNAL_1-BEGIN
CONDITION_SWITCH=COMPLEX
PATTERN_1=completed successfully
ACTION_SWITCH=SIGNAL
SIGNAL_1-END
```

Pour une description sur la manière de spécifier les paramètres pour un travail de branche, voir «Indication des paramètres de travail de branche», à la page 840.

Placement des travaux de branche dans le flot de travaux

Insérez le travail de signal après que le travail est évalué. Attribuez au travail de signal un nom incluant la chaîne `SIGNAL` et une valeur numérique (dans ce scénario, SIGNAL_1 est utilisé).

Le travail de branche suivant est le successeur immédiat du travail de signal. Attribuez-lui un nom avec le suffixe approprié : si c'est le premier travail de branche dans le flot de travaux, le nom peut être BRANCH_1.

Les travaux enfant du travail de branche doivent être nommés selon les instructions communes :

- Le nom de l'enfant correct doit commencer par "G_".
- Le nom de l'enfant incorrect doit commencer par "B_".

Utilisation du travail de branche

Vous pouvez utiliser le travail générique de branche dans votre environnement Tivoli Workload Scheduler.

Pour comprendre comment utiliser les travaux génériques de branche, voir :

- «Configuration requise pour exécuter les travaux de branche sous Windows»
- «Définition du travail de branche et du travail de signal dans la base de données», à la page 837
- «Placement du travail de branche dans le flot de travaux», à la page 839
- «Utilisation du travail ABEND», à la page 840

Configuration requise pour exécuter les travaux de branche sous Windows

Pour exécuter le travail générique de branche sur les systèmes d'exploitation Windows, vérifiez que la configuration système suivante est remplie.

- Etant donné que les systèmes d'exploitation Windows ne peuvent pas de façon native interpréter les scripts de shell UNIX, vous devez installer un interpréteur de shell pour utiliser le script de shell *branch.sh* sur le gestionnaire de domaine maître windows.
- Le répertoire `C:\cygwin\bin` doit pointer vers le sous-répertoire bin de votre répertoire d'installation Cygwin. Si vous avez installé Cygwin dans un répertoire différent du répertoire par défaut, utilisez le chemin correspondant.

Définition du travail de branche et du travail de signal dans la base de données

Pour définir les travaux de signal de branche dans la base de données de Tivoli Workload Scheduler vous pouvez utiliser la console Dynamic Workload Console ou le Composer.

Définition de deux travaux pointant vers un script de shell

Pour exécuter un scénario de travail de branche, vous devez définir un travail de branche. Pour exécuter un scénario de signal, vous devez également définir un travail de signal. Les deux travaux indiquent le même fichier de script de shell à la différence que :

- Pour le travail de branche, vous définissez l'`option de reprise` sur STOP.
- Pour le travail de signal, vous définissez l'`option de reprise` sur CONTINUER.

Vous avez deux travaux différents qui pointent vers le même fichier de script de shell afin d'éviter le placement incorrect du travail de branche ou du travail de signal dans le flot de travaux. Le travail de branche vérifie s'il est correctement placé dans le flot de travaux ; sinon, il se termine de façon anormale. Pour empêcher les successeurs de s'exécuter, le travail de branche a l'`option de reprise` définie sur STOP.

Le travail de signal exécute la même vérification ; s'il est mal placé dans le flot de travaux, le travail renvoie un message d'erreur et s'arrête avec un code retour différent de zéro. Même si l'`option de reprise` est définie sur Continuer, le travail ne libère pas ses successeurs de la dépendance car il a l'indicateur `Confirmation requise` défini dans la définition de flot de travaux. Le travail de signal reste, par conséquent, à l'état PEND en attendant que vous vérifiez son journal des travaux.

Pour plus d'informations, voir «Scénario d'action de signal», à la page 833.

Définition du travail de branche à l'aide de Dynamic Workload Console

Pour définir un travail de branche dans la base de données Tivoli Workload Scheduler à l'aide de la console Dynamic Workload Console, procédez comme suit :

1. Ouvrez Workload Designer.
2. Cliquez sur **Nouveau** -> **Définition de travail** -> **Natif** et, selon votre système d'exploitation, **UNIX** ou **Windows**.
3. Dans le panneau Général, entrez des valeurs dans les zones suivantes :
   - Nom : **BRANCHE**
   - Poste de travail : votre gestionnaire de domaine maître
   - Connexion : nom de l'utilisateur qui soumet et exécute le travail sur le gestionnaire de domaine maître
4. Dans le panneau des tâches :
   - Cochez le bouton radio **Script**.
   - Selon votre système d'exploitation :

     UNIX Dans la zone de tâche, indiquez le chemin complet vers le script de shell `branch.sh`. Utilisez les barres obliques (/).

Windows

Dans la zone de tâche, indiquez deux chemins :

- Le premier chemin pointe vers Cygwin : indiquez le chemin complet vers l'exécutable bash de Cygwin. Utilisez la notation standard Windows, c'est-à-dire les barres obliques inversées (\) comme séparateurs de répertoire.
- Le second chemin pointe vers le script de shell : indiquez le chemin complet vers le script de shell `branch.sh`. Utilisez les barres obliques (/). Si le chemin contient des espaces, insérez-le entre guillemets.

Par exemple, la valeur de la zone tâche peut être :

```
c:\cygwin\bin\bash\ "c:/Program Files/IBM/twa/scripts/branch.sh"
```

5. Sur le panneau des options de reprise, définissez l'action sur **Stop**.

Remarque : Vous pouvez utiliser les paramètres de Tivoli Workload Scheduler comme vous le feriez dans une autre définition de travail.

Définition du travail de signal à l'aide de Dynamic Workload Console

Pour définir un travail de signal dans la base de données Tivoli Workload Scheduler à l'aide de la console Dynamic Workload Console, procédez comme suit :

1. Ouvrez Workload Designer.
2. Cliquez sur **Nouveau** -> **Définition de travail** -> **Natif** et, selon votre système d'exploitation, **UNIX** ou **Windows**.
3. Dans le panneau Général, entrez des valeurs dans les zones suivantes :
   - Nom : **SIGNAL**
   - Poste de travail : votre gestionnaire de domaine maître
   - Connexion : nom de l'utilisateur qui soumet et exécute le travail sur le gestionnaire de domaine maître
4. Dans le panneau des tâches :
   - Cochez le bouton radio **Script**.
   - Selon votre système d'exploitation :

     UNIX Dans la zone de tâche, indiquez le chemin complet vers le script de shell `branch.sh`. Utilisez les barres obliques (/).

     Windows

     Dans la zone de tâche, indiquez deux chemins :

     - Le premier chemin pointe vers Cygwin : indiquez le chemin complet vers l'exécutable bash de Cygwin. Utilisez la notation standard Windows, c'est-à-dire les barres obliques inversées (\) comme séparateurs de répertoire.
     - Le second chemin pointe vers le script de shell : indiquez le chemin complet vers le script de shell `branch.sh`. Utilisez les barres obliques (/). Si le chemin contient des espaces, insérez-le entre guillemets.

     Par exemple, la valeur de la zone tâche peut être :

     ```
     c:\cygwin\bin\bash\ "c:/Program Files/IBM/twa/scripts/branch.sh"
     ```

5. Sur le panneau des options de reprise, définissez l'action sur **Continuer**.

Définition du travail de branche et du travail de signal à l'aide du Composer

Pour définir un travail de branche et un travail de signal dans la base de données Tivoli Workload Scheduler à partir du Composer, procédez comme suit :

1. Connectez -vous au gestionnaire de domaine maître en tant qu'utilisateur disposant des droits permettant d'ajouter des travaux.
2. Suivant votre système d'exploitation, créez une définition de travail comme dans les exemples suivants :

   UNIX

   ```
   $JOBS
   SYDNEY#BRANCH
   SCRIPTNAME "/opt/ibm/tws_scripts/branch.sh"
   STREAMLOGON tws
   TASKTYPE UNIX
   RECOVERY STOP

   SYDNEY#SIGNAL
   SCRIPTNAME "/opt/ibm/tws_scripts/branch.sh"
   STREAMLOGON tws
   TASKTYPE UNIX
   RECOVERY CONTINUE
   ```

   Windows

   ```
   $JOBS
   HELSINKI#BRANCH
   SCRIPTNAME "c:\cygwin\bin\bash \"c:/Program Files/IBM/tws/
                                          scripts/branch.sh\""
   STREAMLOGON tws
   TASKTYPE WINDOWS
   RECOVERY STOP

   $JOBS
   HELSINKI#BRANCH
   SCRIPTNAME "c:\cygwin\bin\bash \"c:/Program Files/IBM/tws/
                                          scripts/branch.sh\""
   STREAMLOGON tws
   TASKTYPE WINDOWS
   RECOVERY CONTINUE
   ```

Placement du travail de branche dans le flot de travaux

Vous pouvez placer le travail générique de branche dans le flot de travaux à l'aide de la console DWC.

Règles générales

Pour placer le travail de branche dans le flot de travaux correctement, vérifiez les éléments suivants :

- Vous disposez d'une seule définition de travail dans la base de données Tivoli Workload Scheduler, comme décrit dans «Définition du travail de branche et du travail de signal dans la base de données», à la page 837.
- Lorsque vous insérez le travail de branche dans le flot de travaux, vous lui attribuez un alias décrivant le fichier du travail de branche dans le flot de travaux.
- Nommez le premier travail de branche BRANCH_1, le deuxième BRANCH_2, et ainsi de suite.

Règles valides pour tous les scénarios à l'exception de l'action de signal

Les règles suivantes s'appliquent à tous les scénarios d'utilisation de travail de branche, à l'exception de l'action SIGNAL :

- Le travail de branche doit posséder un seul prédécesseur.
- Le travail de branche doit comporter exactement deux travaux enfant, nommés comme suit :
  - Le nom de l'enfant correct doit commencer par `G_`.
  - Le nom de l'enfant incorrect doit commencer par `B_`.

Règles valides pour le scénario d'action de signal

Les règles suivantes s'appliquent à l'action SIGNAL :

- Vous devez utiliser deux travaux contrôlés séquentiellement, comme suit :
  - Travail de signal
  - Travail de branche
- Le travail de signal ne doit posséder qu'un enfant, représentant le travail de branche suivant.
- Pour le travail de signal, vous devez définir le paramètre d'entrée `ACTION_SWITCH=SIGNAL`.
- Pour le travail de signal, vous devez définir l'indicateur `Confirmation requise` en modifiant les propriétés du travail de branche.
- Le travail de signal doit posséder un seul prédécesseur.
- Le travail de branche doit être conforme aux règles décrites dans «Règles valides pour tous les scénarios à l'exception de l'action de signal ».

Utilisation du travail ABEND

Utilisez le travail ABEND lorsque vous souhaitez inclure l'état ABEND du statut du flot de travaux.

Si un événement incorrect se produit dans votre flot de travaux et qu'il est détecté par le travail de branche, vous pouvez ainsi propager ce *statut incorrect* au niveau du flot de travaux en insérant le travail ABEND dans la branche incorrecte qui suit le travail de branche.

Le travail ABEND émet uniquement la commande `exit 1`, qui entraîne la fin anormale du travail et garantit que le statut de fin anormale est répercuté au flot de travaux. Aucun travail terminé précédemment de façon anormale ne propage son état au flot de travaux car son option de reprise est réglée sur Continuer (nécessaire pour que le travail de branche s'exécute).

Pour obtenir un exemple de travail ABEND, voir «Placement du travail de branche dans le flot de travaux», à la page 839.

Indication des paramètres de travail de branche

Indiquez les paramètres de travail de branche dans la zone de commentaires du flot de travaux contenant le travail de branche concerné.

Remarque : Veillez à spécifier les paramètres dans la zone de commentaires, et pas la zone de description de l'éditeur de flot de travaux.

Chaque paramètre doit être encadré par un *séparateur de début* et un *séparateur de fin*, même si vous utilisez des paramètres pour un seul travail de branche. Les séparateurs sont construits comme suit :

Séparateur de début
Nom du travail de branche, suivi de la chaîne `-BEGIN`. Par exemple BRANCH_1-BEGIN.

Séparateur de fin
Nom du travail de branche, suivi de la chaîne `-END`, par exemple BRANCH_1-END.

L'exemple suivant illustre une définition de paramètre correspondant à une branche de travail :

```
BRANCH_1-BEGIN
CONDITION_SWITCH=COMPLEX
PATTERN_1=Error
NEGATE_CONDITION_RESULT_1=YES
PATTERN_2=Free space on Primary device
VALUE_2=50
ARITHMETICAL_OPERATOR_2=-gt
BOOLEAN_OPERATOR_2=&&
PATTERN_3=Free space on Secondary device
VALUE_3=60
ARITHMETICAL_OPERATOR_3=-gt
BOOLEAN_OPERATOR_3=||
BRANCH_1-END
```

L'exemple suivant illustre une définition de paramètre correspondant à un flot de travaux avec deux travaux de branche :

```
BRANCH_1-BEGIN
CONDITION_SWITCH=COMPLEX
PATTERN_1=Error
NEGATE_CONDITION_RESULT_1=YES
PATTERN_2=Free space on Primary device
VALUE_2=50
ARITHMETICAL_OPERATOR_2=-gt
BOOLEAN_OPERATOR_2=&&
PATTERN_3=Free space on Secondary device
VALUE_3=60
ARITHMETICAL_OPERATOR_3=-gt
BOOLEAN_OPERATOR_3=||
BRANCH_1-END
BRANCH_2-BEGIN
CONDITION_SWITCH=COMPLEX
PATTERN_1=Backup of Primary device
VALUE_1=ERROR
NEGATE_CONDITION_RESULT_1=YES
BRANCH_2-END
```

Le nom des travaux de signal doit commencer par la chaîne `SIGNAL`. Les séparateurs de paramètre sont également requis :

```
SIGNAL_1-BEGIN
ACTION_SWITCH=SIGNAL
SIGNAL_1-END
```

Référence de paramètres

Syntaxe et signification des paramètres de travail de branche.

Types de paramètre

Il existe deux types de paramètre :

Fixe Vous ne pouvez indiquer des paramètres fixes qu'une seule fois dans un travail.

Indexé
Vous pouvez spécifier des paramètres indexés plusieurs fois dans un travail, ou ne pas les indiquer du tout.

Paramètres fixes

Vous pouvez indiquer les paramètres fixes suivants :

CONDITION_SWITCH
Type de condition qui s'exécute sur le travail parent. Il peut prendre l'une des valeurs suivantes :

PARENT_SUCCESS
La condition est TRUE lorsque le parent se termine à l'état SUCC. Il s'agit de la valeur par défaut. Pour obtenir une description de cette condition, voir «Scénarios basés sur le type de condition», à la page 797.

PARENT_ABEND
La condition est TRUE lorsque le parent se termine à l'état ABEND. Pour obtenir une description de cette condition, voir «Fin anormale du parent», à la page 806.

COMPLEX
La condition est générée à partir d'une ou plusieurs sous-conditions connectées avec les opérateurs booléens (AND ou OR) et est évaluée avec la logique booléenne. Pour obtenir une description de cette condition, voir «Scénario complexe - conditions multiples», à la page 822.

ACTION_SWITCH
Action à effectuer sur la branche d'arrêt. Il peut prendre l'une des valeurs suivantes :

CANCEL
La branche d'arrêt est annulée. Il s'agit de la valeur par défaut.

PAUSE
La branche d'arrêt est mise en pause. Pour obtenir une description de cette condition, voir «Scénario d'actions de pause et de libération», à la page 826.

SIGNAL
Aucune branche n'est annulée. Une recommandation nécessitant votre confirmation est stockée dans le journal de travail. Pour obtenir une description de cette condition, voir «Scénario d'action de signal», à la page 833.

Paramètres indexés

Utilisez les paramètres indexés uniquement avec le paramètre CONDITION_SWITCH=COMPLEX indiqué (sinon, ils sont ignorés lors du processus de branchement).

Lorsque vous définissez CONDITION_SWITCH=COMPLEX, le travail de branche évalue une condition complexe (une condition avec une ou plusieurs sous-conditions). Chaque sous-condition possède son propre *index* qui commence par le numéro 1 et est incrémentiel.

Pour chaque sous-condition, vous devez définir au moins un paramètre Pattern_*i*, où *i* représente la relation entre la sous-condition et le paramètre correspondant. Le suffixe ressemble à *paramètre_indexé_i*, où *_i* est le suffixe. Par exemple, les trois paramètres suivants appartiennent à la même sous-condition, qui est la deuxième dans la condition complexe ; leur affinité est exprimée par le suffixe _2, qui représente l'index de la sous-condition :

```
PATTERN_2=find this row and number in this row
VALUE_2=50
ARITHMETICAL_OPERATOR_2=-lt
```

Regroupez les paramètres associés à une sous-condition en utilisant le même index. Les sous-conditions sont évaluées séparément, puis connectées les unes aux autres à l'aide des opérateurs booléens. La *condition complexe* est ensuite évaluée.

Exemples d'utilisation d'index

L'exemple suivant présente les paramètres à indiquer si vous souhaitez rechercher trois modèles dans le journal du travail parent. l'index sert de *compteur incrémentiel*. Les paramètres appartiennent à trois sous-conditions distinctes :

```
PATTERN_1=premier modèle à trouver
PATTERN_2=deuxième modèle à trouver
PATTERN_3=troisième modèle à trouver
```

Utilisez la syntaxe suivante pour définir les paramètres liés à même sous-condition (par exemple, recherche d'un modèle, puis d'un numéro sur la même ligne et la comparaison arithmétique ultérieure). Vous indiquez le numéro et l'opérateur arithmétique comme paramètres d'entrée.

```
PATTERN_1=Free space on Primary device
VALUE_1=50
ARITHMETICAL_OPERATOR_1=-gt
```

Signification des paramètres indexés

la liste suivante décrit les paramètres indexés et leurs valeurs possibles. Vous pouvez utiliser tous ces paramètres pour créer une seule sous-condition.

PATTERN_*i*

Recherche d'un modèle de texte (par exemple, `ended successfully`). Si le modèle est trouvé, le résultat de la condition est TRUE.

Pour le paramètre PATTERN_*i*, vous pouvez indiquer les paramètres supplémentaires suivants. Si PATTERN_*i* n'est pas spécifié, ils sont ignorés.

VALUE_*i*

STRING ou NUMERIC. Cela est déterminé automatiquement en lisant le paramètre de valeur particulière lors du démarrage du travail de branche.

La valeur spécifiée par VALUE_*i* est recherchée *dans la même ligne* identifiée par la recherche de la chaîne indiquée par PATTERN_*i*.

Deux types de valeurs sont possibles et sont déterminés automatiquement lors de l'analyse syntaxique du contenu du paramètre VALUE_*i* :

Valeur de chaîne
Recherche d'un autre modèle de texte dans la même ligne.

Si les deux modèles sont présents dans la même ligne, le résultat de la condition est TRUE.

Pour obtenir une description de cette fonction, voir«Branche complexe - modèle dans la ligne de modèle», à la page 814.

Valeur numérique
Recherches de la valeur numérique dans la même ligne.

L'opérateur arithmétique que vous avez indiqué est ensuite utilisé pour exécuter la comparaison arithmétique. Si la comparaison arithmétique réussie, la condition est TRUE. Un opérateur arithmétique spécifique est définie pour chaque valeur numérique.

Pour obtenir une description de cette fonction, voir«Branche complexe - comparaison d'une valeur numérique», à la page 819.

ARITHMETICAL_OPERATOR_*i*
Opérateur utilisé pour la comparaison arithmétique.

NEGATE_CONDITION_RESULT_*i*
Cet argument annule le résultat de la sous-condition particulière, ce qui signifie qu'il permute le résultat TRUE ou FALSE de la sous-condition.

BOOLEAN_OPERATOR_*i*
Les sous-conditions définies sont liées les unes aux autres par l'opérateur booléen AND ou OR. Vous pouvez utiliser l'opérateur booléen car `i=2`. Cela signifie que l'index de l'opérateur booléen doit être d'au moins 2.

Par exemple, vous avez deux paramètres dans la liste. Chacun d'eux représente une sous-condition. Chaque sous-condition est évaluée séparément et le résultat est renvoyé comme TRUE ou FALSE. Pour évaluer la condition entière, vous devez joindre les résultats particuliers ensemble.

La signification de BOOLEAN_OPERATOR_*i* est que le résultat de la connexion de *cette* sous-condition avec le résultat de la sous-condition *précédente* utilise l'opérateur booléen AND ou OR.

Tables de référence

Le tableau 131 décrit les paramètres, leurs valeurs possibles, et les valeurs par défaut.

Tableau 131. Paramètres et valeurs

Nom du paramètre	Valeurs possibles	Valeur par défaut
CONDITION_SWITCH	PARENT_SUCCESS PARENT_ABEND COMPLEXE	PARENT_SUCCESS

Tableau 131. Paramètres et valeurs (suite)

Nom du paramètre	Valeurs possibles	Valeur par défaut
ACTION_SWITCH	CANCEL PAUSE SIGNAL	CANCEL (pour les travaux de branche) SIGNAL (pour les travaux de signal)
PATTERN_*i*, où *i* est l'index incrémentiel	Toute chaîne de caractères	
VALUE_*i*, où *i* est l'index incrémentiel	Toute chaîne de caractères Toute valeur numérique (entier ou réel)	
ARITHMETICAL_OPERATOR_*i*, où *i* est l'index incrémentiel	-lt -le -eq -ne -ge -gt	-eq
IS_CASE_SENSITIVE_*i*, où *i* est l'index incrémentiel	OUI NO	OUI
IS_REGULAR_EXPRESSION_*i*, où *i* est l'index incrémentiel	OUI NO	NO
NEGATE_CONDITION_RESULT_*i*, où *i* est l'index incrémentiel	OUI NO	NO
BOOLEAN OPERATOR_*i*, où *i* est l'index incrémentiel	&& ||	&&

Les valeurs pour les opérateurs arithmétiques et booléens utilisent la syntaxe UNIX. Leurs significations sont décrites dans tableau 132.

Tableau 132. Description des opérateurs arithmétiques

Valeur UNIX du paramètre	Interprétation de la valeur du paramètre	Signification de la valeur du paramètre
-lt	<	Inférieur à
-le	<=	Inférieur ou égal à
-eq	=	Egal à
-ne	!=	Différent de
-ge	>=	Supérieur ou égal à
-gt	>	Supérieur à
&&	ET	ET logique
||	OU	OU logique

Respect de la casse

Les règles suivantes s'appliquent lorsque vous indiquez un paramètre :

- Les noms de paramètre ne sont pas sensibles à la casse.
- Les valeurs des paramètres fixes ne sont pas sensibles à la casse.
- Les modèles à rechercher dans le journal de travail parent sont sensibles à la casse. Pour remplacer ce comportement, indiquez IS_CASE_SENSITIVE_*i*=NO, où *i* représente l'index de sous-condition en cours. Par exemple, pour le paramètre suivant :

  ```
  PATTERN_2=text
  ```

  Vous permutez la recherche de modèle sensible à la casse en spécifiant ce paramètre supplémentaire :

  ```
  IS_CASE_SENSITIVE_2=NO
  ```

  Si vous recherchez deux modèles dans la même ligne, ce paramètre les affecte tous les deux.
- Les séparateurs de paramètre requis pour définir le travail de branche sont sensibles à la casse. Vous devez spécifiez les séparateurs en majuscules, sinon ils ne sont pas lu et les valeurs par défaut sont utilisées.

Exemples de statut d'exemple

Les sections ci-dessous récapitulent la construction de jeux de paramètres simples ou plus complexes.

Branche simple, scénarios de branche longue

Les valeurs par défaut suivantes sont utilisées :

- CONDITION_SWITCH=PARENT_SUCCESS
- ACTION_SWITCH=CANCEL

Vous n'avez pas besoin d'indiquer de paramètre d'entrée pour ces scénarios. Les valeurs par défaut sont fournies automatiquement par le travail de branche.

Pause

Cette fonction requiert au moins un paramètre fixe. Vous devez utiliser au moins deux travaux de branche d'implémenter pour implémenter le scénario de pause et de libération.

Pour utiliser la fonction de pause et de libération, vous devez remplacer le comportement par défaut pour le premier travail de branche. Le nom du travail de branche dans cet exemple est BRANCH_1. Si le nom de votre travail de branche a un suffixe différent (par exemple, BRANCH_5), vous devez ajuster les noms du séparateur.

Le second travail de branche dans l'approche de mise en pause et de libération ne nécessite aucun paramètre d'entrée s'il dépend uniquement du statut de son parent (SUCC).

Voici la syntaxe complète avec les séparateurs de paramètre :

```
BRANCH_1-BEGIN
ACTION_SWITCH=PAUSE
BRANCH_1-END
```

Aucun paramètre indexé n'est fourni. Les paramètres répertoriés peuvent être utilisés uniquement en combinaison avec le paramètre fixe CONDITION_SWITCH=COMPLEX. Du fait que le paramètre CONDITON_SWITCH n'est pas inclus, la valeur par défaut CONDITION_SWITCH=PARENT_SUCCESS a été utilisée.

Pour plus d'informations sur la combinaison de l'action de pause avec l'état complexe, voir «Etat complexe avec l'action de pause», à la page 850.

Recherche simple de modèle

Cette fonction est représentée par une sous-condition et elle a besoin d'un paramètre fixe et d'un paramètre indexé.

L'exemple suivant affiche la syntaxe complète avec les séparateurs de paramètres. Le nom du travail de branche est BRANCH_1. Si le nom de votre travail de branche a un suffixe différent (par exemple, BRANCH_5), vous devez ajuster les noms du séparateur.

```
BRANCH_1-BEGIN
CONDITION_SWITCH=COMPLEX
PATTERN_1=find this text
BRANCH_1-END
```

Il n'y existe qu'un paramètre répertorié représentant la seule sous-condition.

Recherche de modèle annulée

Cette fonction est représentée par une sous-condition et elle a besoin d'un paramètre fixe et deux paramètres indexés.

L'exemple suivant affiche la syntaxe complète avec les séparateurs de paramètres. Le nom du travail de branche est BRANCH_1. Si le nom de votre travail de branche a un suffixe différent (par exemple, BRANCH_5), vous devez ajuster les noms du séparateur.

```
BRANCH_1-BEGIN
CONDITION_SWITCH=COMPLEX
PATTERN_1=do not find this text
NEGATE_CONDITION_RESULT_1=YES
BRANCH_1-END
```

Les deux paramètres indexés ont le même suffixe car ils appartiennent à la même sous-condition.

Recherche multiple de modèle

Cette fonction est représentée par plusieurs sous-conditions et elle a besoin d'un paramètre fixe et de plusieurs paramètres indexés.

Dans cet exemple, vous trouverez deux modèle indépendants dans le journal du travail parent. Vous créez deux sous-conditions distinctes, chacune avec son propre index(1 et 2). Vous devez également indiquer si *les deux* modèles doivent être recherché (à l'aide de l'opérateur booléen *AND*) ou si le fait de trouver *au moins un* modèle est suffisant (opérateur booléen *OR*).

L'exemple suivant affiche la syntaxe complète avec les séparateurs de paramètres. Le nom du travail de branche est BRANCH_1. Si le nom de votre travail de branche a un suffixe différent (par exemple, BRANCH_5), vous devez ajuster les noms du séparateur.

```
BRANCH_1-BEGIN
CONDITION_SWITCH=COMPLEX
PATTERN_1=find this text
PATTERN_2=find also this text
BOOLEAN_OPERATOR_2=&&
BRANCH_1-END
```

Les paramètres indexés possèdent des suffixes différents.

Le paramètre PATTERN_1 appartient à la première sous-condition et le paramètre PATTERN_2 appartient à la deuxième sous-condition. Le paramètre BOOLEAN_OPERATOR_2 indique comment la deuxième sous-condition est connectée à la première sous-condition.

Modèle dans une recherche de modèle

Cette fonction est représentée par une sous-condition et elle a besoin d'un paramètre fixe et deux paramètres indexés.

Dans cet exemple, vous recherchez le modèle de texte dans le journal du travail parent. Vous recherchez ensuite un autre modèle sur la même ligne. Tous les paramètres indexés appartiennent à une même sous-condition.

L'exemple suivant affiche la syntaxe complète avec les séparateurs de paramètres. Le nom du travail de branche est BRANCH_1. Si le nom de votre travail de branche a un suffixe différent (par exemple, BRANCH_5), vous devez ajuster les noms du séparateur.

```
BRANCH_1-BEGIN
CONDITION_SWITCH=COMPLEX
PATTERN_1=find this text
VALUE_1=and also this text on the same row
BRANCH_1-END
```

Les deux paramètres indexés ont le même suffixe car ils appartiennent à la même sous-condition.

Recherche de modèle avec comparaison numérique

Cette fonction est représentée par une sous-condition et elle a besoin d'un paramètre fixe et trois paramètres indexés.

Vous recherchez un modèle de texte dans le journal du travail parent. Vous recherchez ensuite un nombre sur la même ligne. Vous souhaitez comparer ce numéro avec un autre numéro que nous fournissons comme paramètre d'entrée. Pour la comparaison arithmétique, vous utilisez l'opérateur arithmétique spécifié comme paramètre d'entrée.

Tous les paramètres indexés appartiennent à une même sous-condition.

L'exemple suivant affiche la syntaxe complète avec les séparateurs de paramètres. Le nom du travail de branche est BRANCH_1. Si le nom de votre travail de branche a un suffixe différent (par exemple, BRANCH_5), vous devez ajuster les noms du séparateur.

```
BRANCH_1-BEGIN
CONDITION_SWITCH=COMPLEX
PATTERN_1=Total backup size
VALUE_1=500
ARITHMETICAL_OPERATOR_1=-lt
BRANCH_1-END
```

Les trois paramètres indexés ont le même suffixe car ils appartiennent à la même sous-condition.

La signification des fonctions appelées par les paramètres spécifiés peut être représentée comme suit :

1. Obtenir le journal de travail parent.
2. Extraire la ligne contenant la chaîne `Total backup size`.
3. Si (ligne trouvée) :
4. Alors passer à l'étape suivante
5. Sinon retourner FALSE
6. Tenter d'extraire la valeur numérique de la ligne.
7. Si (nombre trouvé) :
8. Alors passer à l'étape suivante
9. Sinon retourner FALSE
10. Si (number_from_joblog < 50) :
11. Alors retourner TRUE
12. Sinon retourner FALSE

Combinaison de recherche de modèle et d'action de pause

Une condition autre qu'une condition par défaut peut être associée à une action autre qu'une action par défaut. Cela signifie que nous utilisons des entrées pour CONDITION_SWITCH et ACTION_SWITCH dans ce scénario.

Nous expliquons comment combiner une recherche de modèle unique et l'action de pause.

Cette fonction est représentée par une sous-condition et elle a besoin de deux paramètres fixes et d'un paramètre indexé.

L'exemple suivant affiche la syntaxe complète avec les séparateurs de paramètres. Le nom du travail de branche est BRANCH_1. Si le nom de votre travail de branche a un suffixe différent (par exemple, BRANCH_5), vous devez ajuster les noms du séparateur.

```
BRANCH_1-BEGIN
CONDITION_SWITCH=COMPLEX
ACTION_SWITCH=PAUSE
PATTERN_1=find this text
BRANCH_1-END
```

Les deux paramètres fixes sont remplacés. Un paramètre indexé appartient à une seule sous-condition.

Action de signal

Le scénario de signal requiert des séparateurs différents.

Deux travaux gèrent le flux de flot de travaux :

- Travail de signal
- Travail de branche

Seul le signal utilise les paramètres d'entrée.

Le nom du travail de signal se compose du mot SIGNAL et du suffixe, ce qui signifie que les séparateurs de paramètre sont *différents* les uns des autres. De plus, pour le travail de signal, les règles communes sont valides. Les séparateurs de paramètre doivent ressembler au nom de travail connexe.

Ce scénario illustre la différence ainsi que les paramètres de l'action de signal. L'action de signal est combinée à la condition en fonction de la recherche de modèle dans le journal du travail parent.

La fonction de signal est représentée par un paramètre fixe. La recherche de modèle requiert un paramètre fixe pour le type de condition et un paramètre indexé indiquant la recherche de modèle.

L'exemple suivant affiche la syntaxe complète avec les séparateurs de paramètres. Le nom du travail de branche est BRANCH_1. Si le nom de votre travail de branche a un suffixe différent (par exemple, BRANCH_5), vous devez ajuster les noms du séparateur.

```
SIGNAL_1-BEGIN
CONDITION_SWITCH=COMPLEX
ACTION_SWITCH=SIGNAL
PATTERN_1=find this text
SIGNAL_1-END
```

Etat complexe avec l'action de pause

La condition complexe peut être associée à l'action explicite.

Pour plus d'informations sur la condition complexe, voir «Scénario complexe - conditions multiples», à la page 822. Dans ce scénario, l'état complexe est associé à l'action de pause.

L'exemple suivant affiche la syntaxe complète avec les séparateurs de paramètres. Le nom du travail de branche est BRANCH_1. Si le nom de votre travail de branche a un suffixe différent (par exemple, BRANCH_5), vous devez ajuster les noms du séparateur.

```
BRANCH_1-BEGIN
CONDITION_SWITCH=COMPLEX
ACTION_SWITCH=PAUSE
PATTERN_1=Error
NEGATE_CONDITION_RESULT_1=YES
PATTERN_2=Free space on Primary device
VALUE_2=50
ARITHMETICAL_OPERATOR_2=-gt
BOOLEAN_OPERATOR_2=&&
PATTERN_3=Free space on Secondary device
```

```
VALUE_3=60
ARITHMETICAL_OPERATOR_3=-gt
BOOLEAN_OPERATOR_3=||
BRANCH_1-END
```

La signification de la définition se présente ainsi :

- Subcondition_1 : si vous trouvez le modèle `Error`, renvoyez FALSE, ou TRUE dans le cas contraire. Le résultat inversé est obtenu grâce au paramètre NEGATE_CONDITION_RESULT_1.
- Subcondition_3 : recherchez la ligne contenant `Free space on Primary device`. Extrayez le nombre de cette ligne. Si ce nombre est supérieur à 50, renvoyez TRUE, FALSE dans le cas contraire.
- Subcondition_3 : recherchez la ligne contenant `Free space on Secondary device`. Extrayez le nombre de cette ligne. Si ce nombre est supérieur à 60, renvoyez TRUE, FALSE dans le cas contraire.
- Placez ces trois sous-conditions à l'aide d(opérateurs booléens de sorte que la condition complexe soit générée comme suit :
  1. If (subcondition_1 = TRUE) AND (subcondition_2=TRUE) OR (subcondition_3=FALSE)
  2. Then return TRUE
  3. Else return FALSE
  4. If (complex_condition_result=TRUE)
  5. Then CANCEL bad_branch
  6. Else PAUSE the good_branch

Pour plus d'informations sur les concepts de pause et de libération, voir «Scénario d'actions de pause et de libération», à la page 826.

Travaux de branches multiples dans un même flot de travaux

Vous pouvez définir des paramètres pour plusieurs travaux de branche définis dans le même flot de travaux. Le fait de posséder de plusieurs travaux de branche dans un même flot de travaux signifie que les jeux de paramètres différents sont enserrés par des *séparateurs* différents.

L'exemple suivant affiche la syntaxe complète avec les séparateurs de paramètres. Les noms des travaux de branche sont BRANCH_1 et BRANCH-2. Si le nom de vos travaux de branche ont des suffixes différents (par exemple, BRANCH_5 et BRANCH_6), vous devez ajuster les noms du séparateur.

```
BRANCH_1-BEGIN
CONDITION_SWITCH=COMPLEX
PATTERN_1=Error
NEGATE_CONDITION_RESULT_1=YES
PATTERN_2=Free space on Primary device
VALUE_2=50
ARITHMETICAL_OPERATOR_2=-gt
BOOLEAN_OPERATOR_2=&&
PATTERN_3=Free space on Secondary device
VALUE_3=60
ARITHMETICAL_OPERATOR_3=-gt
BOOLEAN_OPERATOR_3=||
BRANCH_1-END

BRANCH_2-BEGIN
CONDITION_SWITCH=COMPLEX
```

```
PATTERN_1=Backup of Primary device
VALUE_1=ERROR
NEGATE_CONDITION_RESULT_1=YES
BRANCH_2-END
```

Remarques importantes sur le travail de branche

Cette section met en évidence certaines remarques et suppositions sur la conception du travail de branche.

- Le nom du travail de branche dans la base de données est BRANCH. La propriété Recovery options=STOP doit être définie par le travail.
- Le nom du travail de signal dans la base de données est SIGNAL. La propriété Recovery options=CONTINUE doit être définie par le travail.
- Le nom du travail de branche inséré dans le flot de travaux doit être constitué du nom du travail de branche (BRANCH) et du *suffixe*. Le suffixe correspond à la position du travail de branche dans le flot de travaux. Par exemple, BRANCH_3 est le troisième travail de branche dans le flot de travaux.
- Le nom du travail de signal inséré dans le flot de travaux doit être constitué du nom du travail de signal (SIGNAL) et du suffixe. Le suffixe correspond à la position du travail de signal dans le flot de travaux. Par exemple, SIGNAL_3 est le troisième travail de signal dans le flot de travaux.

  Le compteur de suffixe pour les travaux de signal est différent du compteur de suffixe des travaux de branche, c'est pourquoi les travaux BRANCH_3 et SIGNAL_3 peuvent exister dans un flot de travaux en même temps, même si leur suffixe est identique.
- Le travail de branche possède un seul parent. Cela signifie que le travail de branche doit posséder la dépendance de prédécesseur/successeur exactement définie sur un travail dans le même flot de travaux. Ceci est vérifié lors du démarrage du travail de branche. Si cette condition n'est pas satisfaite, le travail de branche se termine anormalement.

  La connexion du travail de branche à plusieurs parents est arrêtée par le travail de branche qui s'arrête anormalement.
- Pour le parent qui est évalué par rapport au statut du résultat (SUCC ou ABEND), vous devez définir la propriété suivante dans la définition de travail, et non dans la définition du flot de travaux :

  ```
  Recovery options=Continue
  ```
- Pour les scénarios d'utilisation de travail de branche suivants, le travail de branche doit comporter exactement deux travaux enfants :
  - ACTION_SWITCH=CANCEL (valeur par défaut)
  - ACTION_SWITCH=PAUSE
  - Les travaux enfants doivent être signalés comme enfant correct et enfant incorrect. Les travaux enfants doivent être nommés comme suit :
    - Le travail qui représente l'enfant correct doit avoir un nom commençant par "G_". Il n'est pas nécessaire de renommer la *définition du travail*, renommez l'*alias du travail* dans le flot de travaux.

      Par exemple, le nom du travail dans la base de données est START_BACKUP. Pour utiliser ce travail comme l'enfant correct du travail de branche, insérez-le dans le flot de travaux et dans le flot de travaux attribuez au travail l'alias G_START_BACKUP.
    - Le travail qui représente l'enfant incorrect doit avoir un nom commençant par "B_". Il n'est pas nécessaire de renommer la définition de travail, renommez l'alias du travail dans le flot de travaux.

Par exemple, le nom du travail dans la base de données est RESUME_DATABASE. Pour utiliser ce travail en tant qu'enfant incorrect du travail de branche, insérez ce travail dans le flot de travaux et dans le flot de travaux attribuez au travail l'alias B_RESUME_DATABASE.

Le nombre des travaux enfants et leurs préfixes corrects sont vérifiés au démarrage du travail de branche. Si ces conditions ne sont pas remplies, le travail de branche se termine anormalement.

- L'utilisation des guillemets n'est pas mise en oeuvre dans la version actuelle du travail de branche. Si vous spécifiez des guillemets dans les paramètres PATTERN_*i* ou VALUE_*i*, les guillemets sont automatiquement supprimés.
- Lors de l'évaluation de la condition complexe, le paramètre de recherche d'origine est un modèle de texte. Ce modèle texte est recherché dans le journal du travail parent. S'il existe plusieurs lignes comportant le modèle recherché, seule la première ligne est identifiée. Les autres lignes qui correspondent sont ignorées.

  Le contenu de la première ligne identifiée est évalué par une autre sous-condition, comme illustré par le modèle dans les scénarios de ligne de modèle et de comparaison de valeur numérique.
- Lors de l'extraction de la valeur numérique de la ligne de modèle, il ne peut y avoir qu'une valeur numérique dans la ligne. Les valeurs numériques de type entier et réel sont acceptées.

  La présence de plusieurs nombres dans la ligne de modèle identifiée engendrera l'extraction d'une valeur numérique incorrecte ; tous les nombres de la ligne seront concaténés et ne représenteront pas une valeur pertinente.
- Lorsque vous utilisez le scénario de signal, le travail de signal doit avoir exactement un enfant. L'enfant doit avoir un travail de branche consécutif et son nom doit être conforme aux conventions d'appellation du travail de branche générique.
- Lors de l'utilisation de la paire d'actions PAUSE et RELEASE, le processus est le suivant :
  - Le premier travail de branche définit la priorité du travail qui doit être mis en pause sur 0.
  - Le second travail de branche définit la priorité du travail qui doit être libéré sur 10.

  Cette fonction est codée. Il n'y a aucun mécanisme mis en oeuvre qui vous permettrait de vous "rappeler" la priorité du travail d'origine.
- Toutes les informations sur le flot de travaux, dans lequel le travail de branche s'exécute, sont extraites de la *base de données* à l'aide de la commande *composer* lors du démarrage du travail de branche. Cela signifie que le travail de branche ne reflète aucune instance de travail ayant été *soumise dans* le flot de travaux mais qui n'existe pas dans la définition du flot de travaux.

Annexe E. 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. Les principales fonctions d'accessibilité de ce produit permettent aux utilisateurs d'effectuer les opérations suivantes :

- Utilisation des technologies d'assistance, comme le logiciel de lecture d'écran et le synthétiseur de parole numérique, pour entendre la description de l'affichage. Pour plus de détails sur l'utilisation de ces technologies avec ce produit, consultez la documentation produit de la technologie d'assistance.
- Utilisation des fonctions spécifiques ou équivalentes à l'aide du clavier uniquement.
- Agrandissement des éléments affichés.

En outre, la documentation produit comprend maintenant des options pour une meilleure accessibilité :

- Entièrement disponible en HTML et PDF convertible, elle permet la plus large utilisation possible du logiciel de lecture d'écran.
- Toutes ses images sont fournies avec un texte secondaire pour que les utilisateurs visuellement déficients puissent en comprendre la teneur.

Navigation dans l'interface avec le clavier

Des touches de raccourci et de clavier standard sont utilisées par le produit et sont documentées par le système d'exploitation. Pour plus de détails, consultez la documentation du système d'exploitation.

Le panneau relatif à l'éditeur de règle d'événement est le seul panneau qui ne permet pas d'effectuer des opérations à l'aide du clavier uniquement et le sous-système de canaux ne peut pas être désactivé. Cependant, vous pouvez autrement effectuer toutes les opérations disponibles dans ce panneau en l'exécutant la commande composer à partir de l'interface de ligne de commande.

Agrandissement des éléments affichés

Vous pouvez agrandir les informations affichées dans les fenêtres du produit, à l'aide des fonctions des systèmes d'exploitation sur lesquels le produit est exécuté. Par exemple, dans un environnement Microsoft Windows, vous pouvez réduire la résolution de l'écran pour agrandir la taille des polices du texte affiché. Pour plus de détails, consultez la documentation du système d'exploitation.

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 IBM 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 un jeu de caractères double octet peuvent être obtenues par écrit auprès d'IBM à l'adresse suivante :

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

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 données qu'ils contiennent. 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 (IBM Customer Agreement), 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 ™), es symboles signalent des marques d'IBM aux Etats-Unis à la date de publication de ce document. Ces marques peuvent également exister et éventuellement avoir été enregistrées dans d'autres pays. La liste actualisée de toutes les marques d'IBM est disponible sur la page Web "Copyright and trademark information" à l'adresse http://www.ibm.com/legal/copytrade.shtml.

Adobe, le logo Adobe, PostScript et le logo PostScript sont des marques d'Adobe Systems Incorporated aux Etats-Unis et/ou dans certains autres pays.

Intel, le logo Intel, Intel Inside, le logo Intel Inside, Intel Centrino, le logo Intel Centrino, Celeron, Intel Xeon, Intel SpeedStep, Itanium et Pentium sont des marques d'Intel Corporation ou de ses filiales aux Etats-Unis et dans certains autres pays.

Microsoft, Windows, Windows NT et le logo Windows sont des marques de Microsoft Corporation aux Etats-Unis et/ou dans certains autres pays.

Java ainsi que tous les logos et toutes les marques incluant Java sont des marques d'Oracle et/ou ses sociétés affiliées.

Linux est une marque de Linus Torvalds aux Etats-Unis et/ou dans certains autres pays.

UNIX est une marque de The Open Group aux Etats-Unis et dans certains autres pays.

Index

Caractères spéciaux

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Numéro de programme : 5698-WSH

SC11-2009-10