IBM WebSphere Business Integration Adapters

IBM WebSphere Business Integration Adapter for Portal Infranet - Guide de l'utilisateur

Version 44.x

IBM WebSphere Business Integration Adapters

IBM WebSphere Business Integration Adapter for Portal Infranet - Guide de l'utilisateur

Version 44.x

Consigne

Avant d'utiliser le présent document et le produit associé, prenez connaissance des informations générales figurant dans la section «Informations légales», à la page 121.

Remarque

Les captures d'écrans et les graphiques de ce manuel ne sont pas disponibles en français à la date d'impression.

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
Tour Descartes
92066 Paris-La Défense Cedex 50

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.

A propos de ce document

La famille de produits IBM WebSphere Business Integration Adapter propose une connectivité d'intégration pour les technologies e-business de pointe, les applications d'entreprise, les systèmes propriétaires et les grands systèmes. Cette famille de produits fournit des outils et des modèles destinés à la personnalisation, la création et la gestion des composants pour l'intégration métier.

Contenu du document

Ce document décrit l'installation, la configuration des propriétés du connecteur, le développement d'objets métier et la résolution des incidents de cet adaptateur IBM WebSphere Business Integration.

Limites du document

Ce document ne décrit pas les indicateurs de déploiement et ne traite pas les questions relatives à la planification de la capacité, telles que l'équilibrage de la charge, le nombre d'unités d'exécution de l'adaptateur, les débits maximum et minimum et les seuils de tolérance.

Ces questions dépendent du déploiement de chaque utilisateur et doivent être étudiées dans les conditions les plus proches possibles de l'environnement exact dans lequel l'adaptateur doit être déployé. Contactez le service de maintenance IBM pour discuter de la configuration de votre site de déploiement, afin d'obtenir des informations sur la planification et l'évaluation de ces types d'indicateurs, en fonction de votre configuration.

Public visé

Ce document s'adresse aux consultants et utilisateurs de WebSphere qui mettent en oeuvre le connecteur dans le cadre d'un système d'intégration métier WebSphere. Pour utiliser les informations de ce document, vous devez avoir des connaissances dans les domaines suivants :

- Développement du connecteur
- Développement de l'objet métier
- Architecture d'applications Portal Infranet

Documents connexes

La documentation complète qui accompagne ce produit présente les caractéristiques et les fonctions communes à toutes les installations de composants WebSphere Business Integration Adapter, et inclut des supports de référence sur des composants spécifiques.

Vous pouvez télécharger la documentation associée sur les sites suivants :

- Pour obtenir des informations générales sur les adaptateurs, pour apprendre à les utiliser avec des courtiers de messages WebSphere (WebSphere MQ Integrator, WebSphere MQ Integrator Broker, WebSphere Business Integration Message Broker) et avec WebSphere Application Server, voir le centre de

documentation IBM WebSphere Business Integration Adapters à l'adresse : http://www.ibm.com/websphere/integration/wbiadapters/infocenter

- Pour utiliser les adaptateurs avec WebSphere InterChange Server, voir les centres de documentation IBM WebSphere InterChange Server : http://www.ibm.com/websphere/integration/wicserver/infocenter http://www.ibm.com/websphere/integration/wbicollaborations/infocenter
- Pour plus d'informations sur les courtiers de messages WebSphere : http://www.ibm.com/software/integration/mqfamily/library/manualsa/
- Pour plus d'informations sur WebSphere Application Server : http://www.ibm.com/software/webservers/appserv/library.html

Ces sites contiennent des explications simples pour télécharger, installer et afficher la documentation.

Conventions typographiques

Ce document utilise les conventions suivantes :

`Police courier`	Indique une valeur littérale, comme le nom d'une commande, le nom d'un fichier, des informations que vous tapez ou que le système affiche à l'écran.
gras	Indique un nouveau terme à sa première occurrence.
italique `italic`	Indique un nom de variable ou une référence croisée.
souligné en bleu	Le soulignement en bleu, visible uniquement lorsque vous consultez le document en ligne, indique un hyperlien de référence croisée. Si vous cliquez sur le terme souligné, vous êtes renvoyé à l'objet de la référence.
`{ }`	Dans une ligne de syntaxe, les accolades entourent un ensemble d'options parmi lesquelles vous ne devez sélectionner qu'une seule.
`[ ]`	Dans une ligne de syntaxe, les crochets entourent un paramètre facultatif.
`...`	Dans une ligne de syntaxe, les points de suspension indiquent une répétition du paramètre précédent. Par exemple, `option[,...]` signifie que vous pouvez entrer plusieurs options séparées par des virgules.
`< >`	Dans une convention de dénomination, les signes inférieur à et supérieur à entourent les différents éléments d'un nom afin de pouvoir les différencier les uns des autres, par exemple, `<nom_serveur><`*`nom_connecteur`*`>tmp.log`.
`/, \`	Dans ce document, les barres obliques inversées (\) sont utilisées pour indiquer les chemins de répertoires. Pour les systèmes UNIX, remplacez les barres obliques inverses par des barres obliques (/). Tous les noms de chemin des produits de systèmes d'intégration WebSphere sont relatifs au répertoire dans lequel le produit est installé sur votre système.
%text% et *$text*	Du texte placé entre des signes pourcentage (%) indique la valeur de la variable système `texte` Windows ou la variable utilisateur. Le symbole équivalent dans un environnement UNIX est *$texte*, indiquant la valeur de la variable d'environnement UNIX *texte*.
ProductDir	Correspond au répertoire où le produit est installé.

Nouveautés de la présente édition

Version 4.4.x

Mise à jour de septembre 2005. L'édition de ce document pour la version 4.4.x de l'adaptateur contient les nouveautés ou modifications suivantes.

Prise en charge de la gestion de l'adaptateur Portal Infranet par IBM Tivoli License Manager (ITLM).

Meilleure gestion des erreurs afin d'enregistrer davantage d'informations spécifiques à l'application sur les erreurs.

Prise en charge de IBM WebSphere Business Integration Adapter Framework V2.6.0.

Prise en charge des plateformes suivantes :

- Windows 2000 SP6
- Solaris 9
- HP-UX 11i v1

Prise en charge de Sun JDK 1.4.2.

Prise en charge de IBM JDK 1.4.2.

Prise en charge de Oracle 9i version 2 9.2.0.3 et supérieure.

Editions précédentes

Nouveautés des éditions précédentes :

Version 4.3.x

Les modifications apportées à la version 4.3.x de ce connecteur n'ont pas d'incidence sur le contenu de ce document.

Version 4.2.x

L'adaptateur pour Portal Infranet est désormais pris en charge sur HP-UX 11i.

Depuis la version 4.2x, l'adaptateur pour Portal Infranet n'est plus pris en charge sous Microsoft Windows NT.

Les informations relatives à l'installation de l'adaptateur ont été déplacées. Reportez-vous au chapitre 2 «Installation de l'adaptateur Portal Infranet et d'autres fichiers», à la page 13 pour connaître le nouvel emplacement de ces informations.

Version 4.1.x

La section «Exemple de définition complète d'objet métier Portal Infranet», à la page 47 a été ajouté pour servir de modèle au développement d'objets métier personnalisés.

L'adaptateur peut désormais utiliser WebSphere Application Server comme courtier d'intégration. Pour plus d'informations, voir «Environnement d'Adapter for Portal Infranet», à la page 11. L'adaptateur fonctionne désormais sur les plateformes suivantes :

- Solaris 7, 8
- AIX 5.x

Version 4.0.x

Mise à jour de mars 2003. Le nom "CrossWorlds" n'est plus utilisé pour décrire un système complet ou pour modifier les noms des composants ou outils, qui restent par ailleurs quasiment inchangés. Par exemple, "CrossWorlds System Manager" est devenu "System Manager," et "CrossWorlds InterChange Server" est maintenant "WebSphere InterChange Server".

Le connecteur a remplacé l'utilitaire CWSAPGEN par l'utilitaire PORTALODA. Pour plus d'informations, voir Chapitre 4, «Génération de définitions d'objets métier à l'aide de PortalODA», à la page 51.

Version 3.1.x

Le connecteur internationalisé est livré avec IBM WebSphere Business Integration Adapter for Portal Infranet.

Version 3.0.x

Cette édition du connecteur contient les nouvelles fonctions suivantes : Le connecteur a été internationalisé. Pour plus d'informations, voir «Traitement des données dépendantes des paramètres régionaux», à la page 8 et Annexe A, «Propriétés standard du connecteur», à la page 65. Le connecteur prend en charge les logiciels suivants sur AIX 4.3.3:

- Oracle 8.1.7 et Portal Infranet 6.2 SP1
- DB2 7.1.0 et Portal Infranet 6.2 SP1

Version 2.5.x

IBM WebSphere Business Integration Adapter for Portal Infranet inclut le connecteur pour Portal Infranet. Cet adaptateur fonctionne avec les courtiers d'intégration InterChange Server (ICS) et WebSphere MQ Integrator. Un courtier d'intégration est une application qui permet l'intégration d'ensembles hétérogènes d'applications. Il fournit divers services dont le routage des données. Cet adaptateur comprend :

- Un composant d'application spécifique à Portal Infranet
- Un exemple d'objet métier
- IBM WebSphere Adapter Framework, composé de :
  - Des outils de développement (y compris Business Object Designer et Connector Configurator)
  - Des API (y compris ODK, JCDK et CDK)

Le présent manuel fournit des informations sur l'utilisation de cet adaptateur avec les deux courtiers d'intégration : InterChange Server (ICS) et WebSphere MQ Integrator.

Remarque : Le connecteur prend désormais en charge Portal Infranet 6.2.0.

Important : Etant donné que le connecteur n'a pas été internationalisé, ne l'exécutez pas avec InterChange Server version 4.1.1 si vous n'êtes pas sûr de ne traiter que des données ISO Latin-1.

Version 2.4.x

Les changements apportés à la version 2.4.x de ce connecteur n'ont pas d'incidence sur le contenu de ce document.

Version 2.3.x

Des changements mineurs ont été apportés afin de corriger des incidents et permettre la compatibilité avec l'infrastructure IBM CrossWorlds version 4.0.0.

Version 2.2.x

La version du connecteur prend désormais en charge Portal Infranet 6.1.0.

Version 2.1.x

- La version du connecteur prend désormais en charge Portal Infranet 6.0.1.
- Le connecteur peut être installé et exécuté sous UNIX.
- Ce document a été en grande partie réorganisé et remanié.

Chapitre 1. Présentation du connecteur

Ce chapitre présente le composant du connecteur de IBM WebSphere Business Integration Adapter for Portal Infranet et inclut les sections suivantes :

Les connecteurs sont composés de deux parties : l'architecture du connecteur et le composant spécifique à l'application. L'architecture du connecteur, dont le code est commun à tous les connecteurs, joue le rôle d'intermédiaire entre le courtier d'intégration et le composant propre à l'application. Le composant propre à l'application contient des codes adaptés à une application ou une technologie spécifique. L'architecture de connecteur fournit les services suivants entre le courtier d'intégration et le composant spécifique à l'application :

- Elle reçoit et envoie des objets métier.
- Elle assure l'échange des messages de démarrage et d'administration.

Le présent document contient des informations sur l'architecture du connecteur et le composant spécifique à l'application, désigné comme étant le connecteur.

Le connecteur permet à WebSphere MQ Integrator Broker ou IBM WebSphere InterChange Server (ICS) de communiquer avec Portal Infranet via l'échange d'objets métier. L'application Infranet est constituée d'un ensemble de programmes développés par le logiciel Portal Infranet afin de gérer des comptes clients. Les informations relatives au client, telles que les numéros de comptes et les données de facturation, sont stockées dans une base de données Infranet.

Le connecteur et l'application Portal Infranet communiquent à l'aide de l'interface de programme d'application Infranet basée sur socket. Le connecteur génère des transactions à l'aide des fonctions offertes par l'interface de programme d'application Infranet. Cette application Infranet notifie ensuite le courtier d'intégration (WebSphere MQ Integrator Broker ou ICS) via le module d'événements lorsque des changements se produisent.

Composants du connecteur

Le connecteur pour Portal Infranet inclut les composants suivants :

- Connecteur : fichier Java `.jar` qui implémente le support de l'instruction de l'objet métier et le mécanisme d'interrogation d'événements.
- Module des fonctions d'événements de WebSphere Business Integration Adapter : DLL C++ sous Windows et fichiers SO sous UNIX, avec implémentation du mécanisme de notification des événements dans l'application Infranet. Il sélectionne les événements Infranet adéquats pour le courtier

d'intégration et les stocke dans une table de la base de données Infranet. Le connecteur interroge régulièrement cette table.

Le connecteur génère des objets métier qu'il transmet au courtier d'intégration. Le connecteur répond également aux requêtes d'objets métier émises par le courtier d'intégration. Il génère des messages de journaliation et de traçage qu'il écrit dans un fichier ou dans la console du connecteur ou envoie au courtier d'intégration.

La figure 1 illustre l'architecture du connecteur et son mécanisme d'événements dans l'application Infranet.

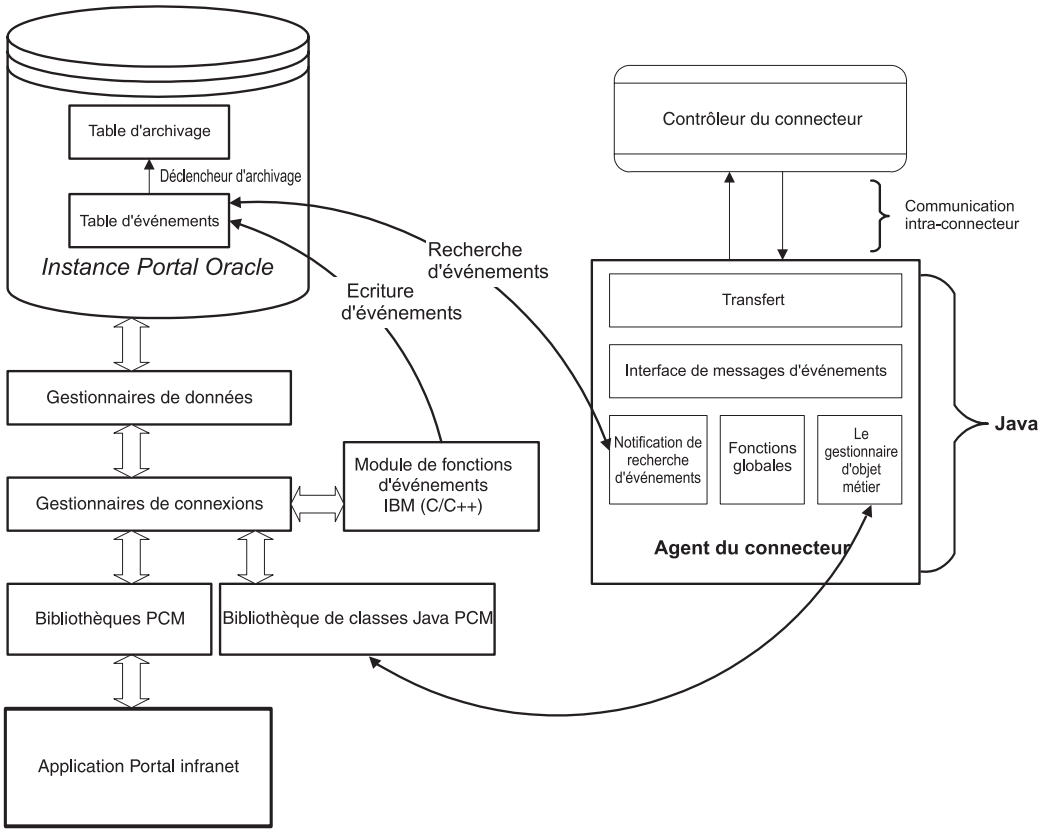


Figure 1. Architecture du connecteur

Le connecteur utilise la bibliothèque de classe Java PCM (Portal Communications Module) comme interface de programme d'application pour interagir avec les gestionnaires de connexion de Portal Infranet. L'avantage de cette architecture réside dans le fait que Portal Infranet prend en charge la bibliothèque de classe Java PCM sur les ordinateurs Java virtuels, à la fois sous Windows et sous UNIX. Ceci permet au connecteur de s'exécuter sur les deux plateformes. Cette application de programme d'application est utilisée par le gestionnaire d'objet métier du connecteur pour échanger des informations entre le courtier d'intégration et Portal. Le module des fonctions d'événements de WebSphere Business Integration Adapter utilise la bibliothèque PCM C++.

Fonctionnement du connecteur

Les sections suivantes décrivent la façon dont le connecteur traite les requêtes d'objets métier et gère les notifications d'événements.

Comportement des métadonnées du connecteur

Le connecteur est contrôlé par les métadonnées. Il est conçu pour gérer l'extraction et l'envoi de tous les objets métier, quel que soit leur type ou les variables qu'ils comportent. Pour que le connecteur soit contrôlé par les métadonnées, les objets métier destinés à Portal Infranet doivent contenir les informations suivantes :

- Le nom de zone de chaque attribut tel qu'identifié par le dictionnaire de données dans Infranet. Ceci inclut un numéro de zone de code confidentiel pour chaque nom de zone de code confidentiel, tel qu'indiqué dans la documentation de l'interface de programme d'application. Le nom de la zone est indiqué sous forme d'informations spécifiques à l'application au niveau de l'attribut, puis converti en nombre par le connecteur à l'aide du dictionnaire de données.
- Les codes opération pris en charge par cet objet métier. Ces codes opération sont indiqués au niveau de l'instruction de l'objet métier. Un code opération Infranet est une opération utilisée par les applications et les scripts clients afin de gérer les données client, créer des comptes en ligne, collecter et assurer le suivi des données client et intégrer les systèmes tiers dans Infranet.

Pour plus d'informations sur les métadonnées d'objets métier pour Portal, voir Chapitre 3, «Présentation des objets métier», à la page 27.

Traitement des objets métier

Lorsque le connecteur reçoit une requête d'objet métier du système WebSphere Business Integration, le gestionnaire d'objets métier du connecteur traite l'objet métier. Le gestionnaire d'objet métier assure la liaison entre l'objet spécifique à l'application et l'interface de programme d'application Portal Infranet. Il est chargé de transmettre une opération Portal Infranet à l'interface de programme d'application et de créer un objet métier spécifique à l'application transmis au système WebSphere Business Integration en tant que résultat d'un événement Infranet. Le gestionnaire d'objets métier utilise les données de l'objet métier et tout type de métadonnées pour appeler l'interface de programme d'application Java Infranet afin d'envoyer un objet stockable vers Portal. Une fois cette opération terminée, un statut est renvoyé vers le courtier d'intégration.

Dans la figure 2, l'organigramme de traitement de l'information illustre à un niveau élevé la façon dont le gestionnaire d'objets métier traite les requêtes d'objets métier. Le gestionnaire d'objets métier extrait l'instruction et les attributs clé depuis l'objet métier. Il utilise l'instruction pour déterminer l'appel de fonction effectué pour gérer l'objet métier. Dans cet exemple, s'il s'agissait d'une instruction Update, la fonction UpdateObject serait appelée.

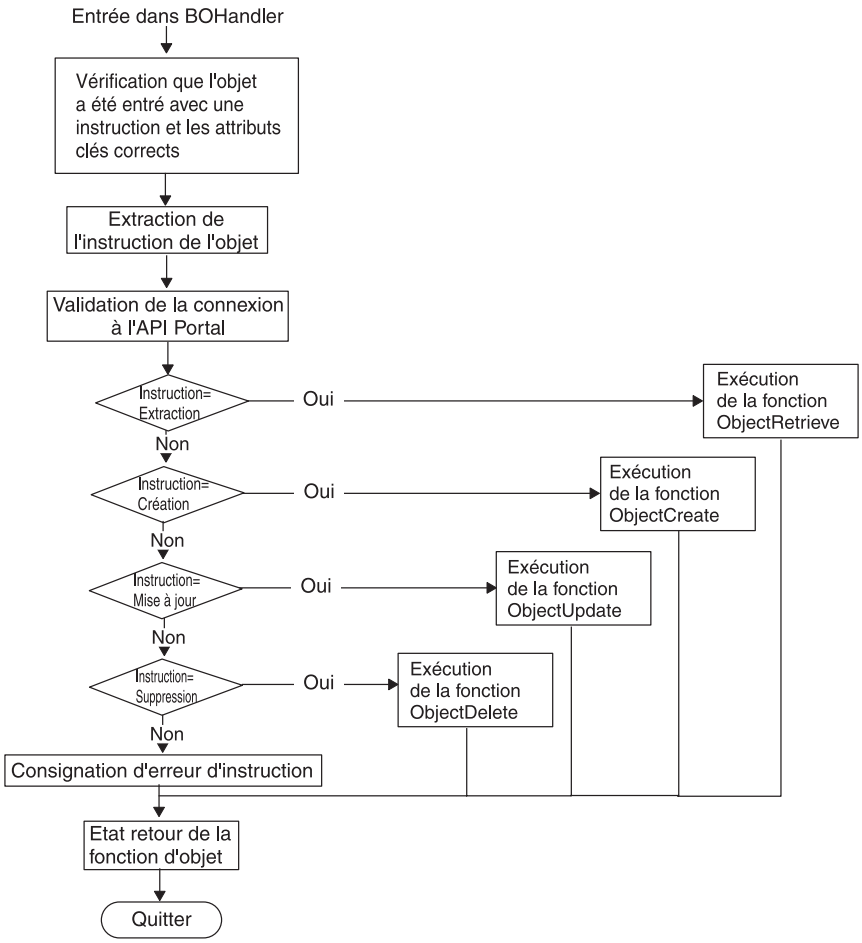


Figure 2. Vue d'ensemble du traitement des objets métier

Traitement de l'instruction Retrieve

La méthode Retrieve du gestionnaire d'objets métier extrait un objet de Portal Infranet et alimente un objet métier de WebSphere Business Integration Adapter avec les informations de l'application. La méthode Retrieve du connecteur effectue les opérations suivantes :

1. Elle vérifie si la connexion de Portal Infranet est valide. Si tel n'est pas le cas, la connexion doit être rétablie. Si la connexion est rompue en cours de traitement, le connecteur renvoie un statut `BON_FAIL` indiquant un problème de connexion avec Infranet.
2. Elle extrait les informations spécifiques à l'application pour l'objet. Les informations spécifiques à l'application indiquées pour une instruction fournissent le code opération qui doit être appelé pour extraire un objet Portal Infranet.
3. Elle prépare la flist destinée au code opération basé sur les informations spécifiques à l'application pour l'objet métier et ses attributs.

   Remarque : Une `flist` est une liste de longueur variable qui contient des paires de zones et de valeurs.Les `flists` fournissent des paramètres d'entrée et de sortie aux codes opération et aux fonctions Infranet.

4. Elle appelle le code opération spécifique avec la `flist` en entrée et une `flist` vide en sortie.
5. Si les étapes précédentes se déroulent normalement, la structure de la `flist` renvoyée contient un objet `flist` intégralement rempli qui correspond à l'objet stockable défini par un objet métier WebSphere Business Integration Adapter. Etant donné que la `flist` a une correspondance un à un avec l'objet métier WebSphere Business Integration Adapter, elle est parcourue de façon à extraire tous les attributs de l'objet métier WebSphere Business Integration Adapter basés sur les informations spécifiques à l'application au niveau de l'attribut qui identifie le nom et le type de zone de cette `flist`.
6. Elle alimente l'objet métier et le transmet au courtier d'intégration.

La figure 3 illustre le fonctionnement de la méthode Retrieve. La méthode détermine l'action à entreprendre sur un attribut, en fonction de son type. S'il s'agit d'un attribut de type basique (tel qu'une chaîne), le gestionnaire d'objets métier alimente la zone de façon dynamique. Si la méthode détecte un objet métier enfant, elle analyse l'objet en profondeur jusqu'à atteindre les attributs de base de cet objet métier enfant. Ensuite, elle passe en revue tous les attributs de base de l'objet enfant avant de procéder de même avec ceux de l'objet parent.

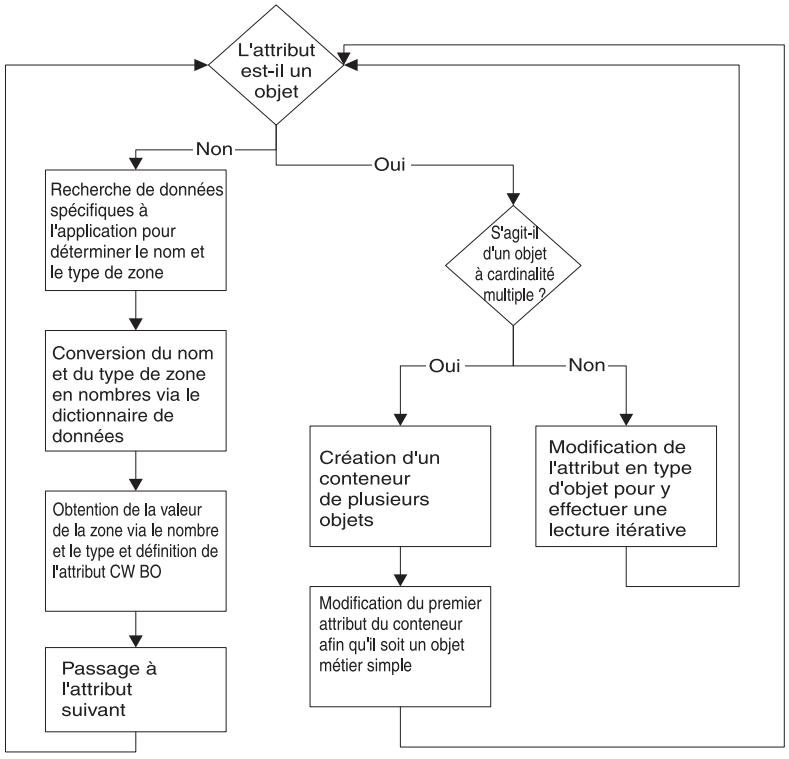


Figure 3. Organigramme pour le traitement de l'instruction Retrieve

Traitement des instructions Create et Update

Pour traiter une instruction Create ou Update, le gestionnaire d'objets métier construit un objet d'interface de programme d'application Infranet (une flist) et le transmet à l'interface de programme d'application. Le gestionnaire d'objets métier suit les étapes suivantes :

1. Elle extrait l'objet métier spécifique à l'application de l'objet métier WebSphere Business Integration Adapter.

2. Elle alimente l'objet métier de l'interface de programme d'application Portal Infranet. Lorsque la flist est instanciée, le connecteur effectue une itération dans l'objet, attribut par attribut, puis identifie les valeurs avec lesquelles alimenter cette flist. Ce processus doit effectuer une recherche dans un objet pour détecter un attribut susceptible de se trouver à l'intérieur des différentes couches qui composent cet objet.
3. Elle vérifie que la connexion Portal Infranet est toujours valide. Si la connexion est interrompue lors du traitement, le connecteur renvoie un statut `BON_FAIL`, lequel signale un problème de connexion avec Infranet, et la connexion doit être réinstanciée.
4. Elle utilise les informations spécifiques à l'application pour l'instruction de l'objet métier afin d'appeler le code opération approprié de façon dynamique pour l'opération Create ou Update. Une fois que les paramètres sont rassemblés et placés dans un tableau et que la chaîne fonctionnelle a été assemblée, la fonction de contexte dotée du code opération adéquat est appelée.
5. Pour traiter des objets métier enfants qui disposent de leur propre code opération, le gestionnaire d'objets métier alimente une flist séparée pour chaque enfant, puis appelle le code opération approprié.
6. Elle renvoie un statut lorsque l'opération est terminée.

Notification d'événement

Infranet dispose d'un mécanisme de notification d'événements qui permet d'assurer le suivi des actions qui se déroulent dans l'application. Lorsqu'un utilisateur réalise une action dans Infranet, l'application génère un événement associé.

Le module d'événements WebSphere Business Integration Adapter examine l'événement et détermine s'il est susceptible d'intéresser le connecteur. Si tel est le cas, le module d'événements génère une entrée dans la table d'événements WebSphere Business Integration Adapter correspondant à l'événement.

Détection d'événements dans Infranet

La détection d'événements dans Infranet est implémentée à l'aide d'un module de fonctions Infranet personnalisées appelé par le mécanisme de notification des événements. Ce module de fonctions est fourni par le courtier d'intégration et fonctionne avec deux fichiers de configuration pour identifier les événements Infranet et écrire des événements dans la table d'événements WebSphere Business Integration Adapter.

Lorsqu'un objet Infranet est modifié, un événement permanent est mis en avant. Infranet peut être configuré de façon à appeler un code opération spécifique lorsqu'un événement survient ; ainsi, le courtier d'intégration fournit un fichier de configuration plat qui configure Infranet de façon qu'il appelle le module de fonctions des événements WebSphere Business Integration Adapter pour les événements associés aux objets métier pour Portal. Ce fichier de configuration est appelé "pin_notify_cw" et est chargé dans Infranet à l'aide de l'utilitaire `load_pin_notify` fourni avec Infranet.

Lorsque le module d'événements reçoit un événement, il extrait les informations de l'objet lié à ce dernier et crée une entrée dans le tableau d'événements WebSphere Business Integration Adapter. Dans Infranet, un événement est en fait une instance de classe stockable Infranet et chaque événement de création, modification, ou suppression se rapporte à une classe stockable Infranet spécifique. Par exemple, si un utilisateur modifie un contact particulier en relation avec un client, Infranet génère une instance de la classe stockable `/event/customer/nameinfo`.

Le module d'événements utilise son propre fichier de configuration pour déterminer l'événement qui est survenu, identifier la partie de la classe stockable ayant été modifiée (et l'objet métier qui s'y rapporte), puis déterminer le type d'action qui s'est produit. A l'aide du fichier de configuration `event_code.txt`, le module d'événements examine l'événement Infranet et alimente la table d'événements WebSphere Business Integration Adapter avec un enregistrement qui reflète cet événement.

Le mécanisme de notification d'événements du connecteur utilise trois tables créées dans une instance de base de données Oracle par Infranet.

- XWORLDS_Events – Stocke tous les événements en attente
- XWORLDS_Archive_Events – Archive les événements que le connecteur a traités
- XWORLDS_Current_Event_ID – Stocke l'ID du dernier événement

Le schéma de la première table indique les informations enregistrées pour chaque événement envoyé par Infranet qui intéresse le connecteur. Sa présentation est aussi utilisée pour la table d'archivage.

La détection des événements et le traitement associé s'effectuent dans une transaction Infranet. Infranet appelle les processus personnalisés dans une transaction et attend les résultats du traitement. Si le processus personnalisé renvoie une erreur, la transaction est abandonnée. Ceci permet de garantir qu'aucun événement ne soit perdu par le connecteur.

Remarque :

Problèmes connus - Le module de notification d'événements vérifie l'ID utilisateur de tous les événements Portal envoyés définis dans le fichier pin_notify_cw. Si aucun PIN_FLD_USERID n'est associé à l'événement envoyé au module, une erreur sera générée et des problèmes se produiront lors de la sauvegarde de l'objet en ligne. Ces types d'événements doivent être ajustés à l'aide de FLists ou de classes stockables afin d'inclure un tel ID. Recherchez ces erreurs dans le fichier journal défini dans le fichier de configuration crossworlds.cnf.

Le module d'événements recherche un ID utilisateur pour empêcher que les événements envoyés dans l'application par le connecteur soient ajoutés à la file d'attente d'événements. Cette procédure est appelée ″travail en alternance″.

″/event/customer/billinfo″ est un type d'événement dans lequel un tel problème existe.

Extraction des événements

Le connecteur recherche des événements potentiels en interrogeant la table XWORLDS_Events qui a été définie dans l'instance de base de données Infranet. Le connecteur effectue l'interrogation en utilisant une instruction SQL SELECT pour extraire les entrées de la table XWORLDS_Events. Le nombre d'événements sélectionnés est indiqué par la propriété PollQuantity du connecteur.

L'interrogation s'effectue dans le connecteur à l'aide de la méthode `pollForEvents()`.Le connecteur interroge la table d'événements en fonction de la propriété PollFrequency définie dans les propriétés du connecteur WebSphere Business Integration Adapter. Si une nouvelle ligne est détectée dans la table, les données de l'événement sont extraites, puis le connecteur traite l'événement de la façon suivante :

1. La fonction d'interrogation crée un objet métier vide, définit l'instruction sur Retrieve, puis configure les clés à l'aide de l'enregistrement de l'événement. L'objet métier est envoyé au gestionnaire d'objets métier du connecteur'.
2. Le gestionnaire d'objets métier utilise les données de l'événement pour appeler l'interface de programme d'application Java Infranet pour extraire l'objet stockable Portal Infranet.
3. Le gestionnaire d'objets métier convertit l'objet stockable en objet métier WebSphere Business Integration Adapter spécifique à une application, définit l'instruction sur l'action indiquée par l'enregistrement de l'événement, puis transmet l'objet métier au courtier d'intégration.

Une fois que l'objet métier est transmis au système WebSphere Business Integration, la table d'événements est archivée dans la table XWORLDS_Archive_Events et supprimée dans la table d'événements.

L'intervalle de temps selon lequel la méthode d'interrogation est appelée peut être ajusté en modifiant la propriété PollFrequency du connecteur. Cette propriété est définie en utilisant le courtier d'intégration.

Connexion à l'application Infranet

Lors de la connexion au gestionnaire de connexions Portal Infranet à l'aide de l'interface de programme d'application, le connecteur effectue les opérations suivantes :

1. Il crée une instance de contexte Portal Infranet.
2. Lorsqu'un code opération doit être exécuté, le connecteur utilise un contexte du groupe disponible et l'ajoute au groupe utilisé.
3. Lorsque la tâche est terminée, le contexte est renvoyé vers le groupe disponible.

L'instruction Connect utilise les valeurs des propriétés de connecteur spécifiques à l'application qui sont définies dans le référentiel.

Les instances du contexte situées dans le groupe sont fermées lorsque le connecteur est clos.

Traitement des données dépendantes des paramètres régionaux

Le connecteur a été internationalisé, il peut prendre en charge les jeux de caractères à deux octets et transmettre le texte du message dans la langue indiquée. Lorsque le connecteur transfère des données depuis un emplacement qui utilise un jeu de codes de caractères spécifique vers un emplacement qui utilise un jeu de codes de caractères différent, il procède à la conversion des caractères afin de conserver le sens des informations. L'environnement d'exécution Java de la machine virtuelle Java (JVM) représente les données dans le jeu de codes de caractères Unicode. Le format Unicode contient des codes pour les caractères présents dans la plupart des jeux de codes de caractères connus (à la fois mono-octet et multi-octets). La plupart des composants du système IBM CrossWorlds sont écrits en Java. Par conséquent, lorsque des données sont transférées entre la plupart des composants IBM

CrossWorlds, la conversion des caractères est inutile. Pour enregistrer les messages d'erreur et d'informations dans la langue et le pays ou territoire approprié, configurez la propriété standard de configuration de paramètres régionaux pour votre environnement. Pour plus d'informations sur ces propriétés, voir l'Annexe A, «Propriétés standard du connecteur», à la page 65.

Chapitre 2. Installation et configuration du connecteur

Le présent chapitre explique comment installer et configurer le connecteur. Il contient les rubriques suivantes :

Le composant connecteur d'IBM WebSphere Business Integration Adapter for Portal Infranet a deux composants qui doivent être installés et configurés :

- Connecteur. Le connecteur est un fichier Java `.jar`. Il implémente le support d'instruction de connecteur et le mécanisme d'interrogation d'événements.
- Module des fonctions d'événements de WebSphere Business Integration Adapter. Ce module est un exécutable qui implémente la notification d'événements. Il sélectionne les événements Infranet adéquats pour le courtier d'intégration et les stocke dans une table de base de données.

Ce chapitre explique comment installer et configurer les composants du connecteur et comment configurer l'application Portal Infranet pour l'utiliser.

Remarque : Dans ce document, les barres obliques inversées (\) sont utilisées pour les chemins de répertoires, sauf dans certains fichiers de code exemple. Pour les systèmes UNIX, remplacez les barres obliques (/) par des barres obliques inverses (\). Tous les noms de chemins de fichiers sont relatifs au répertoire dans lequel le produit WebSphere Business Integration Adapter est installé sur votre système, sauf mention contraire.

Environnement d'Adapter for Portal Infranet

Pour connaître le matériel et les logiciels requis pour cet adaptateur, voir la section IBM WebSphere Adapters and IBM WebSphere Business Integration Adapters: Hardware and Software Requirements. Sélectionnez votre adaptateur dans liste des adaptateurs WebSphere.

Configuration de l'application Infranet

Afin de configurer l'application Infranet pour être utilisée par le connecteur, vous devez définir un compte utilisateur pour le connecteur et créer les tables d'événements et d'archivage dans la base de données Oracle utilisée par Infranet.

Configuration d'un compte Infranet

A l'aide d'Infranet Administrator, définissez un utilisateur CSR (Customer Service Representative) disposant de tous les droits. Cet utilisateur est utilisé par le connecteur et identifie ce dernier. Cet ID utilisateur est défini dans le fichier de configuration du module d'événement `crossworlds.cnf` et dans les paramètres de configuration du connecteur. Le module des fonctions d'événement personnalisées vérifie cette valeur avant d'insérer des événements afin d'éviter que les événements envoyés dans l'application par le connecteur ne soient redistribués à ce dernier. Ce scénario est aussi appelé "travail en alternance"

Création des tables d'événements et d'archivage dans la base de données

Les tables d'événements et d'archivage sont utilisées pour mettre en file d'attente des événements qui doivent être récupérés par le connecteur. Le mécanisme de notification d'événements pour le connecteur requiert que trois tables d'événements soient créées dans l'instance de base de données Oracle utilisée par Infranet. Il s'agit des tables suivantes :

- XWORLDS_Events – Table d'événements utilisée pour stocker tous les événements Infranet en attente pertinents pour le connecteur.
- XWORLDS_Archive_Events – Table d'archivage dans laquelle des événements sont écrits une qu'ils ont été traités par le connecteur.
- XWORLDS_Current_Event_ID – Table utilisée pour stocker le numéro d'ID d'événement le plus récent.

  Remarque : Cette table *doit être* initialisée à 0.

Les deux premières tables indiquent les informations qui seront enregistrées pour chaque événement Infranet pertinents pour le connecteur. La table d'archivage contient tous les événements qui ont été traités par le connecteur.

Pour créer des tables d'événements et d'archivage, chargez le fichier `EventTable.sql`si vous utilisez une base de données Oracle. Si vous utilisez une base de données DB2, chargez le fichier`EventTable2.sql` dans *`%RépProduit%`*`\connectors\Portal\dependencies\config_files`.

Description du schéma des tables d'événements et d'archivage

La table d'événements contient les colonnes suivantes. Sa présentation est aussi utilisée pour la table d'archivage.

Tableau 1. Schéma des tables d'événements et d'archivage

Nom	Type	Description
Event_id	entier	Clé unique pour l'événement. La valeur clé est générée dans la table XWORLDS_Current_Event_ID.
Object_name	char 80	Nom de l'objet métier spécifique à l'application
Object_verb	char 80	Instruction associée à l'événement.
Object_key	VARCHAR	Clé primaire pour l'objet (POID).
Event_time	Date heure	Heure à laquelle s'est produit l'événement
Archive_time	Date heure	Table d'archivage seulement. Heure à laquelle l'événement a été reçu par Portal Infranet.
Event_status	Integer	Etat de l'événement : READY_FOR_POLL 0 SENT_TO_INTERCHANGE 1 UNSUBSCRIBED_EVENT 2 IN_PROGRESS 3 ERROR_PROCESSING_EVENT -1 ERROR_SENDING_EVENT_TO_INTERCHANGE -2
Event_comment	char 255	Chaîne utilisée pour fournir des informations supplémentaires sur l'événement. L'utilisateur peut définir ce commentaire dans le fichier de configuration du module d'événements.
Event_priority	Integer	Priorité associée à l'événement. Plus le nombre est faible, plus la priorité est élevée. L'utilisateur peut définir cette priorité dans le fichier de configuration du module d'événements.

Installation de l'adaptateur Portal Infranet et d'autres fichiers

Pour plus d'informations sur l'installation des produits de WebSphere Business Integration, voir le guide *Installing WebSphere Business Integration Adapters* dans l'Infocenter de WebSphere Business Integration Adapters, sur le site Web suivant :

http://www.ibm.com/websphere/integration/wbiadapters/infocenter

Configuration de l'adaptateur dans un environnement Oracle

Si vous utilisez Oracle comme base de données, suivez ces instructions pour configurer le connecteur :

1. Connectez-vous au système UNIX sous le compte `pin`. Le cas échéant, créez ce compte.
2. Copiez les fichiers Infranet `pcm.jar` et `pcmext.jar`dans `$RépProduit/Connector/Portal/dependencies`. Ce fichier se trouve dans le répertoire`infranet/jars` sur le serveur Infranet 6.7.0.
3. Copiez le fichier `.profile` dans le répertoire principal de l'utilisateur `pin`, par exemple `/home/pin`. Le cas échéant, modifiez le fichier `.profile` pour que les variables d'environnement définies dans votre système soient prises en compte. Effectuez des modifications via un éditeur de texte tel que`vi` .

   Lorsque les variables d'environnement sont correctes, chargez-les dans le système en tapant la commande suivante à l'invite :

   ```
   source .profile
   ```
4. Placez le fichier `fm_crossworlds.so` dans le répertoire `$INFRANET/lib`. . Ce fichier contient les déclencheurs des événements.

   Notez qu'UNIX distingue les majuscules des minuscules, donc, si des fichiers sont introuvables, vérifiez que vous avez entré les noms de répertoires et de fichiers en respectant la casse.
5. Vérifiez que la variable `$LIBRARY_PATH` contient le chemin `$INFRANET/lib` afin que le système puisse reconnaître les fichiers `.so` du connecteur.
6. Copiez les fichiers ci-après dans le répertoire`$CW_PORTAL_PATH` .

- crossworlds.cnf. Ce fichier contient les informations de configuration pour le module d'événements.

  Le cas échéant, modifiez ce fichier pour votre système. Un exemple de contenu pour ce fichier est présenté ci-après :

  ```
  db name = oracle1
  db string = NYNON
  db user = pin
  db password = pin
  crossworlds id = 0.0.0.1\service\admin_client 14088
  log level  = 3
  log file = D:\pinlog.log
  ```

  où :

db name	Pour Oracle, db_name est la variable hôte.
db string	Nom de la base de données.
db user	Nom de l'utilisateur se connectant à la base de données Portal Infranet.
db password	Mot de passe.
crossworlds id	POID représentant l'utilisateur WebSphere Business Integration Adapter dans Portal Infranet.
log level	Nombre représentant le niveau de journalisation : 0 : Pas de trace 1 : Erreur uniquement 2 : Erreur et avertissement 3 : Erreur, avertissement et débogage (toutes les traces)
log file	Nom du fichier journal.

  Remarque : Indiquez des valeurs pour db name et db string si vous utilisez une base de données Oracle. Si vous disposez d'une base de données par défaut sur votre machine locale, il est inutile d'indiquer des valeurs.

- event_code.txt. Ce fichier contient des descriptions d'événements Infranet qui seront utilisées par le module d'événements pour générer des entrées dans la table d'événements WebSphere Business Integration Adapter.

7. Placez pin_notify_cw dans le répertoire $INFRANET/sys/test. Ce fichier contient les noms des événements de connecteur. Si des événements doivent être ajoutés ou supprimés, respectez le format standard du fichier. Notez que /event encapsule toutes les sous-classes, telles que /event/customer, /event/status.
8. Arrêtez et redémarrez l'application Infranet. Vérifiez que$INFRANET/bin est dans la variable $PATH. Suivez ces étapes :
   a. Arrêtez Infranet avec cette commande :

      ```
      stop_all
      ```

   b. Vérifiez que tous le processus Infranet sont arrêtés en tapant la commande ci-après. Notez que les numéros de processus (PID) des processus Infranet actifs.

      ```
      ps -ef|grep portal
      ```

   c. Arrêtez les processus Infranet actif via cette commande :

      ```
      kill -9 <PID>
      ```

d. Redémarrez Infranet via cette commande :

```
start_all
```

9. Dans le répertoire $CM, éditez le fichier pin.conf en ajoutant la ligne suivante à la sectionfm_required. Tapez le chemin de répertoires complet pour$INFRANET .

```
- cm fm_module $INFRANET/lib/fm_crossworlds.so fm_cw_pol_config -pin
```

10. Vérifiez que Infranet est actif en entrant la commandeps -ef|grep portal.
11. Accédez au répertoire $INFRANET/sys/test, ouvrez le fichier pin.conf et vérifiez qu'il contient une ligne similaire à la suivante.

```
- nap cm ptr ip machine_cm_Infranet port_cm
```

Par exemple :

```
- nap cm ptr ip roadrunner 11960
```

roadrunner correspond à *machine_cm_Infranet* et *port_cm* est 11960.

Outre la ligne ci-dessus, le fichierpin.conf doit contenir ces lignes :

```
- nap login_type 1
- nap login_name root.0.0.0.1
- nap login_pw password
```

Il s'agit des informations de connexion à Infranet. Si aucun fichier pin.conf ne se trouve dans le répertoire, copiez-en un.

12. Pour charger des informations de configuration dans l'application Infranet, entrez la commande :

```
load_pin_notify pin_notify_cw
```

La réponse doit être successful. Si une autre réponse s'affiche, consultez le fichierpin_notify_cw. Il contient en effet les codes opération qui seront appelés par Infranet lorsque des événements spécifiques se produisent. Notez que pin_notify_cw doit se trouver dans le même répertoire que l'exécutableload_pin_notify.

13. Dans le répertoire $INFRANET_VAR/cm, consultez le fichier journal et vérifiez que core est mentionné dans $CM, démarrez Infranet Administrator.
14. Pour tester le connecteur, entrez ou modifiez un compte et consultez la table d'événements xworlds_events pour connaître l'entrée d'événement adéquate. Cela aboutissant à un résultat fictif, l'entrée d'événement doit être supprimée à l'issue du test.

Pour lancer le connecteur, voir «Démarrage du connecteur», à la page 24.

Configuration du connecteur

Les adaptateurs ont deux types de propriétés de configuration : standard et spécifiques à l'adaptateur. Vous devez définir les valeurs de certaines de ces propriétés avant d'exécuter le connecteur.

Les propriétés du connecteur sont configurées depuis Connector Configurator (lorsque WebSphere MQ Integrator Broker est le courtier d'intégration ou depuis Connector Configurator, auquel vous accédez depuis System Manager (lorsque ICS est le courtier d'intégration). Pour obtenir des informations de configuration détaillées, voir Annexe B, «Utilisation de Connector Configurator», à la page 91 et *Connector Development Guide for Java*.

Propriétés standard du connecteur

Les propriétés standard de configuration fournissent des informations utilisées par tous les connecteurs. Voir l'Annexe A, «Propriétés standard du connecteur», à la page 65 pour plus d'informations sur ces propriétés.

Important : Ce connecteur prenant en charge tous les courtiers d'intégration, les propriétés de configuration de tous les courtiers lui sont applicables.

Remarque : Ce connecteur étant doté d'une seule unité d'exécution, il ne peut pas tirer parti de la propriété AgentConnections.

Le tableau 2 donne des informations spécifiques à ce connecteur et concernant les propriétés de configuration de l'annexe.

Tableau 2. Informations sur les propriétés spécifiques à ce connecteur

Propriété	Important
CharacterEncoding	Ce connecteur n'utilise pas cette propriété.
Locale	Ce connecteur étant internationalisé, vous pouvez modifier la valeur de cette propriété. Voir les notes d'informations concernant le connecteur pour connaître les paramètres régionaux pris en charge.

Important : WebSphere MQ Integrator Broker ne prend pas en charge plusieurs paramètres nationaux. Vérifiez que chaque composant de votre installation (par exemple, tous les adaptateurs, toutes les applications et le courtier d'intégration) sont définis en fonction des mêmes paramètres nationaux.

Remarque : L'adaptateur Portal Infranet prend en charge la propriété `DuplicateEventElimination`. Pour activer la propriété `DuplicateEventElimination`, attribuez la valeur true à l'attribut`DuplicateEventElimination`. Pour plus de détails sur la propriété`DuplicateEventElimination`, voir Annexe A, «Propriétés standard du connecteur», à la page 65.

Remarque : Ce connecteur prenant en charge tous les courtiers d'intégration, les propriétés de configuration de tous les courtiers lui sont applicables.

Propriétés spécifiques au connecteur

Les propriétés de configuration spécifiques au connecteur fournissent des informations requises par le connecteur au moment de l'exécution. Les propriétés spécifiques au connecteur permettent également de modifier les informations statiques ou logiques dans le connecteur sans devoir le recoder et le recompiler.

Le tableau 3 contient les propriétés de configuration spécifiques au connecteur. Pour obtenir une explication des propriétés, voir les sections suivantes.

Tableau 3. Propriétés de configuration spécifiques au connecteur

Nom	Valeurs possibles	Valeur par défaut	Obligatoire
Mot de passe de l'application	*Mot de passe pour le compte utilisateur*		Oui
ApplicationUserName	*Nom du compte utilisateur*		Oui
CommentFail	*Commentaire relatif à la table d'événements pour les événements ayant échoué*	Echec	Non
CommentSucceed	*Commentaire relatif à la table d'événements pour les événements ayant abouti*	Succès	Non
DatabaseInfo	`jdbc:oracle:thin@CWENGTEST:1521:` portaldb	Nom d'hôte serveur, numéro de port de la base de données Portal et nom de celle-ci.	Oui
DbName	*Nom du programme d'écoute Oracle*		Oui
DbPassword	*Mot de passe de la base de données*		Oui
DbUser	*Nom d'utilisateur de la base de données*	`pin`	Oui
DriverClass	Nom de classe de pilote de la base de données		Non
InfDatabase	*String*	`0.0.0.1`	Oui
InfHost	*Nom et port de la machine hôte*	`//CWENGTEST1:11960`	Oui
InfLogFile	*Nom du fichier journal*	`InfConnection.txt`	Oui
InfranetConnections	*Nombre de connexions Infranet ouvertes par le connecteur pour le regroupement de connexions*	5	Oui
InfService	*String*	`service\admin_client 1`	Oui
InfType	`0` ou `1`	`1`	Oui
InfVersion	*Version de l'application Infranet*		Oui
PollQuantity	*Nombre d'événements à récupérer*	`1`	Non
UseDefaults	`true` ou `false`	`false`	Non

Mot de passe de l'application

Mot de passe défini pour le compte utilisateur du système WebSphere Business Integration.

ApplicationUserName
Nom du compte utilisateur pour le connecteur.

CommentFail
Commentaire sur la table d'événements en cas d'échec. La valeur par défaut est `Fail`.

CommentSucceed
Commentaire sur la table d'événements en cas de succès. La valeur par défaut est `Succeed`.

DatabaseInfo
Cette propriété définit l'URL qui serait utilisée par le pilote JDBC pour établir une connexion avec la base de données.

Exemple :

```
jdbc:oracle:thin:@CWENGTEST1:1521:portaldb
```

Consultez la documentation JDBC pour connaître les formats spécifiques de l'URL

DbName
Nom du programme d'écoute Oracle pour la base de données Infranet.

DbPassword
Mot de passe de la base de données.

DriverClass
Nom de classe de pilote de la base de données.

Pour Oracle, définissez :

```
oracle.jdbc.driver.OracleDriver
```

Si DriverClass n'est pas défini, l'adaptateur utilise `oracle.jdbc.driver.OracleDriver` comme valeur par défaut.

DbUser
Nom d'utilisateur de la base de données, généralement`pin`.

InfDatabase
Chaîne au format Infranet. La valeur par défaut est `0.0.0.1`..

InfHost
Nom et port de la machine hôte, par exemple, `//engtest2:11960`. La valeur par défaut est `//CWENGTEST1:11960`.

InfLogFile
Nom du fichier à utiliser en tant que journal par défaut. La valeur par défaut est `InfConnection.txt`.

InfranetConnections
Cette propriété est utilisée pour définir le nombre de connexions ouvertes par le connecteur avec l'application l'application Infranet pour le regroupement de connexions. Le connecteur gère le nombre affecté de connexions. Lorsqu'un processus d'objet métier a besoin d'une connexion, il en alloue une du groupe. Cette connexion est alors retirée du groupe disponible et ajoutée au groupe utilisé. A l'issue du processus d'objet métier, la connexion est retirée du groupe utilisé et

replacée dans le groupe disponible. Ce mode d'utilisation du regroupement de connexions augmente les performances car il n'est pas nécessaire d'ouvrir et de fermer la connexion pour chaque processus métier.

Remarque : Si cette propriété n'est pas définie, le connecteur va générer l'exception suivante au démarrage du connecteur : `NumberFormatException`

InfService
Chaîne, généralement `service\admin_client 1`. Il s'agit de la valeur par défaut.

InfType
Type de connexion. Les seules valeurs possibles sont 0 ou 1. La valeur par défaut est 1.

InfVersion
Version de l'application Infranet.

PollQuantity
Nombre d'événements récupérés en une seule interrogation. La valeur par défaut est 1.

UseDefaults
Si la propriété UseDefaults a la valeur true ou n'est pas définie, le connecteur vérifie si une valeur correcte ou une valeur par défaut est indiquée pour chaque attribut d'objet métier isRequired. Si une valeur est indiquée, l'opération de création aboutit. Dans le cas contraire, l'opération échoue.

Si le paramètre est défini sur false, le connecteur cherche uniquement des valeurs valides et entraîne l'échec de l'opération Create s'il n'en trouve pas.

La valeur par défaut est ″false″.

Personnalisation du mécanisme d'événement pour les nouveaux objets métier

Si vous créez un objet métier, vous devez déterminer quels sont les événements qui seront générés pour chaque action sur cet objet. Lorsque cela est déterminé, vous devez personnaliser le fichier de configuration du module d'événements afin que la DLL de ce dernier puisse trouver des événements de ce type. Le nom du fichier de configuration du module d'événements est`event_code.txt` ; il se trouve dans `$INFRANET$\sys\cm`.

Infranet génère suffisamment de données relativement à un événement pour permettre au module d'événements d'identifier quelle partie d'une classe stockable (en d'autres termes, quel objet métier) a été appelée. Lorsque le module d'événements extrait un événement, il extrait une instance d'une classe stockable contenant des informations, telles que le compte, l'utilisateur et le programme appelant.

Pour différencier une mise à jour d'une suppression ou d'une création, le module d'événements compare la valeur d'origine et celle mise à jour. Pour savoir si une création ou une suppression a eu lieu, cependant, le module d'événements doit extraire la classe stockable lorsque l'action est effectuée au niveau de la racine ou il

peut consulter l'ID d'élément d'un objet enfant. Lorsqu'un enfant est ajouté, son ID d'élément est positif et contient la position de cet élément dans le tableau. S'il est négatif, il a été supprimé.

Syntaxe du fichier de configuration du module d'événements

Lors de modifications du fichier de configuration du module d'événements, celles-ci doivent être conformes aux règles syntaxiques ci-après. Cette syntaxe doit être respectée à la lettre.

1. Les lignes de commentaire commencent par deux tirets.
2. Un événement est décrit sur une ligne. Les paramètres sont délimités par une barre verticale (|).
3. La syntaxe d'une ligne est la suivante :

   ```
   <événement>|<action Inf.>|<tableau>|< poid clé>|<contraintes>|<instruction OM.>|<priorité>
   |<commentaire>
   ```

   où :

événement	Nom de classe stockable de l'événement
Action Infranet	C (Création), M (Mise à jour) ou S (Suppression). Cela représente l'action effectuée dans Portal Infranet.
array	Code Portal Infranet représentant le tableau sur lequel l'action est effectuée. Le tableau est l'élément Infranet qui doit être extrait des informations d'événement. Dans Portal Infranet, chaque zone a un numéro associé. Par exemple, PIN_FLD_NAMEINFO est associé au code Infranet 156. La liste des zones et des numéros associés se trouve dans le fichier`$INFRANET$\Include\pin_flds.h`.
poid	Code Portal Infranet représentant la zone qui est la clé de la classe stockable créée, mise à jour ou supprimée.
contraintes	Liste des contraintes nécessaires pour déterminer exactement ce qui s'est passé. La contrainte prend en charge ces mots clés : • `exists` ou `not_exists`: true ou false ; spécifie si l'objet existe ou non. • =, >, >=, <=, < : opérateurs de comparaison pour les types numériques • `equal`, `nequal`, `contains` : opérateurs de comparaison pour le type de chaîne. • `&` est utilisé pour séparer les contraintes. Les contraintes sont vraies si toutes les conditions séparées par `&` le sont. Si vous voulez indiquer un `or`, utilisez plusieurs lignes. La première ligne dans laquelle les contraintes ont la valeur true sont exécutées.
instruction OM.	Nom de l'objet métier et instruction correspondant à l'action dans Portal Infranet.
priorité	Priorité de l'événement
commentaire	Commentaire à insérer dans la table d'événements

Exemple de fichier de configuration du module d'événements

Le texte ci-dessous montre un exemple de fichier `event_code.txt`. Cet exemple inclut des lignes indiquant des événements pour la classe stockable `account`. Les lignes commençant par des tirets sont des commentaires.

```
-- Account creation: PIN_FLD_STATUSES[0].PIN_FLD_STATUS[0] = 0 &
      PIN_FLD_SYS_DESCR = "Set Status (acct)": OK
/event/customer/status |U |144  |40  |40 exists&55;5 equal "Set Status
      (acct)"&144:0-145;3 = 0 |Portal_Account.Create |1 |Account Creation
```

```
-- Account Updated (status updated) : PIN_FLD_STATUSES[1].PIN_FLD_STATUS[0] =
      10100 or 10103 or 10102 & & PIN_FLD_SYS_DESCR = "Set Status (acct)" : OK
/event/customer/status |U |144  |40  |40 exists&55;5 equal "Set Status (acct)
      "&144:1-145;3 > 0 |Portal_Account.Update |1 |Status Updated

-- Account Updated (new contact added):OK
/event/customer/nameinfo |C |156 |40|40 exists|Portal_Account.Update|1|new contact

-- Account Updated (contact updated): OK
/event/customer/nameinfo |U |156 |40 |40 exists&17;5 equal "Customer Mngmt. Event
      Log" |Portal_Account.Update |2 |contact_update

-- Account Updated (contact deleted):OK
/event/customer/nameinfo |D |156  |40  |40 exists&67;5 nequal "Automatic Account
      Creation"|Portal_Account.Update |2 |contact_delete

-- Account Updated (billinfo updated) : two PIN_FLD_BILLINFO and
      PIN_FLD_BILLINFO[0].PIN_FLD_BILL_TYPE[0] <> 0 : OK
/event/customer/billinfo |U |126  |40 |40 exists & 126:0-127;3 > 0
      Portal_Account.Update |2 |billinfo_update
```

Définition des entrées du fichier de configuration d'événements

Lorsqu'un événement se produit, Infranet génère une flist représentant l'événement. Le module de fonctions d'événements examine la flist pour identifier l'instruction et l'action qui s'est produite.

Par exemple, supposons qu'une flist représentant un événement soit :

```
0     PIN_FLD_POID            POID [0] 0.0.0.1 /event/customer/nameinfo -1 0
0     PIN_FLD_NAME             STR [0] "Customer Mngmt. Event Log"
0     PIN_FLD_USERID          POID [0] 0.0.0.1 /service/admin_client 2 1
0     PIN_FLD_SESSION_OBJ     POID [0] 0.0.0.1 /event/session 10366 0
0     PIN_FLD_ACCOUNT_OBJ     POID [0] 0.0.0.1 /account 9406 32
0     PIN_FLD_PROGRAM_NAME     STR [0] "Admin Manager"
0     PIN_FLD_START_T       TSTAMP [0] (942104217) 11/08/99 15:36:57
0     PIN_FLD_END_T         TSTAMP [0] (942104217) 11/08/99 15:36:57
0     PIN_FLD_SYS_DESCR        STR [0] "Set Name Info"
0     PIN_FLD_NAMEINFO       ARRAY [0] allocated 20, used 17
1     PIN_FLD_SALUTATION       STR [0] ""
1     PIN_FLD_LAST_NAME        STR [0] "Event Test1"
1     PIN_FLD_LAST_CANON       STR [0] "event test1"
1     PIN_FLD_FIRST_NAME       STR [0] "Event Test1"
1     PIN_FLD_FIRST_CANON      STR [0] "event test1"
1     PIN_FLD_MIDDLE_NAME      STR [0] ""
1     PIN_FLD_MIDDLE_CANON     STR [0] ""
1     PIN_FLD_TITLE            STR [0] "Event Test1  "
1     PIN_FLD_COMPANY          STR [0] "Event Test1"
1     PIN_FLD_ADDRESS          STR [0] "Event Test1"
1     PIN_FLD_CITY             STR [0] "Event Test1"
1     PIN_FLD_STATE            STR [0] "CA"
1     PIN_FLD_ZIP              STR [0] "00000"
1     PIN_FLD_COUNTRY          STR [0] "US"
1     PIN_FLD_EMAIL_ADDR       STR [0] ""
1     PIN_FLD_CONTACT_TYPE     STR [0] "Billing"
1     PIN_FLD_ELEMENT_ID      UINT [0] 1
0     PIN_FLD_NAMEINFO       ARRAY [1] allocated 20, used 19
1     PIN_FLD_SALUTATION       STR [0] ""
1     PIN_FLD_LAST_NAME        STR [0] "Event Test1"
1     PIN_FLD_LAST_CANON       STR [0] "event test1"
1     PIN_FLD_FIRST_NAME       STR [0] "Event Test1"
1     PIN_FLD_FIRST_CANON      STR [0] "event test1"
1     PIN_FLD_MIDDLE_NAME      STR [0] ""
1     PIN_FLD_MIDDLE_CANON     STR [0] NULL str ptr
1     PIN_FLD_TITLE            STR [0] "Event Test1"
```

```
1     PIN_FLD_COMPANY          STR [0] "Event Test1"
1     PIN_FLD_ADDRESS          STR [0] "Event Test1"
1     PIN_FLD_CITY             STR [0] "Event Test1"
1     PIN_FLD_STATE            STR [0] "CA"
1     PIN_FLD_ZIP              STR [0] "00000"
1     PIN_FLD_COUNTRY          STR [0] "US"
1     PIN_FLD_EMAIL_ADDR       STR [0] ""
1     PIN_FLD_CONTACT_TYPE     STR [0] "Billing"
1     PIN_FLD_ELEMENT_ID      UINT [0] 1
1     PIN_FLD_CANON_COUNTRY    STR [0] "US"
1     PIN_FLD_CANON_COMPANY    STR [0] "event test1"
0     PIN_FLD_ITEM_OBJ        POID [0] 0.0.0.1 /item 11454 0
0     PIN_FLD_CURRENCY        UINT [0] 840
```

Lors de l'ajout d'une ligne au fichier du mode d'événements, vous devez indiquer le numéro associé à la zone au lieu du nom de celle-ci. Par exemple, la zone PIN_FLD_NAMEINFO, figurant en gras dans la flist précédente, est représentée par le code 156.

Une contrainte doit être utilisée pour identifier une zone située au delà du premier niveau de la flist. A l'exception d'une contrainte, chaque zone est identifiée par un numéro simple, qui est celui qui lui est associé.

Un événement donné pouvant être généré pour plusieurs raisons, des contraintes sont parfois requises pour déterminer si un événement est le résultat d'un objet métier particulier. Par conséquent, vous pouvez avoir besoin d'ajouter des contraintes telles que `exists`.

La syntaxe suivante doit être utilisée dans une contrainte pour identifier une zone.

```
[<code de tableau:élément de tableau>-zone;type]
```

Par exemple, si le code Infranet 156 représente PIN_FLD_NAMEINFO, et le code 161 représente PIN_FLD_LAST_NAME, la représentation de la zone PIN_FLD_LAST_NAME figurant dans le tableau PIN_FLD_NAMEINFO de l'élément ID 1 est représentée dans une contrainte associée à cette zone avec 156:1-161;5 = PINFLDNAMEINFO[1].PIN_FLD_LASTNAME

Le dernier numéro suivant le point-virgule dans une contrainte représente le type de l'objet. Il s'agit du type de données sur lequel la comparaison doit être effectuée. Par exemple, le numéro 5 indique une chaîne. Pour plus d'informations, voir la liste des types associés dans `$INFRANET$\Include\pin_type.h`.

Ajout d'événements au fichier pin_notify_cw

Lorsque vous ajoutez de nouveaux objets métier, vous devez aussi ajouter des événements au fichier`pin_notify_cw`. Ce fichier contient les noms des événements de connecteur. Pour ajouter des événements, respectez le format standard du fichier, tel qu'il est présenté dans «Définition des entrées du fichier de configuration d'événements», à la page 21.

Le module de notification d'événements CrossWorlds (FmCw.dll) vérifie l'ID utilisateur de tout événement Portal qui lui est envoyé (défini dans le fichier pin_notify_cw). Si aucun PIN_FLD_USERID n'est associé à l'événement envoyé au module, une erreur sera générée et des problèmes se produiront lors de la sauvegarde de l'objet en ligne. Ces types d'événements doivent être ajustés (f-lists ou les classes stockables) pour inclure un tel ID. Recherchez ces erreurs dans le fichier journal défini dans le fichier de configuration crossworlds.cnf.

Le module d'événements recherche un ID utilisateur pour empêcher que les événements envoyés dans l'application par le connecteur soient ajoutés à la file d'attente d'événements. Cette procédure est appelée ″travail en alternance″. ″/event/customer/billinfo″ est un exemple de type d'événement dans lequel un tel problème existe.

Remarque : Afin d'identifier tous les événements générés pour une opération Infranet, appelez le module d'événements pour tous les événements Infranet (en utilisant `/event` dans le fichier de configuration `load_pin_notify`). Le module d'événements sera alors déclenché pour tous les événements générés. Indiquez le niveau de journalisation 3 dans le fichier de configuration du module d'événements, puis, dans le fichier `pinlog.log`, la flist d'événements sera envoyée par Infranet pour cette opération.

Déclaration des configurations facultatives des attributs de personnalisation Infranet

Si l'application Infranet a été personnalisée, les classes Java correspondantes doivent être générées. Un outil est fourni par Infranet pour générer de telles classes. Le connecteur doit connaître ces classes au moment de l'exécution, elles doivent donc être déclarées dans le `CLASSPATH` .

1. Créez un répertoire, par exemple : `c:\CustomAttributes`.
2. Créez un fichier `#include` avec les zones personnalisées et leurs valeurs.
3. Lancez le script `custom_fields.pl` sur ce fichier en indiquant `com.portal.pcm.fields` comme nom de package.
4. Compilez les fichiers java pour créer les fichiers de classe :

   ```
   javac –classpath c:\program files\infranet\java\pcmext.jar *.java
   ```

5. Créez une hiérarchie de répertoires sous votre répertoire en cours`com\portal\pcm\fields`.
6. Copiez le `.class` généré par le compilateur dans ce répertoire.
7. Ajoutez le répertoire en cours (par exemple : `c:\CustomAttributes`) au `CLASSPATH` du connecteur.
8. Le cas échéant, modifiez le fichier`\bin\start_Portal.bat`.

Création de plusieurs instances de connecteur

La création de plusieurs instances d'un connecteur revient pratiquement à créer un connecteur personnalisé. Vous pouvez configurer votre système de sorte qu'il crée et exécute plusieurs instances d'un connecteur en suivant les étapes ci-dessous. Vous devez :

- créer un répertoire pour l'instance du connecteur ;
- vérifier que vous possédez les définitions d'objet métier requises ;
- créer un fichier de définition pour le connecteur ;
- créer un script de démarrage.

Création d'un répertoire

Vous devez créer un répertoire pour chaque instance de connecteur. Vous devez attribuer le nom suivant à ce répertoire de connecteur :

```
ProductDir\connectors\connectorInstance
```

où `connectorInstance` identifie de manière unique l'instance de connecteur.

Si le connecteur possède des méta-objets qui lui sont spécifiques, vous devez créer un méta-objet pour l'instance de connecteur. Si vous enregistrez le méta-objet en tant que fichier, créez le répertoire suivant et stockez le fichier dedans :

`ProductDir\repository\connectorInstance`

Création de définitions d'objet métier

Si les définitions d'objet métier pour chaque instance du connecteur n'existent pas déjà dans le projet, vous devez les créer.

1. Si vous devez modifier les définitions d'objet métier associées au connecteur initial, copiez les fichiers appropriés et utilisez Business Object Designer pour les importer. Vous pouvez copier n'importe quel fichier pour le connecteur initial. Vous devez simplement les renommer si vous les modifiez.
2. Les fichiers pour le connecteur initial doivent résider dans le répertoire suivant :

   `ProductDir\repository\initialConnectorInstance`

   Tous les fichiers supplémentaires que vous créez doivent être placés dans le sous-répertoire `connectorInstance` approprié de `ProductDir\repository`.

Création d'une définition de connecteur

Vous devez créer un fichier de configuration (définition du connecteur) pour l'instance du connecteur dans Connector Configurator. Pour ce faire, procédez comme suit :

1. Copiez le fichier de configuration du connecteur initial (définition du connecteur) et renommez-le.
2. Assurez-vous que chaque instance du connecteur répertorie correctement ses objets métier pris en charge (et tous les méta-objets associés).
3. Personnalisez toutes les propriétés du connecteur le cas échéant.

Création d'un script de démarrage

Pour créer un script de démarrage, procédez comme suit :

1. Copiez le script de démarrage du connecteur initial et attribuez-lui un nom incluant le nom du répertoire du connecteur :

   dirname
2. Placez ce script de démarrage dans le répertoire du connecteur créé à la section «Création d'un répertoire», à la page 23.
3. Créez un raccourci pour le script de démarrage (Windows uniquement).
4. Copiez le texte du raccourci du connecteur et modifiez le nom du connecteur initial (dans la ligne de commande) de sorte qu'il corresponde au nom de la nouvelle instance du connecteur.

A présent, vous pouvez exécuter simultanément les deux instances du connecteur sur votre serveur d'intégration.

Pour plus d'informations sur la création de connecteurs personnalisés, voir *Connector Development Guide for C++ or for Java*.

Démarrage du connecteur

Vous devez démarrer un connecteur de manière explicite à l'aide du **script de démarrage du connecteur**. Sous Windows, le script de démarrage doit résider dans le répertoire d'exécution du connecteur :

ProductDir`\connectors\`*connName*

où *connName* identifie le connecteur.

Sous UNIX, le script de démarrage doit résider dans le répertoire *UNIX ProductDir*/bin.

Le nom du script de démarrage dépend de la plateforme du système d'exploitation, comme le montre le tableau 4.

Tableau 4. Script de démarrage pour un connecteur

Système d'exploitation	Script de démarrage
Systèmes UNIX	`connector_manager`
Windows	`start_connName.bat`

Lorsque le script de démarrage s'exécute, il va chercher par défaut le fichier de configuration dans le *Productdir* (voir commandes ci-dessous). Il s'agit du répertoire dans lequel vous placez le fichier de configuration.

Remarque : Si l'adaptateur utilise le transfert JMS, vous avez besoin d'un fichier de configuration local .

Pour appeler le script de démarrage du connecteur, utilisez l'une des méthodes suivantes :

- Sur les systèmes Windows, dans le menu **Démarrer** :

  Sélectionnez **Programmes>IBM WebSphere Business Integration Adapters>Adapters>Connectors**. Par défaut, le nom du programme est "IBM WebSphere Business Integration Adapters". Cependant, vous pouvez le personnaliser. Vous pouvez également créer sur le bureau un raccourci vers le connecteur.
- A partir de la ligne de commande :
  - Sur les systèmes Windows :

    `start_connName connName brokerName [-cconfigFile]`
  - Sur les systèmes UNIX :

    `connector_manager -start connName brokerName [-cconfigFile ]`

    où *connName* est le nom du connecteur et *brokerName* le nom de votre connecteur d'intégration, comme suit :
    - Pour WebSphere InterChange Server, indiquez à la place de *brokerName* le nom de l'instance ICS.
    - Pour les courtiers de messages WebSphere (WebSphere MQ Integrator, WebSphere MQ Integrator Broker ou WebSphere Business Integration Message Broker) ou WebSphere Application Server, remplacez *brokerName* par une chaîne identifiant le courtier.

    Remarque : Pour un courtier de messages WebSphere ou WebSphere Application Server résidant sur un système Windows, vous *devez* inclure l'option `-c` suivie du nom du fichier de configuration du connecteur. Pour ICS, l'option `-c` est facultative.
- A partir de Adapter Monitor, qui est lancé au démarrage de System Manager, lequel est exécuté avec le courtier WebSphere Application Server ou InterChange Server :

  Vous pouvez charger, activer, désactiver, interrompre, arrêter ou supprimer un connecteur à l'aide de cet outil.

- A partir de System Manager (pour tous les courtiers) :

  Vous pouvez charger, activer, désactiver, interrompre, arrêter ou supprimer un connecteur à l'aide de cet outil.
- Sur les systèmes Windows, vous pouvez configurer le connecteur de sorte qu'il démarre comme un service Windows. Dans ce cas, le connecteur démarre lors de l'amorçage du système Windows (pour un service automatique) ou lorsque vous démarrez le service via la fenêtre Services Windows (pour un service manuel).

Pour plus d'informations sur le démarrage d'un connecteur, notamment sur les options de lancement à partir de la ligne de commande, reportez-vous à l'un des documents suivants :

- Pour WebSphere InterChange Server, voir *System Administration Guide*.
- Pour les courtiers de messages WebSphere, voir *Implementing Adapters with WebSphere Message Brokers*.
- Pour WebSphere Application Server, voir *Implementing Adapters with WebSphere Application Server*.

Arrêt du connecteur

La méthode pour arrêter un connecteur dépend de la manière dont il a été démarré, comme suit :

- Si vous avez démarré le connecteur à partir d'une ligne de commande, avec son script de démarrage :
  - Sur les systèmes Windows, l'appel du script de démarrage crée une fenêtre de "console" séparée pour le connecteur. Dans cette fenêtre, tapez "Q" et appuyez sur Entrée pour arrêter le connecteur.
  - Avec InterChange Server sur les systèmes UNIX, les connecteurs s'exécutent en arrière-plan de sorte qu'ils n'ont pas de fenêtre séparée. Vous devez exécuter la commande suivante pour arrêter le connecteur :

    `connector_manager_`*`connName`* `-stop`

    où *connName* correspond au nom du connecteur.
- A partir d'Adapter Monitor (produit WebSphere Business Integration Adapters uniquement), lancé lorsque vous démarrez System Manager :

  Vous pouvez charger, activer, désactiver, interrompre, arrêter ou supprimer un connecteur à l'aide de cet outil.
- A partir de System Monitor (produit WebSphere InterChange Server uniquement) :

  Vous pouvez charger, activer, désactiver, interrompre, arrêter ou supprimer un connecteur à l'aide de cet outil.
- Sur les systèmes Windows, vous pouvez configurer le connecteur de sorte qu'il démarre comme un service Windows. Dans ce cas, le connecteur s'arrête en même temps que le système Windows.

Chapitre 3. Présentation des objets métier

Ce chapitre décrit la manière dont le connecteur traite les objets métier et fournit des instructions concernant la mise en oeuvre de ces objets pour l'application Portal. Il traite des sujets suivants :

- «Eléments de base de l'application Portal Infranet»
- «Structure des objets métier spécifique à l'application Portal Infranet», à la page 31
- «Instructions relatives à la définition des objets métier», à la page 34
- «Informations d'objets métier spécifiques à l'application», à la page 34

Remarque : Si vous n'êtes pas familiarisé avec l'application Infranet, la conception et la mise en oeuvre des objets métier pour Portal Infranet peuvent s'avérer difficile. Dans ce cas, il est recommandé de travailler en collaboration avec un spécialiste de l'application. Pour de plus amples informations sur les éléments de programmation ou les concepts Infranet, consultez la documentation Infranet.

Eléments de base de l'application Portal Infranet

Cette section fournit une courte description de certains éléments de base de l'application Portal Infranet qui influent sur la conception et la mise en oeuvre des objets métier spécifiques à cette application. Infranet définit quatre éléments de programmation principaux utilisés pour déterminer et étendre les fonctionnalités du système, ou encore pour y accéder. Pour concevoir des objets métier pour l'application Portal, vous devez être familiarisé avec les éléments ci-dessous. Ils sont brièvement décrits dans les sections suivantes.

- Classes stockables
- Objets stockables
- Zones et listes de zones (flists)
- Codes opération

Classes et objets stockables

Dans Infranet, les classes stockables contiennent des zones comportant des informations sur une classe. Parmi les classes stockables standard, citons les classes de compte, de service, de factures, et d'autres classes prédéfinies par Infranet. Pour étendre les fonctionnalités d'Infranet, vous pouvez créer des classes stockables ou des sous-ensembles de classes existantes.

Les classes stockables ne contiennent pas de données réelles ; elles correspondent à des spécifications d'objets, tout comme une définition d'objet métier WebSphere Business Integration Adapter définit une structure d'objet métier, sans contenir de données. Elles incluent diverses zones, telles que des zones simples (par exemple, une zone d'entier ou de chaîne), des tableaux ou des sous-structures.

Lorsqu'une classe stockable a été instanciée et inclut des valeurs de données réelles, elle devient un objet stockable. Chaque objet stockable est identifié par un ID d'objet de portail unique, ou POID. Le POID contient le numéro de la base de données, le nom de la classe stockable, le numéro d'instance de l'objet stockable ainsi que son numéro de révision.

La différence entre une classe stockable et un objet stockable est illustrée dans la figure 4.

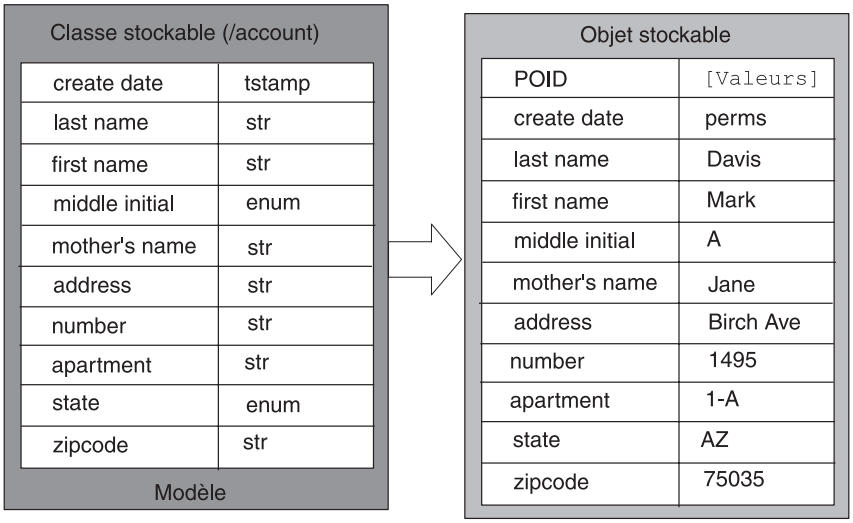


Figure 4. Classe et objet stockables Infranet

Une classe stockable peut définir des fonctionnalités héritées et étendues pour la classe. Par exemple, la classe stockable `/account/email` contient toutes les données de la classe `account` avec des informations complémentaires s'appliquant spécifiquement à la classe étendue `email`. Par conséquent, la classe stockable `/account/email`devient une sous-classe de `/account` comme indiqué dans la figure 5.

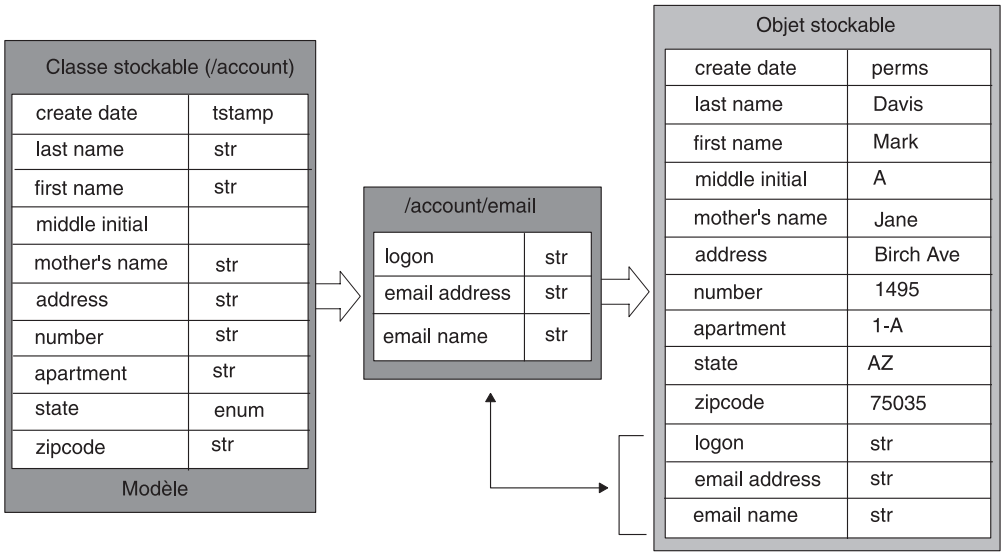


Figure 5. Extension de la classe stockable /account

Les objets stockables sont manipulés au moyen des scripts, outils et programmes d'application Infranet, ou de tout programme ou processus personnalisé. Quel que soit leur type, tous les programmes client exploitent des objets stockables au moyen des bibliothèques de programmation et de l'interface PCM. Les objets stockables sont manipulés par des codes opération, qui correspondent à des routines contenant des listes de zones exploitant ce type d'objet.

Zones et flists

Les zones correspondent à la valeur de donnée la plus simple d'Infranet. Chaque nom de zone du système est associé à un ID, un type et une définition uniques. Les noms de zone sont partagés et utilisés par de nombreuses classes et définitions de codes opération.

Le système intègre un ensemble de base de types de zones pouvant être utilisé pour créer des zones. Le tableau 5 répertorie les types de zones. Les six premiers types correspondent à des types de données de langages de programmation tels que C. Les autres contiennent des données plus complexes et peuvent pointer sur des structures en C utilisées comme valeurs. Les tableaux et les sous-structures contiennent des pointeurs vers d'autres listes de zones.

Tableau 5. Types de données et de zones Infranet

Type de zone	Type de donnée
PIN_FLDT_INT	Entier signé
PIN_FLDT_UINT	Entier sans signe
PIN_FLDT_ENUM	Entier énuméré
PIN_FLDT_NUM	Nombre en virgule flottante
PIN_FLDT_TSTAMP	Horodatage
PIN_FLDT_STR	Chaîne de caractères
PIN_FLDT_BINSTR	Chaîne binaire
PIN_FLDT_BUF	Mémoire tampon de données de taille arbitraire
PIN_FLDT_POID	ID d'objet de portail
PIN_FLDT_ARRAY	Tableau
PIN_FLDT_SUBSTRUCT	Sous-structure

Les listes de zones (flists) sont des structures de données fondamentales utilisées dans les interfaces API de programmation Infranet. Elles correspondent à des conteneurs renfermant des paires de zones de données et de valeurs, et dans certains cas, d'autres flists. Les flists peuvent représenter des calculs en virgule flottante, des mémoires tampons ou des éléments de données volumineux ne pouvant tenir en mémoire. Elles servent à la transmission d'informations entre les objets stockables et les routines ou programmes qui manipulent ces objets.

Un objet stockable (par exemple, dans une classe stockable `/account`) forme une flist (ou une partie d'une flist) qui utilise la spécification de classe stockable. La flist répertorie des zones, chacune étant associée à ses propres attributs, autorisations et valeurs de données. Ces zones définissent conjointement les fonctionnalités de l'objet stockable, comme indiqué dans la figure 6.

Les flists peuvent contenir plusieurs objets stockables. La structure des flists garantit la transmission des informations depuis l'application vers l'objet stockable approprié.

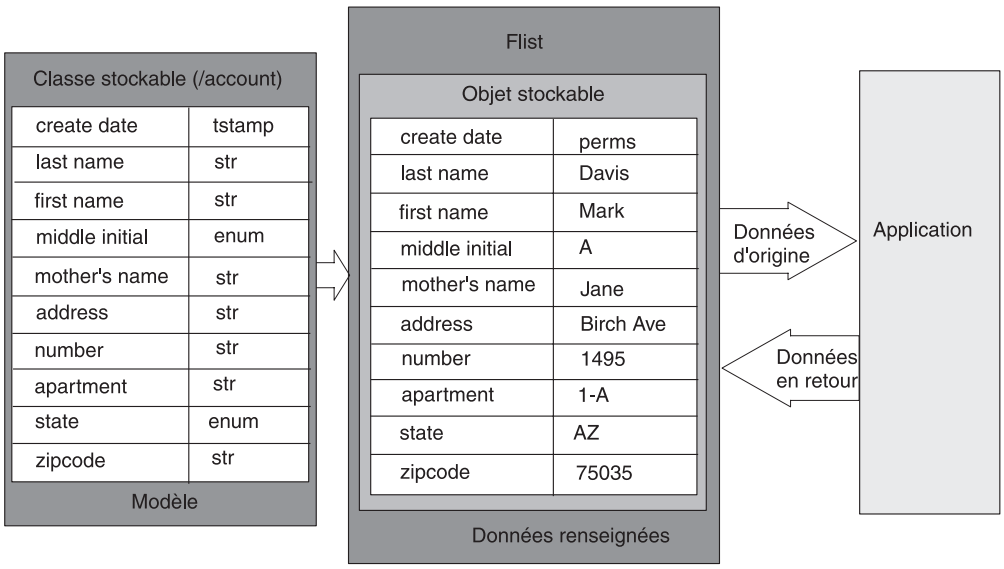


Figure 6. Objet stockable et flist

Codes opération

Une application utilise les codes opération système Infranet pour exécuter des opérations sur les objets stockables et les zones qu'ils contiennent. Il existe plusieurs ensembles de codes opération, qui sont regroupés dans les catégories fonctionnelles suivantes : Base, Module de fonctions Client (FM), Module de fonctions Activité, Module de fonctions Facturation, Module de fonctions Terminal et Module de fonctions Messagerie électronique.

Les opérations de base sur les objets incluent la création, la suppression, l'écriture, la lecture et la recherche. Tous les autres codes opération mettent en oeuvre une sémantique métier (de niveau supérieur) consistant notamment à consigner des activités, facturer l'achat d'un produit sur un compte, vérifier les informations relatives à une carte de crédit, modifier un nom et une adresse, vérifier un mot de passe ou enregistrer des données comptables. Ces codes opération de niveau supérieur sont mis en oeuvre dans des modules de fonctions, où les codes opération de base sont implémentés directement par le gestionnaire de stockage (SM). Les codes opération de niveau supérieur sont convertis en divers codes opération de base par les routines des modules de fonctions au sein des gestionnaires de communications, puis transmis aux gestionnaires de stockage.

Chaque code opération système est associé à une flist d'entrée et de sortie. Une application client détermine ce qu'est un événement "intéressant", appelle le système Infranet avec le code opération approprié et la flist correspondante, puis traite la flist renvoyée et la mémoire tampon des erreurs.

Connecteur contrôlé par les métadonnées

Le connecteur est contrôlé par les métadonnées. Cela signifie que les métadonnées de l'objet métier déterminent le mode de fonctionnement du connecteur. Les métadonnées sont des données sur l'application qui sont stockées dans un objet métier et aident le connecteur à interagir avec une application. Un connecteur contrôlé par les métadonnées traite chaque objet métier qu'il prend en charge

d'après les métadonnées codées dans la définition de l'objet métier plutôt que d'après les instructions codées en dur dans le connecteur.

Le connecteur est contrôlé par les métadonnées au moyen de la valeur Infranet PIN_ *FIELDNAME*au niveau de l'attribut et de la valeur du code opération au niveau de l'instruction. Dans la mesure où le connecteur est contrôlé par les métadonnées, il peut traiter de nouveaux objets métier ou des objets modifiés sans qu'il soit nécessaire de modifier son code. Toutefois, le connecteur fait des hypothèses concernant les aspects suivants de ses objets métier :

- Structure de ses objets métier
- Relations entre les objets métier parents et enfants
- Format des informations spécifiques à l'application
- Représentation d'un objet métier par la base de données

Par conséquent, lorsque vous créez ou modifiez un objet métier pour Portal Infranet, vos modifications doivent être conformes aux règles que le connecteur est supposé suivre, sinon il ne pourra pas traiter correctement les objets métier nouveaux ou modifiés. Les sections suivantes fournissent des informations sur la mise en oeuvre des objets métier pour l'application Portal.

Structure des objets métier spécifique à l'application Portal Infranet

Les objets métier WebSphere Business Integration Adapter sont hiérarchiques : les objets métier parents peuvent contenir des objets métier enfants, susceptibles d'inclure eux-mêmes d'autres objets métier enfants, et ainsi de suite. Le terme de conteneur à cardinalité 1 est utilisé lorsqu'un attribut d'un objet métier parent ne fait référence qu'à un seul objet enfant. La cardinalité n indique qu'un attribut contenu dans un objet métier parent fait référence à un tableau d'objets métier enfants.

Le connecteur prend en charge les relations de cardinalité 1 et de cardinalité n entre les objets métier.

Correspondance entre les objets Portal Infranet et les objets métier WebSphere Business Integration Adapter

Infranet utilise les types de conteneurs suivants :

- Classe stockable
- Tableau
- Sous-structure

Vous devez définir un objet métier spécifique à l'application WebSphere Business Integration Adapter Portal Infranet pour qu'il soit mappé avec la flist Infranet de l'objet stockable correspondant, avec l'ensemble des attributs et des relations nécessaires. La relation entre une flist et un objet métier est univoque.

Pendant le traitement, le connecteur compare un objet métier à la flist correspondante de l'objet Infranet et envoie une exception si les structures ne correspondent pas. Il est possible de définir un objet métier WebSphere Business Integration Adapter représentant un sous-ensemble de la structure flist, mais le processus inverse n'est pas pris en charge.

Pour chaque type de conteneur Infranet, un objet métier spécifique à l'application est créé selon les besoins. En général, une classe stockable devient un objet métier

de niveau supérieur. Un conteneur de type `sous-structure` peut devenir un objet métier enfant de cardinalité 1 et un conteneur de type `tableau` peut devenir un objet métier enfant de cardinalité n. Toutefois, si un sous-conteneur n'est pas important et que le code opération parent est suffisant pour manipuler les objets enfants, un objet métier enfant n'est pas requis.

La figure 7 montre le type de correspondance pouvant exister entre la structure d'un objet métier WebSphere Business Integration Adapter et celle d'une flist Infranet. Pour obtenir des informations sur les flists Infranet, reportez-vous à la documentation Portal Infranet.

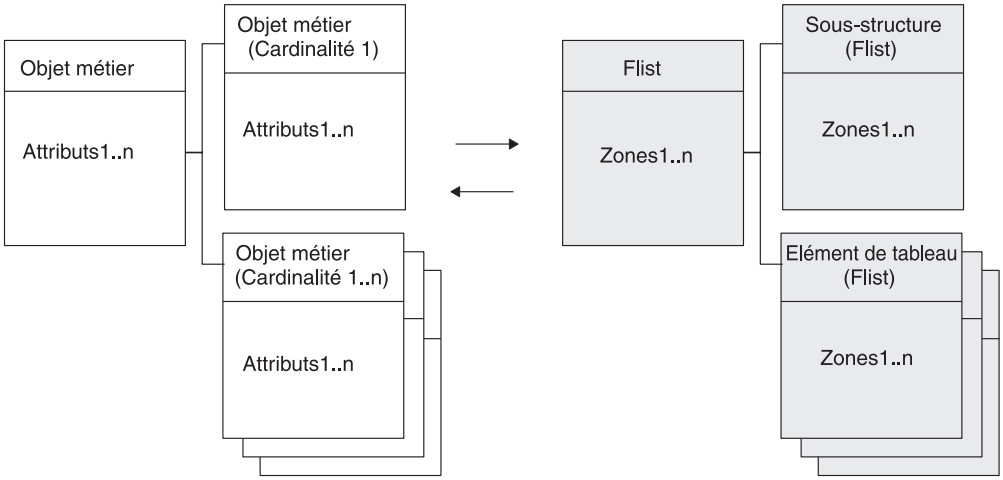


Figure 7. Structure des objets métier et des flists

La figure 8 montre un exemple de coordination possible entre la classe stockable Infranet `/account` et l'objet métier hiérarchique Portal_Account. Le tableau NameInfo de la classe stockable devient un objet métier enfant de cardinalité n dans l'objet métier de niveau supérieur et la sous-structure Balances devient un objet métier enfant de cardinalité 1.

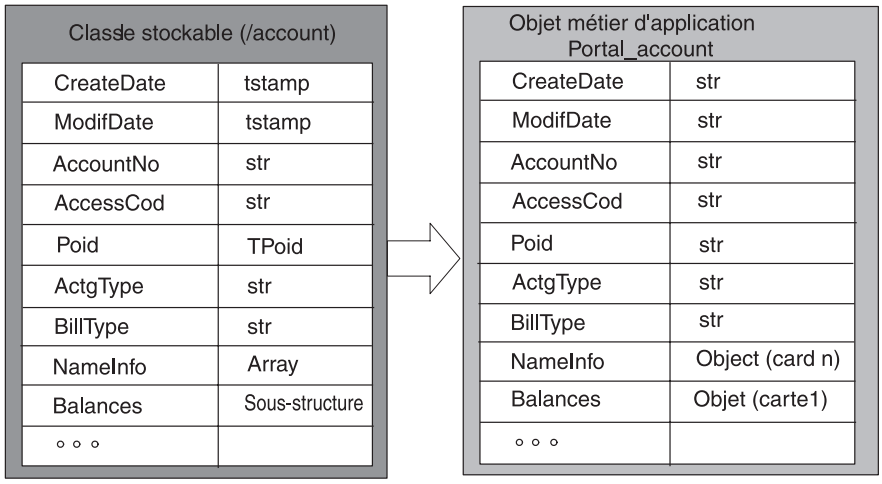


Figure 8. Coordination entre une classe stockable et un objet métier WebSphere Business Integration Adapter

La figure 9 illustre une correspondance possible entre le tableau NameInfo et l'objet métier enfant Portal_Contact. Le tableau NameInfo contient un tableau, appelé Phones, qui devient un objet métier enfant dont l'objet métier parent est Portal_Contact.

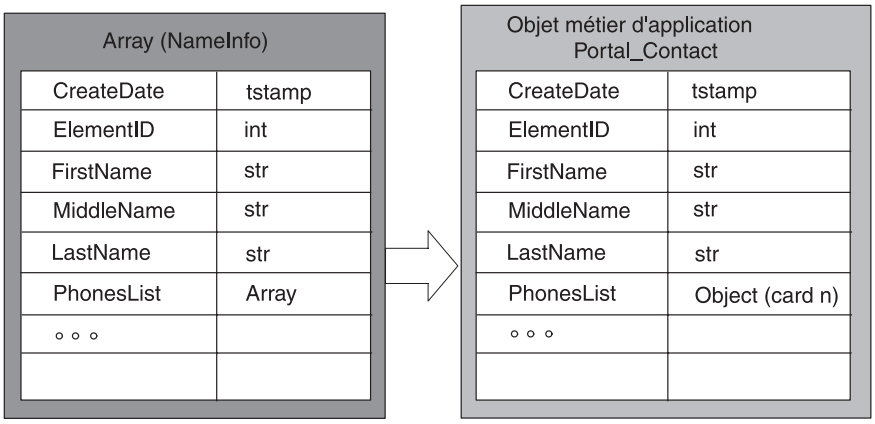


Figure 9. Correspondance entre un tableau de flist et un objet métier enfant

Notez que des attributs spécifiques sont requis pour certaines flists et certains codes opération, mais pas pour d'autres. Dans ce cas, un autre objet métier utilitaire spécifique à l'application peut être employé comme paramètre d'instruction. Cet objet ne correspond à aucune donnée persistante ; il décrit uniquement des zones obligatoires de la flist. Pour de plus amples informations sur les objets métier utilitaires, reportez-vous à la section «Objets métier utilitaires du connecteur», à la page 41.

Propriétés des attributs des objets métier

L'architecture d'un objet métier définit les différentes propriétés s'appliquant aux attributs. Cette section indique comment le connecteur interprète quelques-unes de ces propriétés et décrit la manière de les définir lors de la modification d'un objet métier.

Propriété de clé

Tous les objets métier de Portal Infranet doivent comporter au moins un attribut de clé. Pour chaque attribut correspondant à une clé, définissez la propriété Key sur True.

Remarque : Le connecteur ne prend pas en charge la spécification d'un attribut qui représente un objet métier enfant ou un tableau d'objets métier enfant en tant qu'attribut de clé.

La clé d'un objet métier de niveau supérieur est le POID (Portal Object ID) de l'objet stockable. Un POID est un identificateur sur 64 bits unique, affecté à chaque objet de base de données Infranet lors de sa création. Il s'agit d'une clé unique, associée à un objet Infranet.

Un POID permet plusieurs instanciations de la même classe stockable. Par exemple, chaque objet compte dispose d'un POID unique. Un POID contient les quatre composants suivants : un numéro de base de données, un type d'objet, un ID unique et un numéro de révision.

Valeurs de clés des objets métier enfants

Un tableau Infranet est identifié par son ID d'élément et l'ID d'élément est la clé unique du tableau. Etant donné que les tableaux correspondent généralement à des objets métier enfants, dans les objets métier hiérarchiques Portal Infranet, une clé d'objet métier enfant spécifie l'ID d'élément de tableau et le POID parent.

En règle générale, les objets métier enfants de second niveau n'ont besoin d'un attribut que pour l'ID d'élément. Lors d'une opération de création ou de mise à jour, toutes les valeurs d'ID d'éléments et de POID sont entrées dans l'objet métier.

Propriété de clé étrangère

Cette propriété est utilisée par les objets métier enfants pour établir une relation avec les objets métier parents. Lors du traitement d'une instruction, si l'objet enfant doit être exécuté séparément, les zones de la clé étrangère sont complétées à partir de l'objet métier parent. Le nom de l'attribut de l'objet métier parent utilisé est spécifié dans les informations spécifiques à l'application, au niveau de l'attribut de clé étrangère.

Propriété obligatoire

Le connecteur n'utilise pas la propriété obligatoire.

Propriété de longueur maximale

Définissez la propriété de longueur maximale sur 255.

Propriété de valeur par défaut

Le connecteur utilise la valeur par défaut de l'attribut, si elle est spécifiée.

Instructions relatives à la définition des objets métier

Utilisez les instructions suivantes pour définir un objet métier spécifique à l'application Portal Infranet :

- L'attribut POID d'un objet doit être le premier attribut de sa définition.
- Les autres attributs d'ID de l'objet doivent être indiqués à la suite dans sa définition.
- Les attributs de clés ou de clés étrangères doivent figurer ensuite.
- Les autres attributs, excepté pour les objets référencés et contenus, doivent être indiqués après les attributs de clés, triés de façon logique.
- Tous les objets référencés (cardinalité 1) ou contenus (cardinalité n) doivent figurer à la suite.
- Le dernier attribut doit être ObjectEventId.

Informations d'objets métier spécifiques à l'application

Les informations spécifiques à l'application dans les définitions d'objets métier fournissent au connecteur les instructions dépendantes de l'application concernant le traitement des objets métier. Ces métadonnées sont utilisées avec la structure et les propriétés d'attributs d'un objet métier. Lorsque vous créez des objets métier

spécifiques à l'application Portal Infranet, vous devez veiller à ce que les informations propres à l'application contenues dans la définition d'objet métier correspondent à la syntaxe attendue par le connecteur.

Cette section traite du format des informations d'objets, d'attributs et d'instructions spécifiques à l'application, pour les objets métier associés à Portal.

Informations d'objets métier spécifiques à l'application

Au niveau de l'objet métier, les informations spécifiques à l'application décrivent l'entité Infranet comme suit :

- Pour une classe stockable, les informations d'objets métier spécifiques à l'application spécifient le nom de la classe stockable si la flist d'entrée de l'instruction Create ou Update est semblable à la structure de l'objet métier.
- Si l'objet métier est un objet enfant correspondant à un tableau de flist ou une sous-structure, les informations d'objets spécifiques à l'application contiennent le nom de zone Infranet.

Par exemple, l'objet métier Portal_Account spécifie la classe stockable `CN=/account` pour les informations spécifiques à l'application de niveau objet, tandis que l'objet métier enfant Portal_BillInfo spécifie la zone `FN=PIN_FLD_BILLINFO` pour ce même type d'information.

Informations spécifiques à l'application de niveau attribut

Au niveau attribut, les informations spécifiques à l'application sont utilisées pour préparer la structure flist permettant d'exécuter un code opération particulier. Ces informations correspondent à une liste de paires nom-valeur. Cette structure vous permet de supprimer des contraintes précédemment imposées par l'emploi des objets métier utilitaires, qui ne sont plus nécessaires pour définir la structure d'une flist pour un code opération. Le format des informations spécifiques à l'application d'un attribut est le suivant :

```
FN=FIN_FLD_POID;Create=true;Update=false;Delete=true;
Retrieve=true;CreateT=;UpdateT=PIN_FL_NAMEINFO;
DeleteT=;RetrieveT=;O=false;CreateO=false;UpdateO=true;
DeleteO=false;RetrieveO=false;ParentAtt=;Alone=false
```

Conversion d'anciens objets métier en nouveaux objets

Le format ci-dessus des informations spécifiques à l'application de niveau attribut est pris en charge pour la compatibilité amont dans la version 4.0.x du connecteur. Toutefois, dans les versions ultérieures, la compatibilité amont ne sera pas supportée et vous devrez utiliser PortalODA pour convertir d'anciennes définitions d'objets métier en nouvelles définitions.

Pour convertir au moyen de PortalODA d'anciennes définitions d'objets métier en nouvelles définitions, procédez comme suit :

1. Marquez les zones de clés étrangères dans les objets métier enfants qui associent l'objet métier enfant à l'objet métier parent. Les informations spécifiques à l'application contiennent un indicateur appelé ParentAtt. Définissez la valeur de cet indicateur sur le nom de l'attribut de l'objet métier parent devant être utilisé pour la clé étrangère.
2. Si nécessaire, marquez les attributs de type "object" avec un indicateur d'instruction spécifique. Reportez-vous au tableau 3 qui décrit les zones `CreatT` , `UpdateT`, `DeleteT` et `RetrieveT`.

Le tableau 3, à la page 17 décrit le format des informations spécifiques à l'application qui concernent un attribut :

Tableau 6. Informations spécifiques à l'application concernant un attribut

Nom	Description	Valeur possible	Valeur par défaut
FN	Nom représentant le nom de zone dans Infranet		
Create	Détermine si l'attribut fait partie de la flist associée à l'instruction Create.	true ou false	false
Update	Détermine si l'attribut fait partie de la flist associée à l'instruction Update.	true ou false	false
Delete	Détermine si l'attribut fait partie de la flist associée à l'instruction Delete.	true ou false	false
Retrieve	Détermine si l'attribut fait partie de la flist associée à l'instruction Retrieve.	true ou false	false
CreateT	Nom de zone Infranet jouant le rôle de conteneur pour l'attribut lors de la préparation de la flist associée à l'instruction Create		null
UpdateT	Nom de zone Infranet jouant le rôle de conteneur pour l'attribut lors de la préparation de la flist associée à l'instruction Update		null
DeleteT	Nom de zone Infranet jouant le rôle de conteneur pour l'attribut lors de la préparation de la flist associée à l'instruction Delete		null
RetrieveT	Nom de zone Infranet jouant le rôle de conteneur pour l'attribut lors de la préparation de la flist associée à l'instruction Retrieve		null
O	Détermine si l'attribut doit être mis à jour dans l'objet métier de réponse.	true ou false	false
CreateO	Identifie le nom de la zone Infranet qui représente la valeur de l'attribut utilisé pour mettre à jour l'objet métier de réponse associé au traitement de l'instruction Create. Par exemple, si FN=PIN_FLD_POID et CreateO=PIN_FLD_NAMEINFO, le connecteur recherche la zone PIN_FLD_POID et la valeur de cette zone est entrée dans l'objet métier de réponse associé à cet attribut.		
UpdateO	Identifie le nom de la zone Infranet qui représente la valeur de l'attribut utilisé pour mettre à jour l'objet métier de réponse associé au traitement de l'instruction Update.		

Tableau 6. Informations spécifiques à l'application concernant un attribut (suite)

Nom	Description	Valeur possible	Valeur par défaut
DeleteO	Identifie le nom de la zone Infranet qui représente la valeur de l'attribut utilisé pour supprimer l'objet métier de réponse associé au traitement de l'instruction Delete.		
RetrieveO	Identifie le nom de la zone Infranet qui représente la valeur de l'attribut utilisé pour extraire l'objet métier de réponse associé au traitement de l'instruction Retrieve.		
ParentAtt	Cette zone est utilisée par un objet métier enfant pour définir l'attribut de l'objet métier parent utilisé pour compléter les zones de clés. Ces zones doivent être marquées comme étant des zones de clés étrangères dans l'objet métier enfant.		NULL
Alone	Cette zone est utilisée par un objet métier enfant pour indiquer que l'objet métier enfant doit être exécuté séparément, et non pas dans le cadre de l'objet métier parent.	true ou false	false

La figure 10, à la page 38 illustre l'objet métier Portal_Account et les informations spécifiques à l'application concernant les objets métier et les attributs.

Figure 10. Informations spécifiques à l'application concernant les objets métier et les attributs

Format des informations d'instructions spécifiques à l'application

Les informations d'instructions spécifiques à l'application relatives aux objets métier pour Portal Infranet doivent indiquer un code opération Infranet unique pour l'action que l'objet métier et l'instruction effectueront.

Le code opération transmet des données entre l'application Infranet et sa base de données ; il effectue des opérations sur les objets stockables. Chaque action sur un objet Infranet a un code spécifique.

Les codes opération transmettent des objets stockables Infranet sous la forme de flists, qui sont des listes de paires de noms et de valeurs de zones. Chaque code opération a une flist d'entrée et de sortie spécifique. Le connecteur doit convertir une demande d'objet métier dans la flist d'entrée requise par le code opération. Lors d'une demande d'objet métier, le connecteur suit ces étapes de base :

1. Il crée une flist d'entrée en utilisant les valeurs de l'instance d'objet métier et les informations de la définition d'objet métier.
2. Il exécute le code opération Portal Infranet avec la flist d'entrée en tant qu'argument.
3. Le code opération renvoie une flist de sortie et le connecteur met à jour l'objet métier avec les informations de la flist de sortie.

Les informations d'instructions spécifiques à l'application et dans certains cas, les objets métier utilitaires permettent au connecteur de générer les flists appropriées pour chaque code opération. Un objet métier utilitaire permet au connecteur de créer la flist d'entrée correcte ou de mettre à jour correctement l'objet métier avec la flist de sortie. Les objets métier utilitaires sont fournis dans le cadre de la définition d'objet métier de niveau supérieur. La nouvelle définition des informations d'attributs spécifiques à l'application entraîne la redondance des objets métier utilitaires. La fonction correspondant aux objets métier utilitaires est prise en charge dans la version 4.0.x du connecteur pour la compatibilité amont, mais sera retirée des éditions futures.

Syntaxe des informations d'instructions spécifiques à l'application

La syntaxe requise pour les informations d'instructions spécifiques à l'application est la suivante :

```
<codeop>['#'<indicateur>][';'transaction
enabled][';'<modèle_flist_entrée>][';'<modèle_flist_sortie>]
```

où :

`codeop`	Tout code opération Infranet sans `PCM_OP_`au début de son nom ; `codeop` peut aussi être le mot clé `ERROR` . Le mot clé `ERROR` indique une contrainte Infranet. Le connecteur génère une erreur si l'opération se produit. Cela peut indiquer qu'une opération ne peut pas se produire au niveau enfant. Par exemple, un objet métier Portal_BillInfo ne peut pas être supprimé seul ; il doit être supprimé avec son objet parent.
`indicateur`	Argument facultatif d'un code opération. Pour plus d'informations sur les indicateurs des codes opération, voir la documentation de Portal Infranet.

transaction activée	Certains des codes opération de Portal Infranet gèrent leurs transactions en raison de la nature critique de leur fonctionnalité. Cet indicateur est utilisé pour fournir ces informations au connecteur Portal Infranet. La valeur “true” indique que le code opération gère sa propre transaction. La valeur par défaut utilisée par le connecteur Portal Infranet pour cette propriété est “false.”
modèle_flist_entrée	Nom d’un objet métier utilitaire pour la flist d’entrée et facultativement, des paramètres ou un mot clé. La syntaxe est :<modèle_flist_entrée>[#<paramètre>]Les mots clés possibles sont NotNull , NORMAL ou OnlyPoid . Ces mots clés sont explicités ci-dessous : • NORMAL indique que l’objet métier en cours est le modèle ; en d’autres termes, le connecteur peut effectuer une conversion directe depuis l’objet métier en cours vers la flist d’entrée. • OnlyPoid indique que le code opération en cours requiert uniquement le POID de l’objet métier. Le connecteur peut alors simplifier son traitement. • NotNull indique que cet objet ne doit pas être supprimé en attribuant la valeur Null au tableau. Voir «Objets métier utilitaires du connecteur», à la page 41 pour plus d’informations sur les objets métier utilitaires.
modèle_flist_sortie	Nom d’un objet métier utilitaire pour la flist de sortie et facultativement, des paramètres ou un mot clé. La syntaxe est :<modèle_flist_sortie>[#<paramètre>]Les mots clés possibles sont NoRewrite et Flat. Ces mots clés sont explicités ci-dessous : • NoRewrite indique que la flist de sortie renvoyée par le code opération ne doit pas être utilisée pour écraser l’objet métier. • Flat indique que le connecteur doit obtenir les attributs pour un objet métier enfant depuis l’objet métier de niveau supérieur. Voir «Objets métier utilitaires du connecteur», à la page 41 pour plus d’informations sur les objets métier utilitaires.

Règles de l’application des codes opération

Un ou plusieurs codes opération peuvent exister au niveau de la classe stockable pour les opérations Create, Update et Delete. Cependant, le connecteur ne prend en charge qu’un seul code opération pour chaque instruction. Par conséquent, pour chaque instruction, vous devez choisir le code opération le plus adéquat pour l’objet métier et l’opération d’instruction.

Des codes opération différents peuvent être requis aux niveaux parent et enfant. Lorsqu’un code opération existe pour un objet enfant, Portal Infranet conseille de l’utiliser plutôt que le code opération parent. Si un code opération spécifique est requis pour mettre à jour un sous-composant dans Infranet, vous devez créer un objet métier spécifique à l’application pour cet enfant et indiquer le code opération dans les informations d’instructions spécifiques à l’application.

Lorsque le connecteur crée une flist d’entrée pour un objet métier hiérarchique, si une instruction d’objet métier enfant a le même code opération que le parent, le connecteur place l’enfant dans la même flist que le parent. Sinon, le connecteur crée un flist distincte pour l’enfant. Infranet utilise des niveaux dans les flists pour indiquer des tableaux et des sous-structures. Lorsque le connecteur commence à créer une flist d’entrée, il définit le niveau sur 0. Si l’objet métier a des enfants, le niveau est incrémenté de 1 lors du traitement d’un objet enfant. Si un objet métier

requiert un code opération différent lorsqu'il est exécuté seul plutôt quand dans une hiérarchie, un objet métier différent doit être utilisé avec une structure différente.

Pour les opérations de création, les codes opération parent sont exécutés avant les codes opération enfant. Pour les opérations de suppression, les codes opération enfant sont exécutés avant les codes opération parent. Il n'existe pas d'ordre obligatoire d'exécution pour les opérations de mise à jour et d'extraction.

Toutes les classes stockables peuvent être extraites via le code opération READ_OBJ et le POID de l'objet racine.

Objets métier utilitaires du connecteur

Chaque code opération Infranet requiert une flist d'entrée spécifique et renvoie une flist de sortie spécifique. Pour que le connecteur soit piloté par des métadonnées, l'objet métier doit fournir au connecteur les zones sont il a besoin pour convertir une instance d'objet métier et une instruction vers la flist appropriée. Les flists de codes opération étant différentes, vous ne pourrez peut être pas créer un objet métier pour Portal Infranet qui fournit toutes les informations requises par le connecteur pour chaque flist d'entrée et de sortie. Vous pouvez avoir besoin de définir des définitions d'objets métier utilitaires complémentant la définition d'objet métier spécifique à l'application.

Remarque : La définition des informations d'attributs spécifiques à l'application pour la version 4.0.x du connecteur ne nécessite pas d'objets métier utilitaires. La version 4.0.x du connecteur prend aussi en charge la compatibilité amont pour les informations d'attributs spécifiques à l'application, mais dans les éditions à venir, vous devrez utiliser PortalODA pour convertir d'anciennes définitions d'objets métier en de nouvelles définitions. Voir «Conversion d'anciens objets métier en nouveaux objets», à la page 35pour obtenir des instructions.

Les définitions d'objets métier utilitaires ne deviennent pas des instances d'objets métier envoyées au courtier d'intégration. Le connecteur les utilise simplement pour construire des flists d'entrée et de sortie requises pour des codes opérations spécifiques. Vous devez concevoir et créer des définitions d'objets métier au cours de la conception de l'objet métier spécifique à l'application et vous devez définir le connecteur afin qu'il prenne en charge tous les objets métier utilitaires, ainsi que les objets métier spécifiques à l'application. Pour obtenir une exemple d'objet métier utilitaire, voir «Exemple d'objet métier utilitaire : instruction Create», à la page 42.

Pour déterminer si vous avez besoin d'objets métier utilitaires, examinez la classe stockable Infranet, ainsi que les flists d'entrée et de sortie des codes opération utilisés pour effectuer les opérations d'instructions.

Les objets métier utilitaires de Portal Infranet utilisent des informations spécifiques à l'application qui diffèrent, en terme de format, de celles des objets métier pour Portal. Ce format est décrit dans la section suivante.

Informations spécifiques à l'application pour les objets métier utilitaires

Les informations d'attributs spécifiques à l'application dans les objets métier utilitaires indiquent les zones à ajouter à une flist et contiennent les valeurs à utiliser dans les zones de la flist. Dans les objets métier utilitaires, vous devez

définir des informations d'attributs spécifiques à l'application pour les attributs simples et pour les attributs de conteneur comme suit.

- Pour un attribut simple, indiquez le nom de zone pour la flist et définissez la valeur de cette zone. Une zone peut être extraite de l'attribut correspondant dans l'instance d'objet métier ou il peut s'agir d'une valeur par défaut fournie dans les informations spécifiques à l'application. La syntaxe de cette description est :

  ```
  <nomzone_flist>[:[<nomattribut_objet_métier>]:<valeur_défaut>]
  ```

- Pour les attributs array ou structure, les informations spécifiques à l'application contiennent une liste d'attributs dans l'objet métier spécifique à l'application qui doit être exclue de la flist. Les attributs sont séparés par des points-virgules.

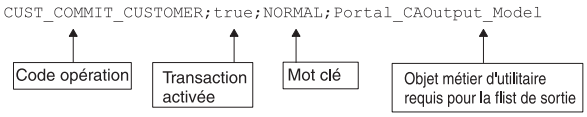
  ```
  [< nomattribut_objet_métier>]([:<
  nomattribut_objet_métier>])*
  ```

La description de l'instruction n'est pas utilisée pour les objets utilitaires.

Les sections ci-après fournissent des exemples d'informations spécifiques à l'application pour les instructions Create, Update, Retrieve et Delete. Les exemples utilisent l'objet métier hiérarchique Portal_Account pour illustrer des aspects des informations d'instructions spécifiques à l'application.

Exemple d'objet métier utilitaire : instruction Create

Par exemple, utilisons les informations d'instructions spécifiques à l'application pour l'instruction Create dans l'objet métier de niveau supérieur Portal_Account.



CUST_COMMIT_CUSTOMER est le code opération utilisé par le connecteur pour créer un compte client (un objet stockable /account). Le code opération a sa propre transaction, par conséquent, "Transaction activée" a la valeur "true." Le mot clé NORMAL dans la zone modèle_flist_entrée indique que le connecteur va établir une correspondance directe depuis l'objet métier vers la flist d'entrée. En d'autres termes, l'objet métier fournit toutes les zones requises par la flist d'entrée et le connecteur n'a pas besoin d'informations supplémentaires pour créer une flist d'entrée.

Le code opération CUST_COMMIT_CUSTOMER pour l'opération Create Account renvoie une flist de sortie contenant l'ID pour le nouveau client dans la zone PIN_FLD_ACCOUNT_OBJ. La valeur de cet ID doit être renvoyée au système WebSphere Business Integration Adapter. Pour que le connecteur puisse obtenir le nouvel ID, le concepteur d'objet métier a créé la définition d'objet métier utilitaire Portal_C[reate]A[ccount]Output_Model. La zone modèle_flist_sortie dans les informations spécifiques à l'application indique Portal_CAOutput_Model en tant qu'objet métier utilitaire que le connecteur va utiliser pour lire la flist de sortie renvoyée par le code opération.

L'objet utilitaire Portal_CAOutput_Model contient un attribut, Poid, dont les informations spécifiques à l'application indiquent au connecteur d'extraire la valeur de PIN_FLD_ACCOUNT_OBJ de la flist renvoyée pour obtenir l'ID d'objet Portal Infranet du nouveau client. Le connecteur insère cette valeur dans l'objet métier qu'il renvoie au courtier d'intégration. L'objet métier utilitaire figure dans figure 11.

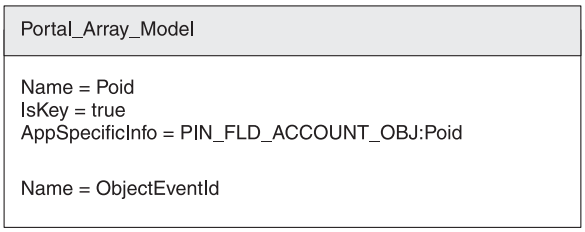


Figure 11. Définition d'objet métier utilitaire Portal_CAOutput_Model

Création de flists pour les opérations Create : L'objet métier Portal_Account est un objet métier hiérarchique semblable à celui présenté dans la figure 12.

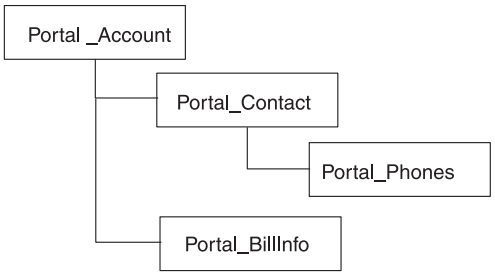


Figure 12. Diagramme de l'objet métier hiérarchique Portal_Account

Lors d'une opération Create, le connecteur examine les informations d'instructions spécifiques à l'application pour les objets métier enfants de façon à déterminer si le code opération est identique à celui utilisé par l'objet métier parent. Concernant un objet métier Portal_Account, le code opération est identique pour les objets métier parents et enfants et le connecteur peut créer une seule flist pour l'opération de création d'objet métier globale. La figure 13 illustre le code opération et la flist uniques que le connecteur utilise pour générer l'appel Create vers Infranet.

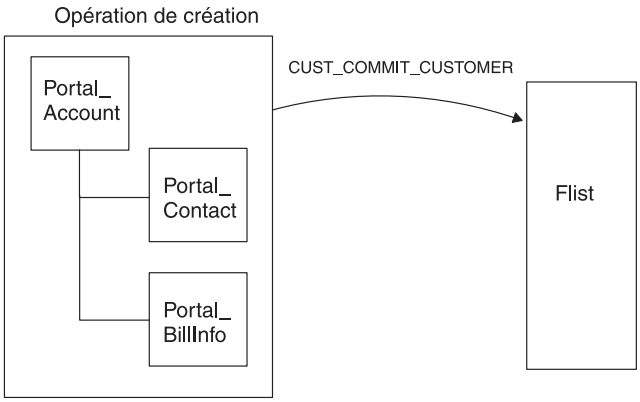


Figure 13. Flist associée à l'opération de création d'objet métier hiérarchique Portal_Account

Notez toutefois que le connecteur doit créer la flist avec des tableaux pour les objets métier enfants. Par conséquent , les informations d'instructions spécifiques à l'application doivent inclure des zones indiquant le niveau auquel le tableau doit se trouver dans la flist. Par exemple, les informations d'instruction Create spécifiques à l'application concernant l'objet métier enfant Portal_Contact sont les suivantes :

```
CUST_COMMIT_CUSTOMER
```

Les informations d'instruction Create spécifiques à l'application concernant l'objet métier enfant Portal_Phone sont les suivantes :

```
CUST_COMMIT_CUSTOMER
```

Exemple d'objet métier utilitaire : instruction Update

Cet exemple présente les informations d'instruction Update spécifiques à l'application qui sont contenues dans l'objet métier de niveau supérieur Portal_Account :

```
CUST_SET_STATUS;false;Portal_Array_Model#PIN_FLD_STATUSES;NoRewrite
```

`CUST_SET_STATUS` est le code opération requis pour la mise à jour d'un objet compte. Le code opération n'est pas associé à sa propre transaction ; par conséquent, le paramètre "Transaction Enabled" a été défini sur "false." Pour cette opération d'instruction, le connecteur ne peut pas faire correspondre directement l'instance d'objet métier avec la flist car l'objet métier Portal_Account ne fournit pas toutes les informations requises par la flist d'entrée. Etant donné que le connecteur a besoin d'informations complémentaires pour créer la flist, la zone `modèle_flist_entrée` spécifie la définition d'objet métier utilitaire que le connecteur utilise pour construire la flist d'entrée. Cet objet utilitaire est appelé Portal_Array_Model.

La zone `modèle_flist_sortie` contient le mot clé renvoyer`NoRewrite` indiquant que la flist de sortie renvoyée par le code opération ne doit pas être utilisée pour écraser l'objet métier.

Objet métier utilitaire Portal_Array_Model : Portal_Array_Model est une définition d'objet métier hiérarchique illustrée dans la figure 14. Elle contient les informations dont le connecteur a besoin pour créer la flist d'entrée pour le code de l'opération Update. Tout particulièrement, la flist d'entrée requiert un tableau que la définition d'objet métier Portal_Account ne contient pas. L'objet utilitaire Portal_Array_Model permet au connecteur de créer le tableau.

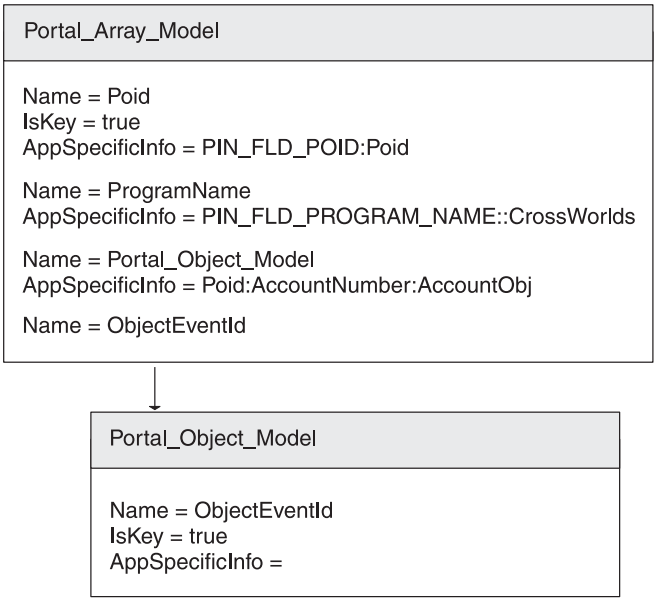


Figure 14. Définition d'objet métier utilitaire Portal_Array_Model

Notez que le texte suivant est associé à la zone modèle_flist_entrée dans les informations d'instructions Update spécifiques à l'application qui concernent Portal_Account :

```
Portal_Array_Model#PIN_FLD_STATUSES
```

Ce texte identifie l'objet métier utilitaire permettant de créer la flist et indique le nom du tableau que le connecteur doit placer dans la flist d'entrée. Lors de l'exécution, le connecteur utilise la définition d'objet métier utilitaire et la définition d'objet métier Portal_Account spécifique à l'application pour créer la flist d'entrée.

Le connecteur crée la flist d'entrée comme suit :

1. Il commence à construire la flist avec une zone PIN_FLD_POID et extrait la valeur du POID de l'instance d'objet métier.
2. Il ajoute une zone pour PIN_FLD_PROGRAM_NAME à la flist. Etant donné que la définition d'objet métier Portal_Account ne contient pas cet attribut, aucune valeur ne lui est associée dans l'instance d'objet métier. Par conséquent, la valeur est définie dans les informations spécifiques à l'application sous la forme de la chaîne CrossWorlds.
3. Il ajoute le tableau PIN_FLD_STATUSES à la flist. Etant donné que la flist associée au code opération PCM_OP_CUST_SET_STATUS requiert un tableau pour PIN_FLD_STATUSES, l'objet Portal_Array_Model doit inclure un attribut de conteneur pour que le connecteur crée un tableau dans la flist. Le connecteur nomme le tableau comme indiqué dans les informations spécifiques à l'application concernant l'instruction Update.

   Le connecteur utilise l'objet métier en cours, Portal_Account, comme modèle de tableau. En d'autres termes, le connecteur va insérer dans le tableau de flist les zones spécifiées dans l'objet métier en cours. Etant donné que certaines zones ne sont peut-être pas nécessaires, les informations spécifiques à l'application concernant l'attribut de conteneur qui sont contenues dans l'objet métier utilitaire indiquent les attributs à ignorer. L'attribut de conteneur Portal_Object_Model spécifie les attributs suivants :

   ```
   Poid:AccountNumber:AccountObj
   ```

   Par conséquent, pour construire le tableau, le connecteur examine la définition d'objet métier associée à Portal_Account, ignore les attributs POID, AccountNumber et AccountObj, et crée le tableau en utilisant uniquement les autres attributs de la définition d'objet métier, PIN_FLD_STATUS et PIN_FLD_STATUS_FLAGS.

La flist d'entrée associée à l'instruction Update se présente donc comme suit :

```
PIN_FLD_POID              POID    <valeur de l'instance d'objet métier>
PIN_FLD_PROGRAM_NAME      STR     "CrossWorlds"
PIN_FLD_STATUSES          ARRAY
   PIN_FLD_STATUS         ENUM    <valeur de l'instance d'objet métier>
   PIN_FLD_STATUS_FLAGS     INT      <valeur de l'instance d'objet métier>
```

Cette flist contient les zones obligatoires associées à la flist d'entrée pour le code opération PCM_OP_CUST_SET_STATUS.

Traitement des objets métier enfants : Pour une opération Update, les codes opération Infranet requis sont différents selon qu'il s'agit de mettre à jour un objet stockable de compte, les informations sur le contact client contenues dans l'objet stockable de compte ou les informations de facturation de l'objet stockable de compte.

Par conséquent, même si les informations sur le contact client et les informations de facturation client font partie de la même classe stockable, le connecteur doit utiliser des codes opération différents pour mettre à jour l'objet métier de niveau supérieur Portal_Account et les objets métier enfants Portal_Contact et Portal_BillInfo. En outre, le connecteur doit générer des flists d'entrée séparées pour chaque code opération. La figure 15 illustre l'ensemble de flists requis pour mettre à jour l'objet métier hiérarchique Portal_Account.

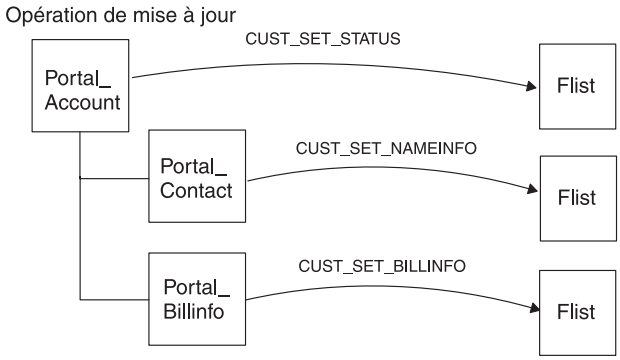


Figure 15. Flists associées à l'opération de mise à jour de l'objet métier hiérarchique Portal_Account

Exemple d'objet métier utilitaire : instruction Retrieve

Cet exemple illustre les informations d'instruction Retrieve spécifiques à l'application qui sont contenues dans l'objet métier de niveau supérieur Portal_Account :

```
READ_OBJ;false;Only_Poid;Flat#Portal_BillInfo
```

Le connecteur utilise le code opération `READ_OBJ` pour lire un objet stockable de la base de données. Ce code opération n'étant pas associé à une transaction, le paramètre "Transaction Enabled" a été défini sur "false." En général, il peut être utilisé pour toutes les opérations Retrieve.

La zone `modèle_flist_entrée` spécifie que le code opération en cours requiert uniquement le POID de l'objet métier. Aucune autre zone n'est nécessaire pour la flist d'entrée.

Le code opération renvoie le POID de l'objet. Toutes les autres zones de l'objet, y compris l'ensemble des éléments de tableau, sont ajoutées à la flist renvoyée, à la suite du POID. La zone `modèle_flist_sortie` contient le mot clé `Flat` associé au paramètre Portal_BillInfo. Cela indique que les informations dont le connecteur a besoin pour créer l'objet métier enfant Portal_BillInfo sont contenues dans la flist Flat plutôt que dans un tableau.

Exemple d'objet métier utilitaire : instruction Delete

Le dernier exemple illustre les informations d'instruction Delete spécifiques à l'application qui sont contenues dans l'objet métier de niveau supérieur Portal_Account :

```
CUST_DELETE_ACCT;false;Only_Poid;NoRewrite
```

Le connecteur utilise le code opération `CUST_DELETE_ACCT` pour supprimer l'objet stockable de la base de données. Ce code opération n'étant pas associé à une transaction, le paramètre "Transaction Enabled" a été défini sur "false." La zone `modèle_flist_entrée` spécifie que le code opération en cours requiert uniquement

le POID de l'objet métier. Aucune autre zone n'est nécessaire pour la flist d'entrée. La zone modèle_flist_sortie contient le mot clé NoRewrite indiquant que la flist de sortie renvoyée par le code opération ne doit pas être utilisée pour écraser l'objet métier.

Notez qu'Infranet est une application de suppression logique. Pour certains objets, une opération de suppression est un changement d'état. Par exemple, pour l'objet métier Portal_Service, les informations d'instruction Delete spécifiques à l'application sont les suivantes :

```
CUST_SET_STATUS;Portal_DSInput_Model#PIN_FLD_STATUSES#NotNull
```

Ce texte spécifie CUST_SET_STATUS en tant que code de l'opération Delete logique et indique que modèle_flist_entrée correspond au paramètre Portal_D[elete]S[ervice]Input_Model de l'objet utilitaire, qui définit une flist associée à un tableau PIN_FLD_STATUSES non défini sur Null.

Fonctionnement d'une instruction dépendant du contexte

Etant donné que le connecteur doit pouvoir traiter un objet métier hiérarchique et un objet métier enfant unique (par exemple, un objet métier Portal_Contact peut être envoyé sans l'objet parent correspondant), certaines décisions commandées par les métadonnées dépendent du contexte de l'objet métier et de l'instruction. Selon l'instruction, vous devez spécifier un code opération unique pour toute la hiérarchie d'objets métier ou un code opération pour chaque objet métier. Pour cette raison, il convient d'indiquer le code opération utilisé par chaque niveau de chaque objet métier.

Pour une instruction, le code opération global (code opération parent) est appliqué si les codes opération du niveau enfant et du niveau parent sont similaires. Dans le cas contraire, le code opération du parent est appliqué en premier, suivi du code opération de chaque enfant.

Par exemple, lorsque vous devez créer un contact, si Portal_Contact est envoyé en tant qu'objet métier individuel, utilisez le code opération CUST_SET_NAMEINFO pour les informations d'instruction Create spécifiques à l'application. Toutefois, si le contact doit être créé avec l'objet métier de compte, utilisez le code opération parent CUST_COMMIT_CUSTOMER. Ce code opération doit être spécifié dans les informations spécifiques à l'application. Pour prendre en charge la fonctionnalité ci-dessus, deux copies de l'objet métier Portal_Contact doivent être créées et utilisées pour deux codes opération différents. Pour un objet métier, l'instruction ASI sera définie sur CUST_SET_NAMEINFO, tandis que pour l'autre, elle sera définie sur CUST_COMMIT_CUSTOMER.

Exemple de définition complète d'objet métier Portal Infranet

La structure ci-dessous décrit les propriétés et informations spécifiques à l'application qui concernent l'objet métier Sample Account dans l'adaptateur Portal Infranet.

```
[BusinessObjectDefinition]
Name = Portal_Account
Version = 1.0.0
AppSpecificInfo = CN=/account
[Attribute]
Name = Poid
Type = String
Cardinality = 1
MaxLength = 255
IsKey = true
IsForeignKey = false
```

```
IsRequired = true
AppSpecificInfo = eu
IsRequiredServerBound = false
[End]
[Attribute]
Name = AccountNumber
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo =
   UpdateO=;FN=PIN_FLD_ACCOUNT_NO;Create=true;O=true;DeleteT=;
   Update=falTse;RetrieveT=;Alone=false;CreateT=;Retrieve=false;
   DeleteO=;ParentAtt=;UpdateT=;RetrieveO=Main;Delete=false;CreateO=
IsRequiredServerBound = false
[End]

[Attribute]
Name = AccountObj
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo =
   UpdateO=;FN=PIN_FLD_ACCOUNT_OBJ;Create=true;O=true;DeleteT=;Update=false;
   RetrieveT=;Alone=false;CreateT=;Retrieve=false;DeleteO=;ParentAtt=;
   UpdateT=;RetrieveO=Main;Delete=false;CreateO=
IsRequiredServerBound = false
[End]

[Attribute]
Name = Status
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo =
   UpdateO=;FN=PIN_FLD_STATUS;Create=true;O=true;DeleteT=;Update=true;
   RetrieveT=;Alone=false;CreateT=;DeleteO=;Retrieve=false;ParentAtt=;
   RetrieveO=Main;UpdateT=PIN_FLD_STATUSES;CreateO=;Delete=false
IsRequiredServerBound = false
[End]

[Attribute]
Name = StatusReason
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo =
   UpdateO=;FN=PIN_FLD_STATUS_FLAGS;Create=true;O=true;DeleteT=;
   Update=true;RetrieveT=;Alone=false;CreateT=;DeleteO=;Retrieve=false;
   ParentAtt=;RetrieveO=Main;UpdateT=PIN_FLD_STATUSES;CreateO=;Delete=false
IsRequiredServerBound = false
[End]

[Attribute]
Name = Portal_Contact
Type = Portal_Contact
```

```
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = n
MaxLength = 0
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo =
   UpdateO=Main;FN=PIN_FLD_NAMEINFO;Create=true;O=true;Update=true;
   Alone=false;Retrieve=false;DeleteO=Main;ParentAtt=;RetrieveO=Main;
   CreateO=Main;Delete=false
IsRequiredServerBound = false
[End]

[Attribute]
Name = PlaceHolder
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]

[Attribute]
Name = Portal_BillInfo
Type = Portal_BillInfo
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = n
MaxLength = 0
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo =
   UpdateO=Main;FN=PIN_FLD_BILLINFO;Create=true;O=true;Update=true;
   Alone=false;Retrieve=false;DeleteO=Main;ParentAtt=;RetrieveO=Main;
   CreateO=Main;Delete=false
IsRequiredServerBound = false
[End]

[Attribute]
Name = ProgramName
Type = String
Cardinality = 1
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo =
   UpdateO=Main;FN=PIN_FLD_PROGRAM_NAME;Create=false;O=true;DeleteT=;
   Update=true;RetrieveT=;Alone=false;CreateT=;DeleteO=Main;Retrieve=false;
   ParentAtt=;PAttName=;RetrieveO=Main;UpdateT=;CreateO=;Delete=true
DefaultValue = CrossWorlds
IsRequiredServerBound = false
[End]

[Attribute]
Name = ObjectEventId
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
```

```
[End]

[Verb]
Name = Create
AppSpecificInfo =
   IFM=;OpCode=CUST_COMMIT_CUSTOMER;OFP=;TFlag=true;
   IFP=;IF=NORMAL;Flag=0;OF=NORMAL
[End]

[Verb]
Name = Delete
AppSpecificInfo =
   IFM=;OpCode=CUST_DELETE_ACCT;OFP=;TFlag=false;IFP=;
   IF=NORMAL;Flag=0;OF=NORMAL
[End]

[Verb]
Name = Retrieve
AppSpecificInfo =
   IFM=;OpCode=READ_OBJ;OFP=Portal_BillInfo;TFlag=false;IFP=;
   IF=NORMAL;Flag=0;OF=NORMAL
[End]

[Verb]
Name = Update
AppSpecificInfo =
   IFM=;OpCode=CUST_SET_STATUS;OFP=;TFlag=false;
   IFP=;IF=NORMAL;Flag=0;OF=NORMAL
[End]

[End]
```

Chapitre 4. Génération de définitions d'objets métier à l'aide de PortalODA

Ce chapitre décrit PortalODA, un agent de reconnaissance d'objets (ou agent ODA) qui génère des définitions d'objets métier pour le connecteur. Le composant PortalODA utilise les API Portal Infranet pour obtenir les informations relatives aux classes stockables Portal Infranet. Ensuite, il se sert de ces informations pour générer de nouvelles définitions d'objets métier. PortalODA permet également la conversion des définitions d'objets métier existantes en définitions prises en charge par le connecteur.

Les sujets traités sont les suivants :

Installation et utilisation

Cette section aborde les points suivants :

Installation de PortalODA

Pour installer PortalODA, utilisez le programme d'installation de WebSphere Business Integration Adapter (WBIA). Suivez les instructions contenues dans le document *System Installation Guide for UNIX* ou *for Windows*. Une fois l'installation terminée, vous devez installer les fichiers suivants dans le répertoire où vous avez installé le produit sur votre système :

- `ODA\Portal\PortalODA.jar`
- `ODA\messages\PortalODAAgent.txt`
- `ODA\Portal\start_PortalODA.bat` (Windows uniquement)
- `ODA/Portal/start_PortalODA.sh` (UNIX uniquement)
- `bin\CWODAEnv.bat` (Windows uniquement)
- `bin/CWODAEnv.sh` (UNIX uniquement)

Remarque : Sauf indication contraire, ce document utilise les barres obliques inverses (\) comme convention dans les chemins de répertoire. Pour les systèmes UNIX, remplacez les barres obliques inverses par des barres obliques (/). Tous les noms de chemin des produits WBIA sont associés au répertoire dans lequel le produit est installé sur votre système.

Avant d'utiliser PortalODA

Pour pouvoir exécuter PortalODA, vous devez copier les fichiers .jar requis de l'application Portal Infranet dans le répertoire *%ProductDir%*/connectors/Portal/dependencies. Les fichiers suivants doivent être copiés dans ce répertoire :

```
pcm.jar
pcmext.jar
```

Les fichiers ci-dessus se trouvent dans le dossier %INFRANET%\jars.

Une fois le composant PortalODA installé, vous devez effectuer les opérations suivantes pour générer ou convertir des objets métier :

1. Lancez l'agent ODA.
2. Démarrez Business Object Designer.
3. Suivez un processus à six étapes dans Business Object Designer pour configurer et exécuter l'ODA.

Les sections suivantes décrivent chaque étape en détail.

Démarrage de PortalODA

Vous pouvez démarrer PortalODA à l'aide d'un des scripts suivantes :

UNIX :

```
start_PortalODA.sh
```

Windows :

```
start_PortalODA.bat
```

Configurez et exécutez PortalODA à l'aide de l'outil Business Object Designer. Business Object Designer recherche chaque ODA par le nom indiqué dans la variable AGENTNAME de chaque script ou fichier de traitement par lots. Le nom par défaut de l'agent ODA défini pour ce connecteur est PortalODA.

Exécution de PortalODA sur plusieurs machines

Vous pouvez exécuter plusieurs instances de l'agent ODA, soit sur l'hôte local, soit sur un hôte distant du réseau. Si plusieurs instances s'exécutent sur une même machine, chacune d'elle doit utiliser un port unique.

La figure 16, à la page 54 illustre la fenêtre dans Business Object Designer à partir de laquelle vous sélectionnez l'ODA à exécuter.

Changement du nom du fichier de messages et d'erreurs

Le fichier de messages d'erreur et de trace (PortalODAAgent.txt) se trouve dans le répertoire \ODA\messages\, sous le répertoire produit. Ce fichier utilise la convention de dénomination suivante :

```
AgentNameAgent.txt
```

Si vous renommez un agent ODA dans la variable AGENTNAME d'un fichier de script ou de traitement par lots, utilisez cette convention pour renommer le fichier de messages d'erreur et de trace associé.

Si vous créez plusieurs instances du fichier de script ou de traitement par lots et indiquez un nom unique pour chaque agent ODA représenté, créez une copie du fichier de messages d'erreur et de trace pour chacun de ces agents. Attribuez un nom à chaque fichier conformément à cette convention. Par exemple, si la variable `AGENTNAME` indique `PortalODA1`, nommez le fichier de messages associé : `PortalODA1Agent.txt`.

Au cours du processus de configuration, indiquez les éléments suivants :

- le nom du fichier dans lequel le composant PortalODA enregistre les informations d'erreur et de trace ;
- le niveau de traçage, compris entre `0` et `5`.

Le tableau 7 décrit les valeurs du niveau de traçage.

Tableau 7. Niveaux de traçage

Niveau de trace	Description
0	Consigne toutes les erreurs
1	Effectue un traçage de tous les messages d'entrée et de sortie pour la méthode
2	Effectue un traçage des propriétés de l'ODA et de leurs valeurs
3	Effectue un traçage des noms de tous les objets métier
4	Effectue un traçage du détail des unités d'exécution générées
5	• Indique les valeurs d'initialisation de l'ODA pour toutes ses propriétés • Effectue un traçage de l'état détaillé de chaque unité d'exécution générée par PortalODA • Exécute le traçage du vidage des définitions d'objets métier

Pour plus d'informations sur l'emplacement dans lequel vous devez configurer ces valeurs, voir «Configuration des propriétés d'initialisation», à la page 54.

Utilisation de PortalODA dans Business Object Designer

Cette section explique comment utiliser PortalODA dans Business Object Designer pour convertir les définitions d'objets métier existantes et pour en générer de nouvelles. Pour ce faire, consultez les informations disponibles directement sur Portal Infranet. Pour plus d'informations sur le démarrage de Business Object Designer, voir le manuel *Business Object Development Guide*.

Après avoir démarré un agent ODA, vous devez démarrer Business Object Designer afin de le configurer et de l'exécuter. Dans Business Object Designer, la génération ou la conversion d'une définition d'objet métier à l'aide d'un ODA comprend six étapes. Business Object Designer propose un assistant pour vous guider à travers ces étapes.

Après avoir démarré l'ODA, procédez comme suit pour démarrer l'assistant :

1. Ouvrez Business Object Designer.
2. Dans le menu File, sélectionnez le sous-menu New Using ODA....

   Business Object Designer affiche la première fenêtre de l'assistant, Select Agent. La figure 16, à la page 54 représente cette fenêtre.

Pour sélectionner, configurer et exécuter l'ODA, procédez comme suit :

1. «Sélection de l'ODA»
2. «Configuration des propriétés d'initialisation», à la page 54
3. «Génération de définitions», à la page 59 et, le cas échéant, «Ajout d'informations complémentaires», à la page 59
4. «Enregistrement des définitions», à la page 61

Sélection de l'ODA

La figure 16, à la page 54 illustre la première boîte de dialogue de l'assistant de Business Object Designer, qui comprend six étapes.

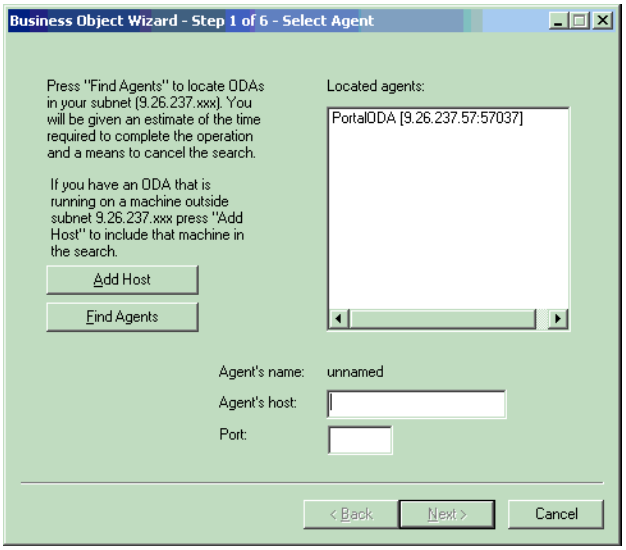


Figure 16. Sélection de l'ODA

Pour sélectionner l'ODA :

1. Cliquez sur le bouton Find Agents pour afficher tous les ODA enregistrés ou en cours d'exécution dans la zone Located agents.

   Vous pouvez également rechercher un agent par nom d'hôte et numéro de port.

   Remarque : Si Business Object Designer ne localise pas l'ODA souhaité, vérifiez la configuration de l'ODA.
2. Sélectionnez l'ODA dans la liste affichée.

   Business Object Designer affiche votre sélection dans la zone Agent's name.

Configuration des propriétés d'initialisation

La première fois que Business Object Designer communique avec PortalODA, il vous invite à entrer un ensemble de propriétés d'initialisation, comme illustré par la figure 17, à la page 55. Vous pouvez enregistrer ces propriétés dans un profil nommé, ainsi vous n'aurez plus besoin des les entrer chaque fois que vous utiliserez PortalODA. Pour plus d'informations sur l'indication d'un profil ODA, voir le document *Business Object Development Guide*.

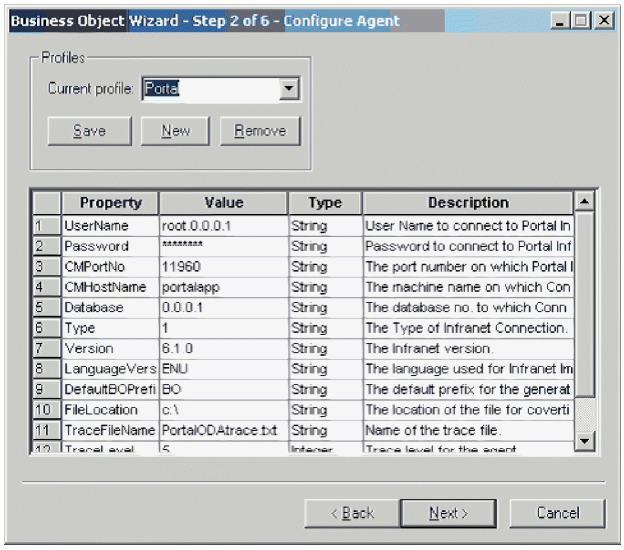


Figure 17. Configuration des propriétés d'initialisation de l'agent

Configurez les propriétés de PortalODA décrites dans le tableau 8.

Important : Toutes les propriétés de PortalODA citées dans le tableau 8 doivent obligatoirement être définies.

Tableau 8. Propriétés de PortalODA

Numéro de ligne	Nom de propriété	Type de propriété	Description
1	`UserName`	String	Nom de connexion à l'application Portal Infranet
2	`Password`	String	Mot de passe de l'application Portal Infranet
3	`CMPortNo`	String	Numéro du port sur lequel le gestionnaire de connexions s'exécute
4	CMHostName	String	Nom ou adresse IP de la machine sur laquelle le gestionnaire de connexions s'exécute
5	Database	String	Numéro de la base de données à laquelle le gestionnaire de connexions est connecté
6	Type	String	Type de connexion de Portal Infranet : 1 correspond à la validation des propriétés UserName et Password, 0 signifie qu'il n'y a pas de validation
7	Version	String	Version de Portal Infranet
8	`LanguageVersion`	String	Exemple : ENU pour l'anglais
9	`DefaultBOPrefix`	String	Exemple : Portal_BO
10	FileLocation	String	Chemin d'accès absolu contenant les fichiers des versions antérieures de définitions d'objets métier. Par exemple, sous Windows, si le chemin d'accès est `C:\PortalBos`, vous devez entrer la valeur `C:\\Portal\\`. Sous UNIX, si le chemin d'accès est `/home/PortalBOs`, vous devez entrer la valeur `/home/PortalBos/`.
11	TraceFileName	String	Nom du fichier de trace
12	`TraceLevel`	Integer	Texte ajouté avant le nom de l'objet métier pour le rendre unique. Vous pouvez modifier le nom par la suite, lorsque Business Object Designer vous demande d'entrer les propriétés de l'objet métier. Pour plus d'informations, voir «Ajout d'informations complémentaires», à la page 59
13	MessageFile	String	Chemin d'accès au fichier de messages

Développement de noeuds, sélection de fichiers référentiels et de classes stockables

Après avoir configuré toutes les propriétés d'initialisation de PortalODA, Business Object Designer affiche l'écran suivant.

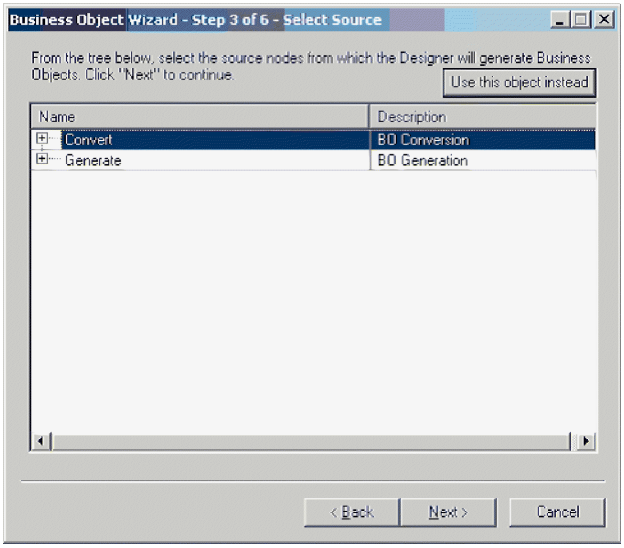


Figure 18. Arborescence offrant deux options de conversion et de génération d'objets métier

Cet écran présente deux options pouvant être développées, Convert et Generate. Si vous devez convertir les anciennes définitions d'objets métier en de nouvelles définitions, développez le noeud Convert. Les fichiers référentiels contenant les définitions d'objets métier à convertir s'affichent.

Conversion d'anciennes définitions d'objets métier

Les anciennes définitions d'objets métier possèdent des informations spécifiques à chaque application présentées sous forme de valeurs séparées par des virgules, tandis que les nouvelles définitions d'objets métier possèdent des informations spécifiques à chaque application présentées sous forme de paires nom-valeur séparées par des virgules. De plus, les anciennes définitions d'objets métier utilisent des méta-objets métier pour transformer la structure d'un objet métier associé à un code opération particulier alors que, dans les nouvelles définitions d'objets métier, cette fonction est remplacée par la paire nom-valeur d'informations spécifiques à chaque application au niveau de l'attribut de l'objet métier.

Sélectionnez les fichiers à convertir, puis cliquez sur Next.

Remarque : Lorsque vous sélectionnez un fichier, toutes les définitions d'objets métier qu'il contient sont converties. Aucune méthode n'est préconisée pour sélectionner un sous-ensemble de définitions d'objets métier à convertir. Toutefois, si vous voulez convertir uniquement un sous-ensemble de définitions d'objets métier, vous pouvez créer un fichier contenant le sous-ensemble souhaité puis le convertir.

Génération de nouveaux objets métier

Si vous devez générer de nouvelles définitions d'objets métier à l'aide d'informations obtenues dans Portal Infranet, développez le noeud Generate. Cette option vous permet d'obtenir tous les noms de classes stockables depuis Portal Infranet et affiche une arborescence.

Ces noms de classes stockables, représentés par des noeuds dans l'arborescence, peuvent être développés (voir figure 19, à la page 57). Les objets métier générés ont des propriétés devant être définies individuellement pour qu'ils puissent être utilisés par le connecteur. Les zones de clé d'un objet métier doivent être marquées en tant que zones de clé dans l'objet métier du système WebSphere Business Integration. En fonction du code opération utilisé pour les différentes instructions, les informations spécifiques à chaque application au niveau des attributs doivent être définies. Ainsi, si un attribut fait partie du code opération de l'instruction Create, la valeur de la propriété "Create" doit être définie en fonction du nom de la zone parent. Reportez-vous à la section «Informations spécifiques à l'application de niveau attribut», à la page 35 pour obtenir des détails sur les diverses propriétés contenues dans les informations spécifiques aux applications d'un attribut.

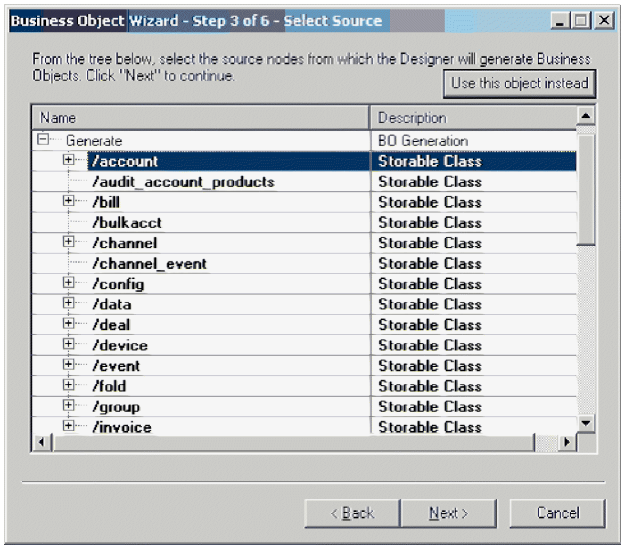


Figure 19. Ecran présentant les classes stockables

Cet écran vous permet de sélectionner une classe stockable à générer dans la liste proposée. Le signe "+" placé avant un nom de classe signifie que la classe possède des objets enfants. Il est possible de sélectionner plusieurs classes à générer.

Remarque : Lorsque vous sélectionnez une classe à générer qui possède des objets enfants, par défaut ces derniers ne sont pas sélectionnés. Vous devez sélectionner de façon explicite les objets enfants si vous souhaitez également les générer. Pour ce faire, maintenez la touche Maj enfoncée tout en sélectionnant les objets enfants.

Confirmation de la sélection des fichiers référentiels et des classes stockables

Après avoir identifié tous les fichiers référentiels ou toutes les classes stockables à associer à la définition d'objets métier générée, Business Object Designer affiche l'écran de confirmation suivant (voir figure 20, à la page 58).

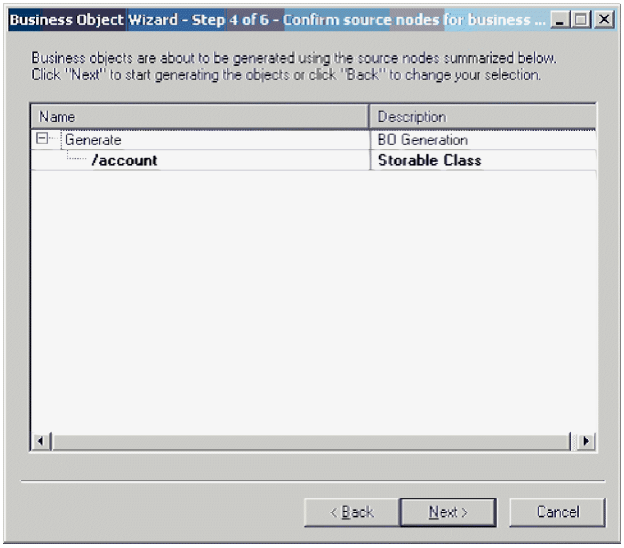


Figure 20. Confirmation de votre sélection

Cette fenêtre contient les options suivantes :

- Pour confirmer la sélection, cliquez sur Next.
- Si la sélection est incorrecte, cliquez sur Back pour revenir à la fenêtre précédente et apporter les modifications nécessaires. Lorsque vous avez terminé, cliquez sur Next.

Génération de définitions

Après avoir confirmé la sélection des objets de la base de données, la boîte de dialogue suivante vous informe que Business Object Designer procède à la création des définitions.

La figure 21, à la page 59 représente cette boîte de dialogue.

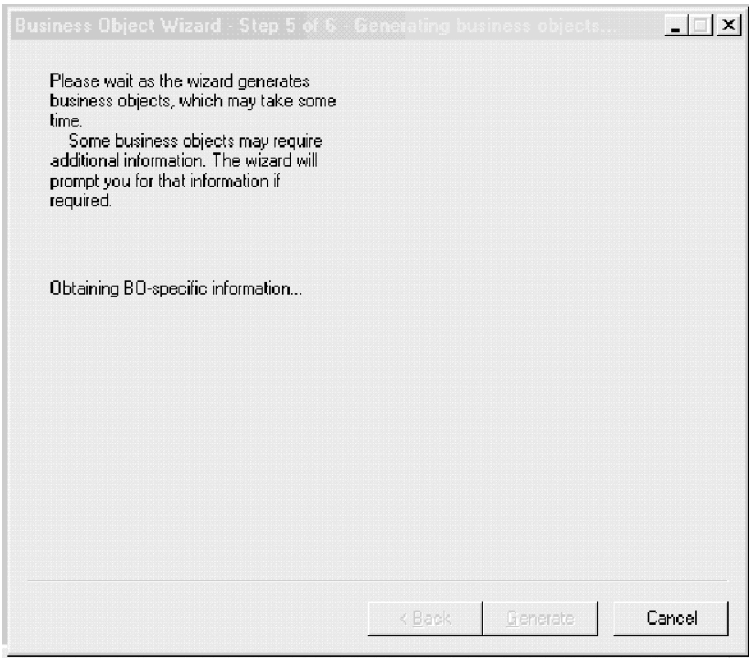


Figure 21. Génération de définitions

Ajout d'informations complémentaires

Si PortalODA requiert des informations complémentaires, Business Object Designer affiche la fenêtre BO Properties, qui vous invite à saisir ces informations. Cela ne se produit qu'en cas de génération d'objets métier. La figure 22, à la page 60 représente cette fenêtre.

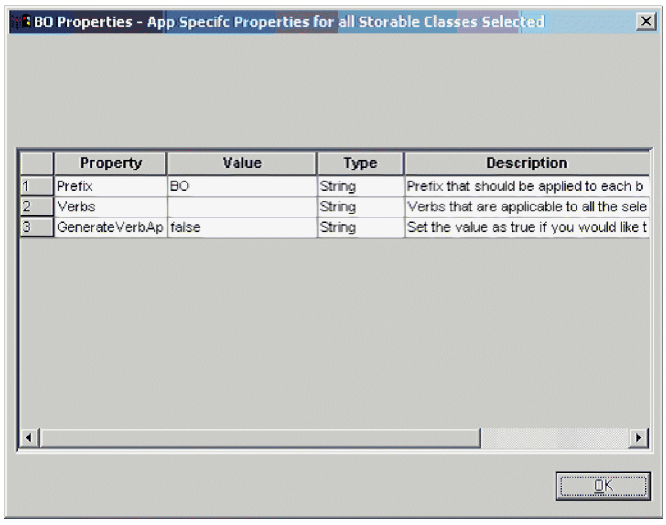


Figure 22. Propriétés des classes stockables spécifiques aux applications

Dans la fenêtre BO Properties, entrez ou modifiez les informations suivantes :

- *Prefix*—Texte ajouté avant le nom de l'objet métier afin de le rendre unique. Si la valeur que vous avez entrée dans la fenêtre Configure Agent pour la propriété *DefaultBOPrefix* vous convient (figure 17, à la page 55), il est inutile de la changer.
- *Verbs*— Cliquez dans la zone *Value* et sélectionnez une ou plusieurs instructions dans le menu en incrustation. Il s'agit d'instructions prises en charge par l'objet métier.

  Remarque : Si une zone dans la boîte de dialogue BO Properties contient plusieurs valeurs, la zone apparaît vide lorsque la boîte de dialogue s'affiche pour la première fois. Cliquez dans la zone pour afficher une liste déroulante contenant ses valeurs.
- *GenerateVerbApp*—Indicateur qui vous permet d'éditer les informations spécifiques aux applications au niveau de l'instruction.

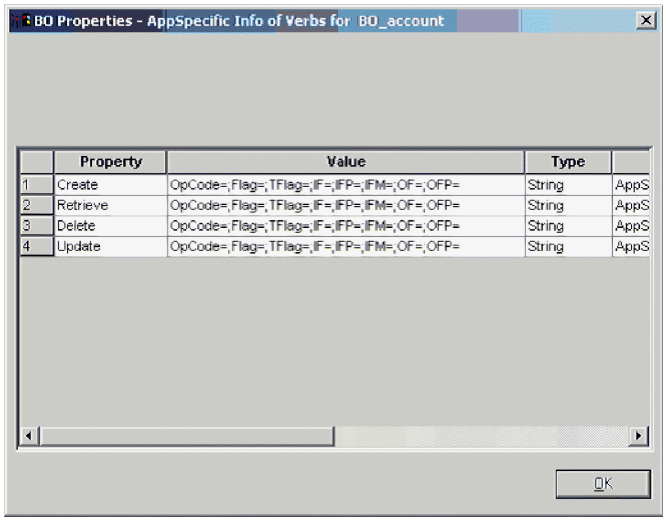


Figure 23. Informations spécifiques aux applications concernant les instructions

Le format des informations spécifiques aux applications au niveau de l'instruction est :

`OpCode=;Flag=;TFlag=;IF=;IFP=;IFM=;OF=OFP=` describes

Le tableau 9 décrit chaque nom cité dans les informations spécifiques aux applications au niveau de l'instruction.

Tableau 9. Informations spécifiques aux applications concernant les instructions

Nom	Description
`Opcode`	Nom du code opération devant être exécuté pour cette instruction
`Flag`	Valeur de l'indicateur à utiliser avec le code opération
`TFlag`	TFlag peut avoir la valeur true ou false, selon que le code opération gère ses propres transactions ou non.
`IF`	IF (Input Flist) est le nom de l'objet métier utilisé pour préparer une flist d'entrée pour le code opération
`IFP`	IFP (Input Flist Parameter) est le nom du paramètre facultatif pouvant être utilisé pour préparer la flist d'entrée.
`IFM`	IFM (Input Flist Mode) est la valeur qui définit le type de conversion de flist effectuée
`OF`	OF (Output Flist) est le paramètre qui régit la façon dont la flist renvoyée lors de l'exécution du code opération doit être convertie en objet métier
`OFP`	OFM (Output Flist Mode) est la valeur qui définit le type de mise à jour des objets métier effectuée à partir de la flist de sortie du code opération

Enregistrement des définitions

Après avoir fourni toutes les informations requises dans la boîte de dialogue BO Properties et cliqué sur OK, Business Object Designer affiche la boîte de dialogue finale de l'assistant. Dans cette fenêtre, vous pouvez enregistrer la définition sur le serveur ou dans un fichier ou encore ouvrir la définition en vue de la modifier dans Business Object Designer. Pour plus d'informations, voir le document *Business Object Development Guide*.

La figure 24, à la page 62 représente cette boîte de dialogue.

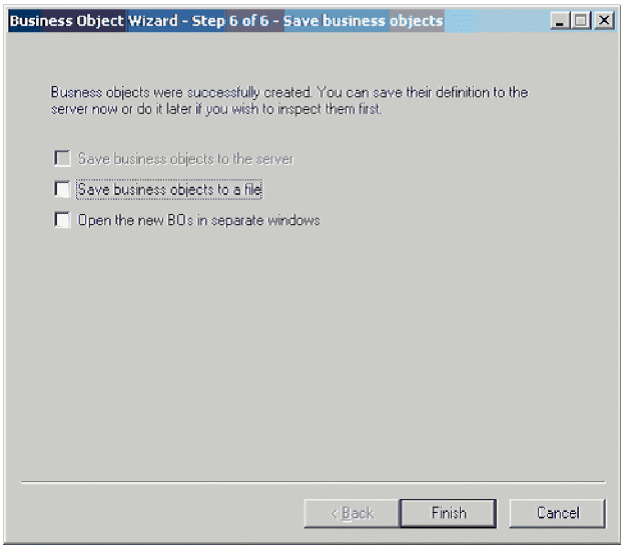


Figure 24. Enregistrement d'une définition d'objet métier

Contenu de la définition générée

La définition d'objet métier générée par PortalODA contient :

- un attribut pour chaque colonne des tables et vues de la base de données spécifiée ;
- les instructions définies dans la fenêtre BO Properties (figure 23, à la page 60) ;
- les informations spécifiques à l'application :
  - sur l'objet métier
  - pour chaque attribut
  - pour chaque instruction

Lorsque vous générez des objets métier à partir des informations fournies par Portal Infranet, les informations spécifiques à l'application générées ne concernent que les attributs simples. L'exception à cette règle s'applique quand l'attribut de conteneur est un lien à plusieurs valeurs. Dans tous les autres cas, l'utilisateur doit entrer les informations spécifiques à l'application, comme décrit au Chapitre 3, «Présentation des objets métier», à la page 27.

Cette section aborde les points suivants :

- «Propriétés des objets métier», à la page 62
- «Propriétés des attributs», à la page 63
- «Instructions», à la page 64

Propriétés des objets métier

PortalODA génère les informations suivantes sur les objets métier :

- nom de l'objet métier ;
- valeur par défaut `1.0.0` de la version ;
- informations spécifiques à l'application.

Au niveau de l'objet métier, les informations spécifiques à l'application contiennent le nom du composant métier Portal Infranet correspondant.

Propriétés des attributs

Cette section décrit les propriétés générées par PortalODA pour chaque attribut.

Important : Toute édition faite par l'utilisateur et décrite dans les sections suivantes fait uniquement référence à la génération d'objets métier, non à leur conversion.

Propriété de nom

PortalODA extrait la valeur du nom de l''attribut de l'attribut correspondant dans le composant métier Portal Infranet.

Propriété Data type

Lorsqu'il définit le type d'un attribut, PortalODA prend le type de données de l'attribut dans le composant métier Portal Infranet et le convertit au type de données correspondant, comme indiqué dans le tableau 10. Cela ne s'applique qu'à la génération d'objets métier, puisque la conversion ne s'applique qu'à des objets métier existants.

Tableau 10. Correspondance des types de données

Application	Système d'intégration WebSphere	Longueur
PIN_FLDT_INT	Integer	
PIN_FLDT_ENUM	Integer	
PIN_FLDT_STR	String	Longueur de l'attribut correspondant dans Portal Infranet
PIN_FLDT_BUF	String	Longueur de l'attribut correspondant dans Portal Infranet
PIN_FLDT_POID	String	Longueur de l'attribut correspondant dans Portal Infranet
PIN_FLDT_TSTAMP	Date	
PIN_FLDT_ARRAY	Object	
PIN_FLDT_SUBSTRUCT	Object	
PIN_FLDT_BINSTR	String	Longueur de l'attribut correspondant dans Portal Infranet
PIN_FLDT_DECIMAL	Float	

Remarque : Si le type de données d''un attribut ne fait pas partie de ceux figurant dans le tableau 10, PortalODA ignore la colonne et affiche un message indiquant que la colonne ne peut être traitée.

Propriété de cardinalité

PortalODA définit la cardinalité de tous les attributs simples sur 1 et celle des attributs du conteneur sur `n`. L'utilisateur doit changer la cardinalité des attributs du conteneur dès que cela s'avère nécessaire.

Propriété MaxLength

PortalODA obtient la longueur de l'attribut à partir de Portal Infranet.

Propriété IsKey

PortalODA ne marque aucun attribut comme étant une zone de clé. Vous devez marquer les zones de clé manuellement, une fois les objets métier générés.

Propriété IsRequired

Si une zone est désignée comme "not null" dans la table ou la vue, PortalODA la marque comme étant un attribut obligatoire. En revanche, PortalODA ne marque pas la zone de clé comme étant obligatoire car l'application Portal Infranet génère ses propres valeurs d'ID lorsqu'elle crée un enregistrement.

Propriété AppSpecificInfo

L'utilisateur doit éditer cette propriété si les attributs de conteneur n'ont pas été générés ou s'assurer qu'elle est correcte si les attributs de conteneur ont été générés.

Instructions

PortalODA génère les instructions spécifiées dans la fenêtre BO Properties (comme illustré par la figure 23, à la page 60). Il crée une propriété AppSpecificInfo pour chaque instruction mais n'indique pas de valeur.

Ajout d'informations à la définition d'objet métier

Dans la mesure où les classes stockables Portal Infranet sont susceptibles de ne pas contenir toutes les informations requises par les objets métier, il peut s'avérer nécessaire d'ajouter des informations à la définition d'objet métier créée par PortalODA, en particulier lors de la génération de nouveaux objets métier.

Pour examiner la définition de l'objet métier ou recharger un définition révisée dans le référentiel, utilisez Business Object Designer.

Remarque : Sinon, si ICS est le courtier d'intégration, vous pouvez utiliser la commande repos_copy pour charger la définition dans le référentiel ; si le courtier d'intégration est WebSphere MQ Integrator Broker, vous pouvez utiliser une commande système pour copier le fichier dans le répertoire du référentiel.

Annexe A. Propriétés standard du connecteur

Cette annexe décrit les propriétés de configuration standard pour le composant de connecteur de WebSphere Business Integration Adapters. Les informations présentées concernent les connecteurs qui s'exécutent sur les courtiers d'intégration suivants :

- WebSphere InterChange Server (ICS)
- WebSphere MQ Integrator, WebSphere MQ Integrator Broker et WebSphere Business Integration Message Broker, appelés collectivement courtiers de messages WebSphere (et WMQI dans Connector Configurator).
- Information Integrator (II)
- WebSphere Application Server (WAS)

Si votre adaptateur prend en charge DB2 Information Integrator, utilisez les options et les propriétés standard DB2 II (voir la colonne Remarques du tableau 11, à la page 67.)

Les propriétés définies pour l'adaptateur dépendent du courtier d'intégration utilisé. Vous sélectionnez ce dernier à l'aide de Connector Configurator. Une fois le courtier choisi, Connector Configurator dresse la liste des propriétés standard à configurer pour l'adaptateur.

Pour plus d'informations sur les propriétés spécifiques au connecteur, voir la section correspondante de ce guide.

Nouvelles propriétés

La propriété standard suivante a été ajoutée à cette édition :

- BOTrace

Présentation des propriétés de connecteur standard

Les connecteurs ont deux types de propriétés de configuration :

- Les propriétés de configuration standard, utilisées par l'architecture
- Les propriétés de configuration spécifiques à une application ou à un connecteur, et utilisées par l'agent

Ces propriétés déterminent l'architecture de l'adaptateur et le comportement d'exécution de l'agent.

Cette section indique comment démarrer Connector Configurator et décrit les caractéristiques communes à toutes les propriétés. Pour plus d'informations sur les propriétés de configuration spécifiques à un connecteur, reportez-vous au guide d'utilisateur de l'adaptateur approprié.

Démarrage de Connector Configurator

Vous pouvez configurer les propriétés du connecteur à partir de Connector Configurator, accessible via System Manager. Pour plus d'informations sur l'utilisation de Connector Configurator, voir les sections associées de ce guide.

Connector Configurator et System Manager s'exécutent uniquement sous Windows. Si vous exécutez le connecteur sous UNIX, vous devez posséder une machine Windows sur laquelle ces outils sont installés.

Pour définir les propriétés d'un connecteur s'exécutant sous UNIX, vous devez démarrer System Manager sur la machine Windows, établir une connexion au courtier d'intégration UNIX et mettre à jour Connector Configurator pour le connecteur.

Présentation des valeurs des propriétés de configuration

Le connecteur utilise l'ordre suivant pour déterminer la valeur d'une propriété :

1. Valeur par défaut
2. Référentiel (valide uniquement si WebSphere InterChange Server (ICS) est le courtier d'intégration)
3. Fichier de configuration locale
4. Ligne de commande

La longueur par défaut d'une propriété est de 255 caractères. La longueur d'un type de propriété STRING n'est pas limitée. La longueur d'un type INTEGER est déterminée par le serveur sur lequel l'adaptateur fonctionne.

Un connecteur obtient ses valeurs de configuration lors du démarrage. Si vous modifiez la valeur d'une ou plusieurs propriétés du connecteur pendant une session d'exécution, la méthode de mise à jour de la propriété détermine la manière dont les modifications prennent effet.

Les caractéristiques de mise à jour d'une propriété, c'est-à-dire à quel moment et de quelle façon la modification des propriétés du connecteur prend effet, dépendent de la nature de la propriété.

Il existe quatre méthodes de mise à jour pour les propriétés standard du connecteur :

- **Dynamique**
  La nouvelle valeur prend effet dès que la modification est enregistrée dans System Manager. Toutefois, si le connecteur est en mode autonome (indépendamment de System Manager), par exemple avec l'un des courtiers de message WebSphere, vous ne pouvez modifier les propriétés que via le fichier de configuration. Dans ce cas, une mise à jour dynamique n'est pas possible.
- **Redémarrage de l'agent (ICS uniquement)**
  La nouvelle valeur ne prend effet qu'une fois que vous avez arrêté et redémarré l'agent du connecteur.
- **Redémarrage du composant**
  La nouvelle valeur ne prend effet qu'après que le connecteur ait été arrêté et redémarré dans System Manager. Vous n'avez pas besoin d'arrêter et de redémarrer l'agent ni le processus du serveur.
- **Redémarrage du système**
  La nouvelle valeur ne prend effet qu'une fois que vous avez arrêté et redémarré l'agent du connecteur et le serveur.

Pour déterminer la manière dont une propriété donnée est mise à jour, reportez-vous à la colonne **Update Method**µ dans la fenêtre Connector Configurator, ou à la colonne Update Method dans le tableau 11, à la page 67.

Une propriété standard peut figurer à trois endroits. Certaines propriétés peuvent figurer à plusieurs emplacements.

- **ReposController**
  La propriété réside dans le contrôleur du connecteur et n'est effective qu'à cet emplacement. Le fait de modifier la valeur du côté de l'agent n'a pas d'effet sur le contrôleur.
- **ReposAgent**
  La propriété réside dans l'agent et n'est effective qu'à cet emplacement. Selon la propriété, une configuration locale peut remplacer cette valeur.
- **LocalConfig**
  La propriété réside dans le fichier de configuration du connecteur et n'agit que par l'intermédiaire de ce fichier. Le contrôleur ne peut pas modifier la valeur de la propriété et n'est pas informé des modifications apportées au fichier de configuration, à moins que le système ne soit redéployé pour remettre à jour le contrôleur explicitement.

Référence rapide des propriétés standard

Le tableau 11 fournit une description rapide des propriétés standard de configuration des connecteurs. Les connecteurs n'exigent pas tous toutes ces propriétés, et les paramètres de propriété peuvent différer d'un courtier d'intégration à l'autre.

Voir la section qui suit le tableau pour une description de chaque propriété.

Remarque : Dans la colonne "Remarques" du tableau 11, la phrase "La valeur de RepositoryDirectory est égale à <REMOTE>" indique que le courtier est InterChange Server. Lorsque le courtier est WMQI ou WAS, le répertoire du référentiel est défini sur <*ProductDir*>\repository

Tableau 11. Récapitulatif des propriétés de configuration standard

Nom de propriété	Valeurs possibles	Valeur par défaut	Méthode de mise à jour	Remarques
AdapterHelpName	Un des sous-répertoires valides de <*ProductDir*>\bin\Data \App\Help\ contenant un répertoire <*RegionalSetting*> valide	Nom du modèle, si valide, ou zone vide	Redémarrage du composant	Paramètres régionaux pris en charge. Incluent `chs_chn`, `cht_twn`, `deu_deu`, `esn_esp`, `fra_fra`, `ita_ita`, `jpn_jpn`, `kor_kor`, `ptb_bra`, et `enu_usa` (par défaut).
AdminInQueue	Nom de file d'attente JMS valide	<*CONNECTORNAME*> `/ADMININQUEUE`	Redémarrage du composant	Cette propriété n'est valide que si la valeur de DeliveryTransport est `JMS`
AdminOutQueue	Nom de file d'attente JMS valide	<*CONNECTORNAME*> `/ADMINOUTQUEUE`	Redémarrage du composant	Cette propriété n'est valide que si la valeur de DeliveryTransport est `JMS`
AgentConnections	`1` à `4`	`1`	Redémarrage du composant	Cette propriété n'est valide que si la valeur de DeliveryTransport est `MQ` ou `IDL`, la valeur de Repository Directory est <REMOTE> et la valeur de BrokerType est ICS.

Tableau 11. Récapitulatif des propriétés de configuration standard (suite)

Nom de propriété	Valeurs possibles	Valeur par défaut	Méthode de mise à jour	Remarques
AgentTraceLevel	0 à 5	0	Dynamique si le courtier est ICS ; sinon Redémarrage du composant	
ApplicationName	Nom d'application	Valeur précisée pour le nom de l'application du connecteur	Redémarrage du composant	
BiDi.Application	Toute combinaison valide de ces attributs bidirectionnels : 1ère lettre : I,V 2e lettre : L,R 3e lettre : Y, N 4e lettre : S, N 5e lettre : H, C, N	ILYNN (cinq lettres)	Redémarrage du composant	Cette propriété n'est valide que si la valeur de BiDi.Transforma tion est true
BiDi.Broker	Toute combinaison valide de ces attributs bidirectionnels : 1ère lettre : I,V 2e lettre : L,R 3e lettre : Y, N 4e lettre : S, N 5e lettre : H, C, N	ILYNN (cinq lettres)	Redémarrage du composant	Cette propriété n'est valide que si la valeur de BiDi.Transformation est true. Si la valeur de BrokerType est ICS, la propriété est en lecture seule.
BiDi.Metadata	Toute combinaison valide de ces attributs bidirectionnels : 1ère lettre : I,V 2e lettre : L,R 3e lettre : Y, N 4e lettre : S, N 5e lettre : H, C, N	ILYNN (cinq lettres)	Redémarrage du composant	Cette propriété n'est valide que si la valeur de BiDi.Transformation est true.
BiDi.Transformation	true ou false	false	Redémarrage du composant	Cette propriété n'est valide que si la valeur de BrokerType n'est pas WAS.
BOTrace	none ou keys ou full	none	Redémarrage du composant	Cette propriété n'est valide que si la valeur de AgentTraceLevel est inférieure à 5.
BrokerType	ICS , WMQI, WAS	ICS	Redémarrage du composant	
CharacterEncoding	Tout code pris en charge. La liste indique le sous-ensemble : ascii7, ascii8, SJIS, Cp949, GBK, Big5, Cp297, Cp273, Cp280, Cp284, Cp037, Cp437	ascii7	Redémarrage du composant	Cette propriété n'est valide que pour les connecteurs C++.
CommonEventInfrastruc ture	true ou false	false	Redémarrage du composant	
CommonEventInfrastruc tureURL	Une chaîne URL, par exemple, corbaloc:iiop: host:2809.	Aucune valeur par défaut.	Redémarrage du composant	Cette propriété n'est valide que si la valeur de CommonEvent Infrastructure est true.

Tableau 11. Récapitulatif des propriétés de configuration standard (suite)

Nom de propriété	Valeurs possibles	Valeur par défaut	Méthode de mise à jour	Remarques
ConcurrentEventTriggeredFlows	1 à 32,767	1	Redémarrage du composant	Cette propriété n'est valide que si la valeur de RepositoryDirectory est <REMOTE> et la valeur de BrokerType est ICS.
ContainerManagedEvents	Vide ou JMS	Vide	Redémarrage du composant	Cette propriété n'est valide que si la valeur de Delivery Transport est JMS
ControllerEventSequencing	true ou false	true	Dynamique	Cette propriété n'est valide que si la valeur de Repository Directory est égale à <REMOTE> et la valeur de BrokerType est ICS.
ControllerStoreAndForwardMode	true ou false	true	Dynamique	Cette propriété n'est valide que si la valeur de Repository Directory est égale à <REMOTE> et la valeur de BrokerType est ICS.
ControllerTraceLevel	0 à 5	0	Dynamique	Cette propriété n'est valide que si la valeur de RepositoryDirectory est <REMOTE> et la valeur de BrokerType est ICS.
DeliveryQueue	Tout nom valide de file d'attente JMS valide	*<CONNECTORNAME>* /DELIVERYQUEUE	Redémarrage du composant	Cette propriété n'est valide que si la valeur de Delivery Transport est JMS
DeliveryTransport	MQ, IDL ou JMS	IDLlorsque la valeur de RepositoryDirectory est <REMOTE>, sinon JMS	Redémarrage du composant	Si la valeur de RepositoryDirectory n'est pas <REMOTE>, la seule valeur valide pour cette propriété est JMS
DuplicateEventElimination	true ou false	false	Redémarrage du composant	Cette propriété n'est valide que si la valeur de DeliveryTransport est JMS
EnableOidForFlowMonitoring	true ou false	false	Redémarrage du composant	Cette propriété n'est valide que si la valeur de BrokerType est ICS.
FaultQueue	Tout nom de file d'attente valide.	*<CONNECTORNAME>* /FAULTQUEUE	Redémarrage du composant	Cette propriété n'est valide que si la valeur de DeliveryTransport est JMS
jms.FactoryClassName	CxCommon.Messaging.jms.IBMMQSeriesFactory, CxCommon.Messaging.jms.SonicMQFactory, ou n'importe quel nom de classe Java	CxCommon.Messaging.jms.IBMMQSeriesFactory	Redémarrage du composant	Cette propriété n'est valide que si la valeur de DeliveryTransport est JMS
jms.ListenerConcurrency	1 à 32767	1	Redémarrage du composant	Cette propriété n'est valide que si la valeur de jms.TransportOptimized est true.

Tableau 11. Récapitulatif des propriétés de configuration standard (suite)

Nom de propriété	Valeurs possibles	Valeur par défaut	Méthode de mise à jour	Remarques
jms.MessageBrokerName	Si la valeur de jms.FactoryClassName est IBM, utilisez crossworlds.queue.manager.	crossworlds.queue.manager	Redémarrage du composant	Cette propriété n'est valide que si la valeur de DeliveryTransport est JMS
jms.NumConcurrent Requests	Entier positif	10	Redémarrage du composant	Cette propriété n'est valide que si la valeur de DeliveryTransport est JMS
jms.Password	Tout mot de passe valide		Redémarrage du composant	Cette propriété n'est valide que si la valeur de DeliveryTransport est JMS
jms.TransportOptimized	true ou false	false	Redémarrage du composant	Cette propriété n'est valide que si la valeur de DeliveryTransport est JMS et la valeur de BrokerType est ICS.
jms.UserName	Tout nom valide		Redémarrage du composant	Cette propriété n'est valide que si la valeur de Delivery Transport est JMS.
JvmMaxHeapSize	Taille de segment en mégaoctets	128 Mo	Redémarrage du composant	Cette propriété n'est valide que si la valeur de Repository Directory est égale à <REMOTE> et la valeur de BrokerType est ICS.
JvmMaxNativeStackSize	Taille de la pile en kilo-octets	128 Ko	Redémarrage du composant	Cette propriété n'est valide que si la valeur de Repository Directory est égale à <REMOTE> et la valeur de BrokerType est ICS.
JvmMinHeapSize	Taille de segment en mégaoctets	1 Mo	Redémarrage du composant	Cette propriété n'est valide que si la valeur de Repository Directory est égale à <REMOTE> et la valeur de BrokerType est ICS.
ListenerConcurrency	1 à 100	1	Redémarrage du composant	Cette propriété n'est valide que si la valeur de DeliveryTransport est MQ.
Locale	Il s'agit d'un sous-ensemble des paramètres régionaux pris en charge : en_US, ja_JP, ko_KR, zh_CN, zh_TW, fr_FR, de_DE, it_IT, es_ES, pt_BR	en_US	Redémarrage du composant	
LogAtInterchangeEnd	true ou false	false	Redémarrage du composant	Cette propriété n'est valide que si la valeur de Repository Directory est égale à <REMOTE> et la valeur de BrokerType est ICS.

Tableau 11. Récapitulatif des propriétés de configuration standard (suite)

Nom de propriété	Valeurs possibles	Valeur par défaut	Méthode de mise à jour	Remarques
MaxEventCapacity	1 à 2147483647	2147483647	Dynamique	Cette propriété n'est valide que si la valeur de Repository Directory est égale à <REMOTE> et la valeur de BrokerType est ICS.
MessageFileName	Nom de fichier valide	InterchangeSystem.txt	Redémarrage du composant	
MonitorQueue	Tout nom de file d'attente valide	*<CONNECTORNAME>* /MONITORQUEUE	Redémarrage du composant	Cette propriété n'est valide que si la valeur de DuplicateEventElimination est true et si ContainerManagedEvents n'a pas de valeur.
OADAutoRestartAgent	true ou false	false	Dynamique	Cette propriété n'est valide que si la valeur de Repository Directory est égale à <REMOTE> et la valeur de BrokerType est ICS.
OADMaxNumRetry	Un nombre entier positif	1000	Dynamique	Cette propriété n'est valide que si la valeur de Repository Directory est égale à <REMOTE> et la valeur de BrokerType est ICS.
OADRetryTimeInterval	Un nombre entier positif en minutes	10	Dynamique	Cette propriété n'est valide que si la valeur de Repository Directory est égale à <REMOTE> et la valeur de BrokerType est ICS.
PollEndTime	HH = 0 à 23 MM = 0 à 59	HH:MM	Redémarrage du composant	
PollFrequency	Un nombre entier positif (en millisecondes)	10000	Dynamique si le courtier est ICS ; sinon Redémarrage du composant	
PollQuantity	1 à 500	1	Redémarrage de l'agent	Cette propriété n'est valide que si la valeur de ContainerManagedEvents est JMS.
PollStartTime	HH = 0 à 23 MM = 0 à 59	HH:MM	Redémarrage du composant	
RepositoryDirectory	<REMOTE> si le courtier est ICS ; sinon tout répertoire local valide.	Pour ICS, la valeur est définie sur <REMOTE> Pour WMQI et WAS, la valeur est *<ProductDir* \repository	Redémarrage de l'agent	
RequestQueue	Nom de file d'attente JMS valide	*<CONNECTORNAME>* /REQUESTQUEUE	Redémarrage du composant	Cette propriété n'est valide que si la valeur de DeliveryTransport est JMS

Tableau 11. Récapitulatif des propriétés de configuration standard (suite)

Nom de propriété	Valeurs possibles	Valeur par défaut	Méthode de mise à jour	Remarques
ResponseQueue	Nom de file d'attente JMS valide	*<CONNECTORNAME>* /RESPONSEQUEUE	Redémarrage du composant	Cette propriété n'est valide que si la valeur de DeliveryTransport est JMS.
RestartRetryCount	0 à 99	7	Dynamique si ICS ; sinon Redémarrage du composant	
RestartRetryInterval	Une valeur en minutes de 1 à 2147483647	1	Dynamique si ICS ; sinon Redémarrage du composant	
ResultsSetEnabled	true ou false	false	Redémarrage du composant	Utilisé uniquement par les connecteurs qui prennent en charge DB2II. Cette propriété n'est valide que si la valeur de DeliveryTransport est JMS, et la valeur de BrokerType est WMQI.
ResultsSetSize	Entier positif	0 (indique que la taille de l'ensemble de résultats est illimitée)	Redémarrage du composant	Utilisé uniquement par les connecteurs qui prennent en charge DB2II. Cette propriété n'est valide que si la valeur de ResultsSetEnabled est true.
RHF2MessageDomain	mrm ou xml	mrm	Redémarrage du composant	Cette propriété n'est valide que si la valeur de DeliveryTransport est JMS et la valeur de WireFormat est CwXML.
SourceQueue	Tout nom de file d'attente WebSphere MQ	*<CONNECTORNAME>* /SOURCEQUEUE	Redémarrage de l'agent	Cette propriété n'est valide que si la valeur de ContainerManagedEvents est JMS.
SynchronousRequest Queue	Tout nom de file d'attente valide.	*<CONNECTORNAME>* /SYNCHRONOUSREQUEST QUEUE	Redémarrage du composant	Cette propriété n'est valide que si la valeur de DeliveryTransport est JMS.
SynchronousRequest Timeout	0 à n'importe quelle valeur (millisecondes)	0	Redémarrage du composant	Cette propriété n'est valide que si la valeur de DeliveryTransport est JMS.
SynchronousResponse Queue	Tout nom de file d'attente valide	*<CONNECTORNAME>* /SYNCHRONOUSRESPONSE QUEUE	Redémarrage du composant	Cette propriété n'est valide que si la valeur de DeliveryTransport est JMS.
TivoliMonitorTransaction Performance	true ou false	false	Redémarrage du composant	

Tableau 11. Récapitulatif des propriétés de configuration standard (suite)

Nom de propriété	Valeurs possibles	Valeur par défaut	Méthode de mise à jour	Remarques
WireFormat	CwXML ou CwBO	CwXML	Redémarrage de l'agent	La valeur de cette propriété doit être CwXML si la valeur de RepositoryDirectory n'est pas définie sur <REMOTE>. La valeur doit être CwBO si la valeur de RepositoryDirectory est définie sur <REMOTE>.
WsifSynchronousRequest Timeout	0 à n'importe quelle valeur (millisecondes)	0	Redémarrage du composant	Cette propriété n'est valide que si la valeur de BrokerType est WAS.
XMLNameSpaceFormat	short ou long ou no	short	Redémarrage de l'agent	Cette propriété n'est valide que si la valeur de BrokerType est WMQI ou WAS

Propriétés standard

Cette section décrit les propriétés standard de configuration du connecteur.

AdapterHelpName

La propriété AdapterHelpName est le nom d'un répertoire contenant des fichiers d'aide étendue spécifiques au connecteur. Le répertoire doit figurer dans <*ProductDir*>\bin\Data\App\Help et contenir au moins le répertoire de langue enu_usa. Il peut contenir d'autres répertoires selon les paramètres régionaux.

La valeur par défaut est le nom du modèle s'il est valide, ou elle est vide.

AdminInQueue

La propriété AdminInQueue précise la file d'attente utilisée par le courtier d'intégration pour envoyer des messages administratifs au connecteur.

La valeur par défaut est <*CONNECTORNAME*>/ADMININQUEUE

AdminOutQueue

La propriété AdminOutQueue précise la file d'attente utilisée par le connecteur pour envoyer des messages administratifs au courtier d'intégration.

La valeur par défaut est <*CONNECTORNAME*>/ADMINOUTQUEUE

AgentConnections

La propriété AgentConnections contrôle le nombre de connexions ORB (Object Request Broker) ouvertes à l'initialisation de ORB.

Elle n'est valide que si la valeur de RepositoryDirectory est définie sur <REMOTE> et si la valeur de la propriété DeliveryTransport est MQ ou IDL.

La valeur par défaut de cette propriété est 1.

AgentTraceLevel

La propriété AgentTraceLevel définit le niveau des messages de trace pour le composant spécifique à l'application. Le connecteur fournit tous les messages de trace applicables au niveau de trace défini et à un niveau inférieur.

La valeur par défaut est `0`.

ApplicationName

La propriété ApplicationName identifie de façon unique le nom de l'application du connecteur. Ce nom permet à l'administrateur système de surveiller l'environnement d'intégration. Vous devez attribuer une valeur à cette propriété avant d'exécuter le connecteur.

La valeur par défaut est le nom du connecteur.

BiDi.Application

La propriété BiDi.Application précise le format bidirectionnel des données provenant d'une application externe et entrant dans l'adaptateur, sous la forme d'un objet métier pris en charge par cet adaptateur. La propriété définit les attributs bidirectionnels des données de l'application. Ces attributs sont les suivants :

- Type de texte : implicite ou visuel (`I` ou `V`)
- Direction du texte : de gauche à droite ou de droite à gauche (`L` ou `R`)
- Permutation symétrique : activée ou désactivée (`Y` ou `N`)
- Mise en forme (arabe): activée ou désactivée (`S` ou `N`)
- Mise en forme numérique (arabe) : hindi, contextuel, ou nominal (`H`, `C` ou `N`)

Cette propriété n'est valide que si la valeur de la propriété BiDi.Transformation est définie sur true.

La valeur par défaut est `ILYNN` (implicite, gauche à droite, activé, désactivé, nominal).

BiDi.Broker

La propriété BiDi.Broker précise le format de script bidirectionnel pour les données envoyées depuis l'adaptateur au courtier d'intégration sous la forme d'un objet métier pris en charge. Elle définit les attributs bidirectionnels des données, indiqués sous `BiDi.Application` ci-dessous.

Cette propriété n'est valide que si la valeur de la propriété BiDi.Transformation est définie sur true. Si la propriété BrokerType est ICS, sa valeur est en lecture seule.

La valeur par défaut est `ILYNN` (implicite, gauche à droite, activé, désactivé, nominal).

BiDi.Metadata

La propriété BiDi.Metadata définit le format bidirectionnel ou les attributs des métadonnées qui sont utilisées par le connecteur pour établir et maintenir un lien vers l'application externe. Les paramètres de l'attribut sont spécifiques à chaque adaptateur qui utiliser des capacités bidirectionnelles. Si votre adaptateur prend en charge le traitement bidirectionnel, voir la section relative aux propriétés spécifiques à l'adaptateur pour plus d'informations.

Cette propriété n'est valide que si la valeur de la propriété BiDi.Transformation est définie sur true.

La valeur par défaut est `ILYNN` (implicite, gauche à droite, activé, désactivé, nominal).

BiDi.Transformation

La propriété BiDi.Transformation détermine si le système procède ou non à une transformation bidirectionnelle lors de l'exécution.

Si la valeur de la propriété est définie sur `true`, les propriétés BiDi.Application, BiDi.Broker et BiDi.Metadata sont disponibles. Si la valeur de la propriété est définie sur `false`, elles sont cachées.

La valeur par défaut est `false`.

BOTrace

La propriété BOTrace indique si les messages de trace d'objet métier sont activés ou non lors de l'exécution.

Remarque : Ceci ne s'applique que si la propriété AgentTraceLevel est inférieure à 5.

Lorsque le niveau de trace est inférieur à 5, vous pouvez utiliser ces paramètres de ligne de commande pour réinitialiser la valeur de BOTrace.

- Entrez `-xBOTrace=Full` pour afficher tous les attributs d'objets métier.
- Entrez `-xBOTrace=Keys` pour n'afficher que les clés d'objets métier.
- Entrez `-xBOTrace=None` pour désactiver l'affichage des attributs d'objets métier.

La valeur par défaut est `false`.

BrokerType

La propriété BrokerType identifie le type de courtier d'intégration que vous utilisez. Les valeurs possibles sont `ICS`, `WMQI (pour WMQI, WMQIB ou WBIMB)` ou `WAS`.

CharacterEncoding

La propriété CharacterEncoding indique le jeu de codes de caractères utilisé pour mettre en correspondance un caractère (une lettre de l'alphabet, un chiffre ou un signe de ponctuation) et une valeur numérique.

Remarque : Les connecteurs Java n'utilisent pas cette propriété. Les connecteurs C++ utilisent la valeur `ascii7` pour cette propriété.

Par défaut, n'est affiché qu'un sous-ensemble des codages de caractères pris en charge. Pour ajouter d'autres valeurs prises en charge à la liste, vous devez modifier manuellement le fichier \Data\Std\stdConnProps.xml dans le répertoire produit (<*ProductDir*>). Pour plus d'informations, voir l'annexe Connector Configurator de ce guide.

CommonEventInfrastructure

L'infrastructure Common Event Infrastructure (CEI) est une fonction simple de gestion des événements chargée de traiter les événements générés. La propriété CommonEventInfrastructure indique si le CEI doit être appelé lors de l'exécution.

La valeur par défaut est `false`.

CommonEventInfrastructureContextURL

CommonEventInfrastructureContextURL est utilisé pour accéder au serveur WAS qui exécute l'application du serveur CEI (Common Event Infrastructure). Cette propriété précise l'URL à utiliser.

Cette propriété n'est valide que si la valeur de CommonEventInfrastructure est définie sur `true`.

La valeur par défaut est une zone vide.

ConcurrentEventTriggeredFlows

La propriété ConcurrentEventTriggeredFlows détermine le nombre d'objets métier pouvant être traités simultanément par le connecteur pour la transmission des événements. Vous définissez la valeur de cet attribut sur le nombre d'objets métiers qui sont simultanément mappés et livrés. Par exemple, si vous définissez la valeur de cette propriété sur `5`, cinq objets métier sont traités simultanément.

La définition de cette propriété sur une valeur supérieure à 1 permet au connecteur d'une application source de mapper plusieurs objets métier d'événement en même temps et de les transmettre simultanément à plusieurs instances de collaboration. Cela augmente la rapidité de transmission des objets métier au courtier d'intégration, en particulier si les objets métier utilisent des mappes complexes. L'augmentation du taux d'arrivée des objets métier aux instances de collaboration peut améliorer les performances générales du système.

Pour implémenter le traitement simultané d'un flux entier (d'une application source vers une application cible), vous devez configurer les propriétés suivantes :

- La collaboration doit être configurée de façon à utiliser plusieurs unités d'exécution simultanées, en indiquant pour la propriété `Maximum number of concurrent events` une valeur suffisamment élevée.
- Le composant spécifique à l'application de destination doit être configuré pour traiter les requêtes simultanément. C'est à dire qu'il doit avoir plusieurs unités d'exécution ou être capable d'utiliser le parallélisme de l'agent du connecteur et être configuré pour plusieurs processus. Attribuez une valeur supérieure à 1 à la propriété de configuration Parallel Process Degree.

La propriété ConcurrentEventTriggeredFlows property n'a aucun effet sur l'interrogation du connecteur, laquelle n'a qu'une seule unité d'exécution et est exécutée en série.

La propriété n'est valide que si la valeur de la propriété RepositoryDirectory est définie sur <REMOTE>.

La valeur par défaut est 1.

ContainerManagedEvents

La propriété ContainerManagedEvents permet à un connecteur activé par JMS et utilisant un magasin d'événements JMS d'effectuer une transmission garantie d'événement, dans laquelle un événement est retiré de la file d'attente source et placé sur la file d'attente cible en tant qu'une transaction JMS.

Lorsque cette propriété est définie sur JMS, les propriétés suivantes doivent également être définies pour activer la transmission garantie d'événement :

- PollQuantity = 1 à 500
- SourceQueue = /SOURCEQUEUE

Vous devez aussi configurer un gestionnaire de données avec les propriétés MimeType et DHClass (classe de gestionnaire de données). Vous pouvez également ajouter DataHandlerConfigMOName (le nom de méta-objet facultatif). Pour définir ces valeurs, utilisez l'onglet **Data Handler** dans Connector Configurator.

Bien que ces propriétés soient spécifiques à l'adaptateur, voici quelques exemples de valeurs :

- MimeType = text\xml
- DHClass = com.crossworlds.DataHandlers.text.xml
- DataHandlerConfigMOName = MO_DataHandler_Default

Les zones qui correspondent à ces valeurs dans l'onglet **Data Handler** ne s'affichent que si vous avez défini la propriété ContainerManagedEvents sur la valeur JMS.

Remarque : Lorsque ContainerManagedEvents a la valeur JMS, le connecteur n'appelle pas sa méthode pollForEvents(), ce qui en désactive la fonctionnalité.

La propriété ContainerManagedEvents n'est valide que si la valeur de la propriété DeliveryTransport est définie sur JMS.

Il n'y a pas de valeur par défaut.

ControllerEventSequencing

La propriété ControllerEventSequencing autorise le séquençage des événements dans le contrôleur du connecteur.

Cette propriété n'est valide que si la valeur de la propriété RepositoryDirectory est définie sur <REMOTE> (BrokerType égal à ICS).

La valeur par défaut est true.

ControllerStoreAndForwardMode

La propriété ControllerStoreAndForwardMode définit le comportement du contrôleur du connecteur après avoir détecté l'indisponibilité du composant spécifique à l'application cible.

Si cette propriété a la valeur true et que le composant spécifique à l'application cible n'est pas disponible lorsqu'un événement atteint l'ICS, le contrôleur du connecteur empêche la requête d'accéder au composant spécifique à l'application. Lorsque le composant spécifique à l'application redevient opérationnel, le contrôleur lui envoie la requête.

Toutefois, si le composant d'application cible devient indisponible après que le contrôleur du connecteur lui a envoyé la requête d'appel de service, celle-ci échoue.

Si cette propriété a la valeur `false`, le contrôleur du connecteur met toutes les requêtes d'appels de service en échec dès qu'il détecte l'indisponibilité du composant spécifique à l'application.

Cette propriété n'est valide que si la valeur de la propriété RepositoryDirectory est définie sur <REMOTE> (la valeur de la propriété BrokerType est ICS).

La valeur par défaut est `true`.

ControllerTraceLevel

La propriété ControllerTraceLevel définit le niveau des messages de trace pour le contrôleur de connecteur.

La propriété n'est valide que si la valeur de la propriété RepositoryDirectory est définie sur <REMOTE>.

La valeur par défaut est `0`.

DeliveryQueue

La propriété DeliveryQueue définit la file d'attente utilisée par le connecteur pour envoyer des objets métier au courtier d'intégration.

Cette propriété n'est valide que si la valeur de la propriété DeliveryTransport est définie sur `JMS`.

La valeur par défaut est *<CONNECTORNAME>*`/DELIVERYQUEUE`.

DeliveryTransport

La propriété DeliveryTransport spécifie le mécanisme de transfert pour la transmission des événements. Les valeurs possibles sont `MQ` pour WebSphere MQ, `IDL` pour CORBA IIOP et `JMS` pour Java Messaging Service.

- Si la valeur de la propriété RepositoryDirectory est définie sur <REMOTE>, la valeur de la propriété DeliveryTransport peut être `MQ`, `IDL` ou `JMS`, et la valeur par défaut est `IDL`.
- Si la valeur de la propriété RepositoryDirectory est un répertoire local, l'unique valeur possible est `JMS`.

Si la valeur de la propriété RepositoryDirectory est `MQ` ou `IDL`, le connecteur envoie des requêtes d'appel de service et des messages administratifs par CORBA IIOP.

Si la valeur de la propriété DeliveryTransport est `MQ`, vous pouvez définir le paramètre de ligne de commande `WhenServerAbsent` dans le script de démarrage de l'adaptateur, de façon à indiquer si l'adaptateur doit se mettre en pause ou se fermer lorsque InterChange Server est arrêté.

- Entrez `WhenServerAbsent=pause` pour mettre l'adaptateur en pause lorsque ICS n'est pas disponible.
- Entrez `WhenServerAbsent=shutdown` pour arrêter l'adaptateur lorsque ICS n'est pas disponible.

WebSphere MQ et IDL

Utilisez WebSphere MQ plutôt que IDL pour le transfert d'événement, sauf si vous ne devez avoir qu'un seul produit. WebSphere MQ présente les avantages suivants par rapport à IDL :

- Communication asynchrone :
  WebSphere MQ permet au composant spécifique à l'application d'interroger et de stocker de manière permanente les événements, même lorsque le serveur n'est pas disponible.
- Performance côté serveur :
  WebSphere MQ offre plus de rapidité du côté du serveur. En mode optimisé, WebSphere MQ ne stocke que le pointeur désignant un événement dans la base de données du référentiel, tandis que l'événement correspondant reste dans la file d'attente de WebSphere MQ. Ceci empêche d'écrire des événements potentiellement volumineux dans la base de données du référentiel.
- Performance côté agent :
  WebSphere MQ offre plus de rapidité du côté du composant spécifique à l'application. Avec WebSphere MQ, l'unité d'exécution d'interrogation du connecteur sélectionne un événement, le place dans la file d'attente du connecteur, puis sélectionne l'événement suivant. Cette méthode est plus rapide que celle d'IDL, dans laquelle l'unité d'exécution d'interrogation du connecteur doit sélectionner un événement, accéder au réseau dans le processus du serveur, stocker l'événement de manière permanente dans la base de données du référentiel, puis sélectionner l'événement suivant.

JMS

Le mécanisme de transfert JMS active la communication entre le connecteur et l'architecture du connecteur client à l'aide de Java Messaging Service (JMS).

Si vous sélectionnez JMS en tant que transfert, d'autres propriétés JMS supplémentaires telles que `jms.MessageBrokerName`, `jms.FactoryClassName`, `jms.Password` et `jms.UserName` apparaissent dans Connector Configurator. Les propriétés `jms.MessageBrokerName` et `jms.FactoryClassName` sont obligatoires pour ce transfert.

Il peut y avoir une limitation de mémoire si vous utilisez le mécanisme de transfert JMS pour un connecteur dans l'environnement suivant :

- AIX 5.0
- WebSphere MQ 5.3.0.1
- ICS est le courtier d'intégration

Dans cet environnement, vous rencontrerez peut être des difficultés pour démarrer simultanément le contrôleur du connecteur (du côté du serveur) et le connecteur (du côté du client), en raison de l'utilisation de la mémoire dans le client WebSphere MQ. Si votre installation utilise une taille de segment de processus inférieure à 768 Mo, définissez la variable et la propriété suivantes :

- Définissez la variable d'environnement LDR_CNTRL dans le script CWSharedEnv.sh.

  Ce script se trouve dans le répertoire \bin sous le répertoire produit (<*ProductDir*>). A l'aide d'un éditeur de texte, ajoutez la ligne suivante à la première ligne du script CWSharedEnv.sh :

  ```
  export LDR_CNTRL=MAXDATA=0x30000000
  ```

Cette ligne de commande restreint l'utilisation du segment de mémoire à un maximum de 768 Mo (3 segments * 256 Mo). Si la mémoire du processus dépasse cette limite, un échange de pages peut se produire, ce qui peut affecter les performances de votre système.

- Définissez la valeur de la propriété IPCCBaseAddress sur 11 ou 12. Pour plus d'informations sur cette propriété, voir le document *System Installation Guide for UNIX*.

DuplicateEventElimination

Lorsque la valeur de cette propriété est `true`, un connecteur activé pour JMS peut vérifier que des doublons ne sont pas transmis à la file d'attente de transmission. Pour utiliser cette fonction, le connecteur doit recevoir pendant son développement un identificateur d'événement unique défini en tant qu'attribut ObjectEventId de l'objet métier dans le code spécifique à l'application.

Remarque : Lorsque la valeur de cette propriété est `true`, la propriété MonitorQueue doit être activée pour garantir la livraison de l'événement.

La valeur par défaut est `false`.

EnableOidForFlowMonitoring

Lorsque la valeur de cette propriété est `true`, l'exécution de l'adaptateur marque le ObjectEventID entrant en tant que clé étrangère pour la surveillance du flot.

La propriété n'est valide que si la propriété BrokerType est définie sur ICS.

La valeur par défaut est `false`.

FaultQueue

Si le connecteur rencontre une erreur lors du traitement d'un message, il transmet ce message à la file d'attente indiquée dans la propriété FaultQueue (accompagné d'un indicateur de statut et d'une description de l'incident).

La valeur par défaut est *<CONNECTORNAME>*`/FAULTQUEUE`.

jms.FactoryClassName

La propriété jms.FactoryClassName indique le nom de classe à instancier pour un fournisseur JMS. Cette propriété doit être définie si la valeur de la propriété DeliveryTransport property est `JMS`.

La valeur par défaut est `CxCommon.Messaging.jms.IBMMQSeriesFactory`.

jms.ListenerConcurrency

La propriété jms.ListenerConcurrency indique le nombre de programmes d'écoute simultanés pour le contrôleur JMS. Elle précise le nombre d'unités d'exécution qui extraient et traitent les messages simultanément, dans un contrôleur.

Cette propriété n'est valide que si la valeur de la propriété jms.OptimizedTransport est `true`.

La valeur par défaut est 1.

jms.MessageBrokerName

jms.MessageBrokerName précise le nom de courtier à utiliser pour le fournisseur JMS. Vous devez définir cette propriété de connecteur si vous précisez JMS en tant que mécanisme de transfert (dans la propriété DeliveryTransport).

Lorsque vous vous connectez à un courtier de messages éloigné, cette propriété exige les valeurs suivantes :
QueueMgrName:*Channel*:*HostName*:*PortNumber*
où :
QueueMgrName est le nom du gestionnaire de files d'attente.
Channel est le canal utilisé par le client.
HostName est le nom de la machine sur laquelle doit résider le gestionnaire de files d'attente.
PortNumber est le numéro de port sur lequel écoute le gestionnaire de files d'attente.

Par exemple :
```
jms.MessageBrokerName = WBIMB.Queue.Manager:CHANNEL1:RemoteMachine:1456
```

La valeur par défaut est crossworlds.queue.manager. Utilisez la valeur par défaut lorsque vous vous connectez à un courtier de messages local.

jms.NumConcurrentRequests

La propriété jms.NumConcurrentRequests indique le nombre maximal de requêtes d'appel de service pouvant être envoyées simultanément à un connecteur. Lorsque ce nombre maximal est atteint, les nouveaux appels de service sont bloqués et mis en attente de traitement.

La valeur par défaut est 10.

jms.Password

La propriété jms.Password indique le mot de passe défini pour le fournisseur JMS. Cette valeur est facultative.

Il n'y a pas de valeur par défaut.

jms.TransportOptimized

La propriété jms.TransportOptimized détermine si l'opération en cours (WIP) est optimisée. Pour l'optimiser, vous devez disposer d'un fournisseur WebSphere MQ. Pour que le WIP optimisé fonctionne, le fournisseur de messagerie doit pouvoir :

1. Lire un message sans le retirer de la file d'attente
2. Supprimer un message ayant un ID donné sans transférer le message entier dans l'espace mémoire du réceptionnaire
3. Lire un message en utilisant un ID donné (nécessaire pour la récupération)
4. Déterminer à quel moment apparaissent les événements qui n'ont pas été lus.

Les API JMS ne peuvent par être utilisées pour le WIP optimisé car elles ne remplissent pas les conditions 2 et 4 ci-dessus. Les API MQ Java remplissent les quatre conditions et sont donc requises pour le WIP optimisé.

Cette propriété n'est valide que si la valeur de DeliveryTransport est JMS et la valeur de BrokerType est ICS.

La valeur par défaut est `false`.

jms.UserName

La propriété jms.UserName indique le nom d'utilisateur du fournisseur JMS. Cette valeur est facultative.

Il n'y a pas de valeur par défaut.

JvmMaxHeapSize

La propriété JvmMaxHeapSize indique la taille de segment maximale pour l'agent (en megaoctets).

La propriété n'est valide que si la valeur de la propriété RepositoryDirectory est définie sur <REMOTE>.

La valeur par défaut est 128 Mo.

JvmMaxNativeStackSize

La propriété JvmMaxNativeStackSize indique l'espace mémoire natif maximal pour l'agent (en kilo-octets).

La propriété n'est valide que si la valeur de la propriété RepositoryDirectory est définie sur <REMOTE>.

La valeur par défaut est 128 Ko.

JvmMinHeapSize

La propriété JvmMinHeapSize indique la taille de segment minimale pour l'agent (en megaoctets).

La propriété n'est valide que si la valeur de la propriété RepositoryDirectory est définie sur <REMOTE>.

La valeur par défaut est 1 Mo.

ListenerConcurrency

La propriété ListenerConcurrency prend en charge le traitement de plusieurs unités d'exécution dans WebSphere MQ Listener lorsque ICS est le courtier d'intégration. Elle permet l'écriture par lots de plusieurs événements sur la base de données, ce qui améliore les performances du système.

Cette propriété n'est valide que pour les connecteurs qui utilisent le transfert MQ. La valeur de la propriété DeliveryTransport doit être définie sur `MQ`.

La valeur par défaut est 1.

Locale

La propriété Locale indique le code de langue, le pays ou le territoire et, le cas échéant, le jeu de codes de caractères associé. La valeur de cette propriété détermine les conventions culturelles telles que le classement et l'ordre de tri des données, les formats de date et d'heure, ainsi que les symboles monétaires utilisés.

Le format d'un nom d'environnement local est le suivant :
ll_TT.codeset

où :
ll est un code de langue à deux caractères (en minuscules)
TT est un code pays ou territoire à deux caractères (en majuscules)
codeset est le nom du jeu de codes de caractères associé (facultatif).

Par défaut, n'est affiché qu'un sous-ensemble des paramètres régionaux pris en charge. Pour ajouter d'autres valeurs prises en charge à la liste, modifiez le fichier \Data\Std\stdConnProps.xml dans le répertoire <*ProductDir*>\bin. Pour plus d'informations, voir l'annexe Connector Configurator de ce guide.

Si le connecteur n'a pas été internationalisé, la seule valeur correcte pour cette propriété est `en_US`. Pour déterminer si un connecteur spécifique a été internationalisé, consultez le guide utilisateur de l'adaptateur.

La valeur par défaut est `en_US`.

LogAtInterchangeEnd

La propriété LogAtInterchangeEnd indique s'il faut consigner les erreurs dans le journal du courtier d'intégration.

La consignation des erreurs dans le journal active également la notification par courrier électronique qui, lorsque des erreurs ou erreurs fatales ont lieu, génère des messages électroniques pour le destinataire spécifié par la valeur MESSAGE_RECIPIENT dans le fichier InterchangeSystem.cfg. Par exemple, lorsque la connexion entre un connecteur et son application est interrompue, si la valeur de LogAtInterChangeEnd est `true`, un courrier électronique est envoyé au destinataire indiqué.

Cette propriété n'est valide que si la valeur de la propriété RespositoryDirectory est définie sur <REMOTE> (la valeur de BrokerType est ICS).

La valeur par défaut est `false`.

MaxEventCapacity

La propriété MaxEventCapacity indique le nombre maximal d'événements contenus dans la mémoire tampon du contrôleur. Cette propriété est utilisée par la fonctionnalité de contrôle de flux.

Cette propriété n'est valide que si la valeur de la propriété RespositoryDirectory est définie sur <REMOTE> (la valeur de BrokerType est ICS).

La valeur peut être un nombre entier positif compris entre 1 et 2147483647.

La valeur par défaut est 2147483647.

MessageFileName

La propriété MessageFileName indique le nom du fichier de messages du connecteur. L'emplacement standard de ce fichier de messages est \connectors\messages, dans le répertoire produit. Si le fichier de messages n'est pas situé à l'emplacement standard, indiquez son nom dans un chemin d'accès absolu.

S'il n'existe pas de fichier de messages, le connecteur utilise `InterchangeSystem.txt` comme fichier de messages. Ce fichier est situé dans le répertoire produit.

Remarque : Pour déterminer si un connecteur a son propre fichier de messages, reportez-vous au guide d'utilisation de l'adaptateur.

La valeur par défaut est `InterchangeSystem.txt`.

MonitorQueue

La propriété MonitorQueue indique la file d'attente logique utilisée par le connecteur pour surveiller les événements en double.

Elle n'est valide que si la valeur de la propriété DeliveryTransport est `JMS` et la valeur de DuplicateEventElimination est `true`.

La valeur par défaut est *<CONNECTORNAME>*`/MONITORQUEUE`

OADAutoRestartAgent

La propriété OADAutoRestartAgent indique si le connecteur utilise la fonction de redémarrage automatique et éloigné. Cette fonction utilise le démon d'activation d'objets (OAD, Object Activation Daemon) déclenché par WebSphere MQ pour redémarrer le connecteur après un arrêt anormal ou pour démarrer un connecteur éloigné à partir du moniteur système.

Cette propriété doit avoir la valeur `true` pour que la fonction de redémarrage automatique et à distance soit activée. Pour plus d'informations sur la configuration de la fonction de l'OAD déclenché par WebSphere MQ, voir le document *Installation Guide for Windows* ou *for UNIX*.

Cette propriété n'est valide que si la valeur de la propriété RespositoryDirectory est définie sur <REMOTE> (la valeur de BrokerType est ICS).

La valeur par défaut est `false`.

OADMaxNumRetry

La propriété OADMaxNumRetry indique le nombre maximal de tentatives de redémarrage du connecteur après un arrêt anormal, automatiquement tentées par l'OAD déclenché par WebSphere MQ. Pour cela, la propriété OADAutoRestartAgent doit avoir la valeur `true`.

Cette propriété n'est valide que si la valeur de la propriété RespositoryDirectory est définie sur <REMOTE> (la valeur de BrokerType est ICS).

La valeur par défaut est `1000`.

OADRetryTimeInterval

La propriété OADRetryTimeInterval indique pour l'OAD déclenché par WebSphere MQ la durée en minutes entre les tentatives de relance. Si l'agent du connecteur ne redémarre pas durant cet intervalle, le contrôleur du connecteur demande à l'OAD de redémarrer l'agent du connecteur. L'OAD répète cette opération autant de fois que spécifié par la propriété OADMaxNumRetry. Pour cela, la propriété OADAutoRestartAgent doit avoir la valeur `true`.

Cette propriété n'est valide que si la valeur de la propriété RespositoryDirectory est définie sur <REMOTE> (la valeur de BrokerType est ICS).

La valeur par défaut est `10`.

PollEndTime

La propriété PollEndTime indique l'heure d'arrêt de l'interrogation de la file d'attente des événements. Le format est *HH:MM*, dans lequel *HH* représente les heures (de 0 à 23) et *MM* représente les secondes (de 0 à 59).

Vous devez saisir une valeur correcte pour cette propriété. La valeur par défaut est `HH:MM` sans valeur indiquée, mais elle doit être précisée.

Si l'exécution de l'adaptateur détecte :

- que PollStartTime est défini et PollEndTime n'est pas défini, ou
- que PollEndTime est défini et PollStartTime n'est pas défini,

l'interrogation utilisera la valeur configurée pour la propriété PollFrequency.

PollFrequency

La propriété PollFrequency indique la durée (en millisecondes) entre la fin de la dernière interrogation et le début de la suivante. Il ne s'agit pas de l'intervalle entre chaque interrogation. Le principe est le suivant :

- Lancez une interrogation pour obtenir le nombre d'objets spécifié par la propriété PollQuantity.
- Traitez ces objets. Pour certains connecteurs, ceci peut se faire en partie sur des unités d'exécution séparées, qui s'exécutent de manière asynchrone jusqu'à l'interrogation suivante.
- Attendez pendant l'intervalle indiqué par la propriété PollFrequency.
- Répétez le cycle.

La valeurs suivantes sont valides pour cette propriété :

- Un nombre de millisecondes (un entier positif).
- Le mot `no`, pour que le connecteur n'émette pas d'interrogation. Saisissez le mot en minuscules.
- Le mot `key`, pour que le connecteur émette des interrogations uniquement lorsque vous tapez la lettre `p` dans la fenêtre d'invite de commande du connecteur. Saisissez le mot en minuscules.

La valeur par défaut est `10000`.

Important : Certains connecteurs sont limités dans l'utilisation de cette propriété. Ces restrictions sont décrites dans le chapitre sur l'installation et la configuration de l'adaptateur.

PollQuantity

La propriété PollQuantity désigne le nombre d'éléments de l'application pour lesquels le connecteur doit émettre des interrogations. Si l'adaptateur dispose d'une propriété spécifique au connecteur pour définir le nombre d'interrogations, la valeur définie dans cette propriété spécifique remplace la valeur de la propriété standard.

Cette propriété n'est valide que si la valeur de la propriété `DeliveryTransport` est `JMS`, et si la propriété `ContainerManagedEvents` a une valeur.

Un message électronique est également considéré comme étant un événement. Lorsqu'il est interrogé pour un courrier électronique, le connecteur agit comme suit :

- Lorsqu'il est interrogé une fois, le connecteur détecte le corps du message, qu'il lit comme une pièce jointe. Comme aucun gestionnaire de données n'a été spécifié pour ce type mime, il ignorera le message.
- Le connecteur traite la première pièce jointe BO. Le gestionnaire de données est disponible pour ce type MIME : l'objet métier est envoyé à Visual Test Connector.
- Lorsqu'il est interrogé pour la deuxième fois, le connecteur traite la deuxième pièce jointe BO. Le gestionnaire de données est disponible pour ce type MIME : l'objet métier est envoyé à Visual Test Connector.
- Une fois acceptée, la troisième pièce jointe BO peut être transmise.

PollStartTime

La propriété PollStartTime indique l'heure de démarrage de l'interrogation de la file d'attente des événements. Le format est *HH:MM*, dans lequel *HH* représente les heures (de 0 à 23) et *MM* représente les secondes (de 0 à 59).

Vous devez saisir une valeur correcte pour cette propriété. La valeur par défaut est `HH:MM` sans valeur indiquée, mais elle doit être modifiée.

Si l'exécution de l'adaptateur détecte :

- que PollStartTime est défini et PollEndTime n'est pas défini, ou
- que PollEndTime est défini et PollStartTime n'est pas défini,

l'interrogation utilisera la valeur configurée pour la propriété PollFrequency.

RepositoryDirectory

La propriété RepositoryDirectory indique l'emplacement du référentiel à partir duquel le connecteur lit les schémas XML qui stockent les métadonnées pour la définition des objets métier.

Si ICS est le courtier d'intégration, cette valeur doit être définie sur <REMOTE>, car le connecteur obtient ces informations à partir du référentiel d'InterChange Server.

Lorsque le courtier d'intégration est un courtier de message WebSphere ou WAS, cette valeur est définie par défaut sur <*ProductDir*>\repository. Toutefois, elle peut être définie sur tout nom valide de répertoire.

RequestQueue

La propriété RequestQueue précise la file d'attente utilisée par le courtier d'intégration pour envoyer des objets métier au connecteur.

Cette propriété n'est valide que si la valeur de la propriété DeliveryTransport est `JMS`.

La valeur par défaut est <*CONNECTORNAME*>/`REQUESTQUEUE`.

ResponseQueue

La propriété ResponseQueue désigne la file d'attente de réponses JMS, qui transmet un message de réponse depuis l'architecture du connecteur vers le courtier d'intégration. Lorsqu'ICS est le courtier d'intégration, le serveur envoie la requête et attend un message de réponse dans la file d'attente de réponses JMS.

Cette propriété n'est valide que si la valeur de la propriété DeliveryTransport est définie sur `JMS`.

La valeur par défaut est `<CONNECTORNAME>/RESPONSEQUEUE`.

RestartRetryCount

La propriété RestartRetryCount indique le nombre de tentatives de redémarrage du connecteur. Lorsqu'elle est utilisée pour un connecteur parallèle, cette propriété indique le nombre de tentatives de redémarrage du composant spécifique à l'application du connecteur client par le composant spécifique à l'application du connecteur maître.

La valeur par défaut est `7`.

RestartRetryInterval

La propriété RestartRetryInterval indique l'intervalle en minutes pendant lequel le connecteur tente de se redémarrer. Lorsqu'elle est utilisée pour un connecteur parallèle, cette propriété indique l'intervalle entre les tentatives de redémarrage du composant spécifique à l'application du connecteur client par le composant spécifique à l'application du connecteur maître.

Les valeurs possibles vont de 1 à 2147483647.

La valeur par défaut est `1`.

ResultsSetEnabled

La propriété ResultsSetEnabled active ou désactive la prise en charge de l'ensemble des résultats lorsque Information Integrator est actif. Cette propriété ne peut être utilisée que si l'adaptateur prend en charge DB2 Information Integrator.

Cette propriété n'est valide que si la valeur de DeliveryTransport est `JMS` et la valeur de BrokerType est `WMQI`.

La valeur par défaut est `false`.

ResultsSetSize

La propriété ResultsSetSize définit le nombre maximum d'objets métier pouvant être retournés à Information Integrator. Cette propriété ne peut être utilisée que si l'adaptateur prend en charge DB2 Information Integrator.

Cette propriété n'est valide que si la propriété ResultsSetEnabled est définie sur `true`.

La valeur par défaut est `0`. Ce qui signifie que la taille de l'ensemble de résultats est illimitée.

RHF2MessageDomain

La propriété RHF2MessageDomain vous permet de configurer la valeur du nom de domaine de la zone dans l'en-tête JMS. Lorsque les données sont envoyées à un courtier de message WebSphere par transfert JMS, l'architecture de l'adaptateur écrit les informations de l'en-tête JMS, avec un nom de domaine et une valeur fixe `mrm`. Un nom de domaine configurable vous permet d'analyser comment le courtier de messages WebSphere traite les données de message.

Voici un exemple d'en-tête :

```
<mcd><Msd>mrm</Msd><Set>3</Set><Type>
Retek_POPhyDesc</Type><Fmt>CwXML</Fmt></mcd>
```

Cette propriété n'est valide que si la valeur de BrokerType est WMQI ou WAS. De plus, elle n'est valide que si la valeur de la propriété DeliveryTransport est `JMS` et si la valeur de WireFormat est `CwXML`.

Les valeurs possibles sont `mrm` et `xml`. La valeur par défaut est `mrm`.

SourceQueue

La propriété SourceQueue désigne la file d'attente source JMS de l'architecture du connecteur qui assure la transmission garantie d'événements pour les connecteur activés par JMS qui utilisent un magasin d'événements JMS. Pour plus d'informations, voir «ContainerManagedEvents», à la page 76.

Cette propriété n'est valide que si la valeur de DeliveryTransport est `JMS` et si une valeur est indiquée pour ContainerManagedEvents.

La valeur par défaut est *<CONNECTORNAME>*`/SOURCEQUEUE`.

SynchronousRequestQueue

La propriété SynchronousRequestQueue transmet les messages de requête qui requièrent une réponse synchrone depuis l'architecture du connecteur vers le courtier. Cette file d'attente n'est nécessaire que si le connecteur utilise l'exécution synchrone. Avec l'exécution synchrone, l'architecture du connecteur envoie un message à la file d'attente de requêtes synchrones et attend une réponse du courtier sur la file d'attente de réponse synchrone. La réponse envoyée au connecteur a un ID de corrélation qui correspond à l'ID du message d'origine.

Cette propriété n'est valide que si la valeur de la propriété DeliveryTransport est définie sur `JMS`.

La valeur par défaut est *<CONNECTORNAME>*`/SYNCHRONOUSREQUESTQUEUE`

SynchronousRequestTimeout

La propriété SynchronousRequestTimeout indique la durée d'attente en millisecondes d'une réponse à une requête synchrone. Si la réponse n'est pas reçue dans l'intervalle de temps indiqué, le connecteur transfère le message de requête synchrone original, accompagné d'un message d'erreur, dans la file d'attente des erreurs.

Cette propriété n'est valide que si la valeur de la propriété DeliveryTransport est définie sur `JMS`.

La valeur par défaut est `0`.

SynchronousResponseQueue

La propriété SynchronousResponseQueue transmet les messages de réponse à une requête synchrone entre le courtier et l'architecture du connecteur. Cette file d'attente n'est nécessaire que si le connecteur utilise l'exécution synchrone.

Cette propriété n'est valide que si la valeur de la propriété DeliveryTransport est définie sur `JMS`.

La valeur par défaut est `<CONNECTORNAME>/SYNCHRONOUSRESPONSEQUEUE`

TivoliMonitorTransactionPerformance

La propriété TivoliMonitorTransactionPerformance indique si IBM Tivoli Monitoring for Transaction Performance (ITMTP) est appelé lors de l'exécution.

La valeur par défaut est `false`.

WireFormat

La propriété WireFormat précise le format de message sur le transfert :

- Si la valeur de la propriété RepositoryDirectory est un répertoire local, la valeur est `CwXML`.
- Si la valeur de la propriété RepositoryDirectory est un répertoire éloigné, la valeur est `CwBO`.

WsifSynchronousRequestTimeout

La propriété WsifSynchronousRequestTimeout indique la durée d'attente en millisecondes d'une réponse à une requête synchrone. Si la réponse n'est pas reçue dans l'intervalle de temps indiqué, le connecteur transfère le message de requête synchrone original, accompagné d'un message d'erreur, dans la file d'attente des erreurs.

Cette propriété n'est valide que si la valeur de BrokerType est définie sur `WAS`.

La valeur par défaut est 0.

XMLNameSpaceFormat

La propriété XMLNameSpaceFormat précise des espaces de nom courts ou longs dans le format XML des définitions d'objet métier.

Cette propriété n'est valide que si la valeur de BrokerType est WMQI ou WAS.

La valeur par défaut est `short`.

Annexe B. Utilisation de Connector Configurator

Cette annexe décrit comment utiliser Connector Configurator afin de définir les valeurs des propriétés de configuration pour votre adaptateur.

Connector Configurator vous permet de :

- créer un modèle de propriété spécifique au connecteur pour la configuration de votre connecteur ;
- créer un fichier de configuration ;
- définir les propriétés dans un fichier de configuration.

Les sujets traités dans cette annexe sont les suivants :

Présentation de Connector Configurator

Connector Configurator vous permet de configurer le connecteur de votre adaptateur à utiliser avec ces courtiers d'intégration :

- WebSphere InterChange Server (ICS) ;
- WebSphere MQ Integrator, WebSphere MQ Integrator Broker et WebSphere Business Integration Message Broker, appelés courtiers de messages WebSphere (WMQI) ;
- WebSphere Application Server (WAS).

Si votre adaptateur prend en charge DB2 Information Integrator, utilisez les options et les propriétés standard DB2 II (voir la colonne Remarques de l'annexe Propriétés standard).

Connector Configurator vous permet de :

- créer un **modèle de propriété spécifique au connecteur** pour la configuration de votre connecteur ;
- créer un **fichier de configuration du connecteur** (vous devez créer un fichier de configuration pour chaque connecteur installé) ;
- définir les propriétés dans un fichier de configuration.
  Vous devez peut-être modifier les valeurs par défaut définies pour les propriétés dans les modèles du connecteur. Vous devez également déterminer les définitions d'objet métier prises en charge, indiquer les mappes à utiliser avec les collaborations à l'aide d'ICS et spécifier les paramètres d'application de messagerie, de journalisation, de trace ainsi que ceux du gestionnaire de données, le cas échéant.

Le mode dans lequel vous exécutez Connector Configurator et le type de fichier de configuration que vous utilisez peuvent différer en fonction du courtier

d'intégration que vous exécutez. Par exemple, si vous utilisez WMQI comme courtier, vous exécutez Connector Configurator directement, et non à partir de System Manager (voir «Exécution de Connector Configurator en mode autonome», à la page 92).

Les propriétés de configuration du connecteur incluent des propriétés de configuration standard (les propriétés communes à tous les connecteurs) et des propriétés spécifiques au connecteur (propriétés requises par le connecteur pour une technologie ou une application spécifique).

Dans la mesure où les **propriétés standard** sont utilisées par tous les connecteurs, vous n'avez pas besoin de définir ces propriétés de tout pièce ; Connector Configurator les incorpore à votre fichier de configuration dès que vous créez ce fichier. Cependant, vous devez définir la valeur de chaque propriété standard dans Connector Configurator.

L'intervalle des propriétés standard peut être différent pour tous les courtiers et toutes les configurations. Certaines propriétés ne sont disponibles que si vous attribuez une valeur spécifique à d'autres propriétés. La fenêtre des propriétés standard dans Connector Configurator affiche les propriétés disponibles pour votre configuration spécifique.

Cependant, pour les **propriétés spécifiques au connecteur**, vous devez d'abord définir les propriétés, puis leur attribuer une valeur. Pour ce faire, créez un modèle de propriétés spécifiques au connecteur pour votre adaptateur particulier. Il se peut qu'un modèle soit déjà configuré dans votre système, auquel cas vous pouvez l'utiliser. Dans le cas contraire, suivez les étapes dans la section «Création d'un modèle», à la page 94 pour configurer un nouveau modèle.

Utilisation des connecteurs sous UNIX

Connector Configurator s'exécute uniquement dans un environnement Windows. Si vous exécutez le connecteur dans un environnement UNIX, utilisez Connector Configurator dans Windows pour modifier le fichier de configuration, puis copiez le fichier dans votre environnement UNIX.

Certaines propriétés de Connector Configurator utilisent des chemins de répertoire, par défaut avec la convention de dénomination propre à Windows. Si vous utilisez le fichier de configuration sous UNIX, vous devrez modifier les chemins d'accès aux répertoires pour qu'ils respectent les conventions UNIX. Afin d'activer les bonnes règles de système d'exploitation pour la validation étendue, sélectionnez le système d'exploitation cible dans la liste déroulante de la barre d'outils.

Démarrage de Connector Configurator

Vous pouvez démarrer et exécuter Connector Configurator dans l'un de ces deux modes :

- de manière indépendante, en mode autonome ;
- à partir de System Manager.

Exécution de Connector Configurator en mode autonome

Vous pouvez exécuter Connector Configurator en mode autonome, sans exécuter le System Manager, et utiliser les fichiers de configuration quel que soit votre courtier.

Pour ce faire, procédez comme suit :

- Dans **Démarrer>Programmes**, cliquez sur **IBM WebSphere Business Integration Adapters>IBM WebSphere Business Integration Toolset>Connector Configurator**.
- Sélectionnez **File>New>Connector Configuration**.
- Lorsque vous cliquez sur le menu déroulant en regard de **System Connectivity Integration Broker**, vous pouvez sélectionner ICS, WebSphere Message Brokers ou WAS, en fonction de votre courtier.

Vous pouvez choisir d'exécuter Connector Configurator en mode autonome pour créer le fichier, puis de vous connecter à System Manager afin de l'enregistrer dans un projet System Manager (voir «Remplissage d'un fichier de configuration», à la page 99).

Exécution de Connector Configurator à partir de System Manager

Vous pouvez exécuter Connector Configurator à partir de System Manager.

Pour exécuter Connector Configurator, procédez comme suit :

1. Ouvrez System Manager.
2. Dans la fenêtre System Manager, développez l'icône **Integration Component Libraries** et mettez en évidence **Connectors**.
3. Dans la barre de menus de System Manager, cliquez sur **Tools>Connector Configurator**. La fenêtre Connector Configurator affiche la boîte de dialogue **New Connector**.
4. Lorsque vous cliquez sur le menu déroulant en regard de **System Connectivity Integration Broker**, vous pouvez sélectionner ICS, WebSphere Message Brokers ou WAS, en fonction de votre courtier.

Pour modifier un fichier de configuration existant, procédez comme suit :

- Dans la fenêtre System Manager, sélectionnez l'un des fichiers de configuration répertoriés dans le dossier du connecteur et cliquez dessus avec le bouton droit. Connector Configurator affiche le fichier de configuration avec le type de courtier d'intégration et le nom de fichier dans la partie supérieure.
- Dans Connector Configurator, sélectionnez **File>Open**. Sélectionnez le nom du fichier de configuration du connecteur dans un projet ou dans le répertoire dans lequel il est stocké.
- Cliquez sur l'onglet Standard Properties (Propriétés standard) pour afficher les propriétés contenues dans ce fichier de configuration.

Création d'un modèle de propriétés spécifiques au connecteur

Pour créer un fichier de configuration pour votre connecteur, vous avez besoin d'un modèle de propriétés spécifiques au connecteur et des propriétés standard fournies par le système.

Vous pouvez créer un nouveau modèle pour les propriétés spécifiques au connecteur ou utiliser comme modèle une définition de connecteur existante.

- Pour créer un modèle, voir «Création d'un modèle», à la page 94.
- Pour utiliser un fichier existant, il vous suffit de modifier un modèle existant et de l'enregistrer sous le nouveau nom. Vous pouvez trouver des modèles existants dans le répertoire `\WebSphereAdapters\bin\Data\App`.

Création d'un modèle

Cette section décrit comment créer des propriétés dans le modèle, définir les valeurs et les caractéristiques générales de ces propriétés et indiquer toutes les dépendances entre les propriétés. Ensuite, vous pouvez utiliser le modèle comme base pour la création d'un nouveau fichier de configuration du connecteur.

Pour créer un modèle dans Connector Configurator, procédez comme suit :

1. Cliquez sur **File>New>Connector-Specific Property Template**.
2. La boîte de dialogue **Connector-Specific Property Template** s'affiche.
   - Entrez le nom du nouveau modèle dans la zone **Name** située sous **Input a New Template Name**. Vous voyez de nouveau ce nom lorsque vous ouvrez la boîte de dialogue pour créer un fichier de configuration à partir d'un modèle.
   - Pour afficher les définitions de propriétés spécifiques au connecteur dans n'importe quel modèle, sélectionnez le nom de ce modèle dans l'écran **Template Name**. La liste des définitions de propriétés contenues dans ce modèle apparaît dans l'écran **Template Preview**.
3. Vous pouvez utiliser un modèle existant dont les définitions de propriétés sont similaires à celles requises par votre connecteur comme point de départ pour votre modèle. Si aucun des modèles n'affiche les propriétés spécifiques au connecteur, vous devez en créer un.
   - Si vous prévoyez de modifier un modèle existant, sélectionnez le nom de ce modèle dans la liste située dans le tableau **Template Name** sous **Select the Existing Template to Modify: Find Template**.
   - Ce tableau affiche les noms de tous les modèles disponibles. Vous pouvez également rechercher un modèle.

Indication des caractéristiques générales

Lorsque vous cliquez sur **Next** pour sélectionner un modèle, la boîte de dialogue **Properties - Connector-Specific Property Template** s'affiche. Cette boîte de dialogue contient des onglets pour les caractéristiques générales des propriétés définies et pour les restrictions liées aux valeurs. L'écran général contient les zones suivantes :

- **General:**
  Property Type
  Property Subtype
  Updated Method
  Description
- **Flags**
  Standard flags
- **Custom Flag**
  Flag

Le **Property Subtype** peut être sélectionné si le **Property Type** est String. Cette valeur facultative entraîne la vérification de la syntaxe lors de l'enregistrement du fichier de configuration. La valeur par défaut est un espace, ce qui signifie que la propriété n'a reçu aucun sous-type.

Une fois que vous avez sélectionné les caractéristiques générales de la propriété, cliquez sur l'onglet **Value**.

Indication de valeurs

L'onglet **Value** vous permet de définir la longueur maximum, le nombre maximum de valeurs multiples, une valeur par défaut ou un intervalle de valeurs pour la propriété. Il autorise également les valeurs modifiables. Pour ce faire, procédez comme suit :

1. Cliquez sur l'onglet **Value**. Le panneau d'affichage des valeurs remplace le panneau d'affichage général.
2. Sélectionnez le nom de la propriété dans l'écran **Edit properties**.
3. Dans les zones relatives à la **longueur maximum** et au **nombre maximum de valeurs multiples**, entrez les valeurs de votre choix.

Pour créer une valeur de propriété, procédez comme suit :

1. Cliquez à l'aide du bouton droit de la souris sur le carré à gauche de l'en-tête de colonne Value.
2. Dans le menu en incrustation, sélectionnez **Add** pour afficher la boîte de dialogue Property Value. Selon le type de propriété, vous pourrez entrer une valeur avec ou sans un intervalle.
3. Entrez la nouvelle valeur de propriété et cliquez sur OK. La valeur apparaît dans le panneau **Value** situé dans la partie droite.

Le panneau **Value** contient un tableau comprenant trois colonnes :

La colonne **Value** contient la valeur que vous avez entrée dans la boîte de dialogue **Property Value** et toutes les valeurs que vous avez précédemment créées.

La colonne **Default Value** vous permet d'indiquer n'importe quelle valeur comme valeur par défaut.

La colonne **Value Range** contient l'intervalle que vous avez entré dans la boîte de dialogue **Property Value**.

Une fois que vous avez créé une valeur et qu'elle apparaît dans la grille, vous pouvez la modifier dans le tableau.

Pour modifier une valeur existante dans le tableau, sélectionnez une ligne entière en cliquant sur le numéro de ligne. Ensuite, cliquez avec le bouton droit dans la zone **Value** et cliquez sur **Edit Value**.

Définition des dépendances

Une fois les modifications apportées aux onglets **General** et **Value**, cliquez sur **Next**. La boîte de dialogue **Dependencies - Connector-Specific Property Template** s'affiche.

Une propriété dépendante est une propriété qui est incluse dans le modèle et utilisée dans le fichier de configuration *uniquement si* la valeur d'une autre propriété respecte une condition spécifique. Par exemple, `PollQuantity` apparaît dans le modèle uniquement si JMS est le mécanisme de transfert et que `DuplicateEventElimination` a la valeur `True`.
Pour faire en sorte qu'une propriété soit dépendante et définir la condition dont elle dépend, procédez comme suit :

1. Dans l'écran **Available Properties**, sélectionnez la propriété qui doit devenir dépendante.
2. Dans la zone **Select Property**, utilisez le menu déroulant pour sélectionner la propriété qui conservera la valeur conditionnelle.

3. Dans la zone **Condition Operator**, sélectionnez l'une des valeurs suivantes :
   == (égal à)
   != (différent de)
   > (supérieur à)
   < (inférieur à)
   >= (supérieur ou égal à)
   <= (inférieur ou égal à)
4. Dans la zone **Conditional Value**, entrez la valeur requise pour que la propriété dépendante soit incluse dans le modèle.
5. La propriété dépendante est mise en évidence dans l'écran **Available Properties** ; cliquez sur une flèche pour la déplacer vers l'écran **Dependent Property**.
6. Cliquez sur **Finish**. Connector Configurator stocke les informations que vous avez entrées sous la forme d'un document XML, sous `\data\app` dans le répertoire `\bin` où vous avez installé Connector Configurator.

Configuration des noms de chemins

Voici quelques règles générales pour la configuration des noms de chemins :

- Sous Windows et UNIX, un nom de fichier est limité à 255 caractères.
- Sous Windows, le nom de chemin absolu doit respecter le format [Unité:][Répertoire]\nom_de_fichier. Par exemple C:\WebSphereAdapters\bin\Data\Std\StdConnProps.xml
  Sous UNIX, le premier caractère doit être /.
- Un nom de file d'attente ne doit pas comporter d'espaces, ni à l'intérieur ni à la fin.

Création d'un fichier de configuration

Lorsque vous créez un fichier de configuration, vous devez lui attribuer un nom et sélectionner un courtier d'intégration.

Vous sélectionnerez également un système d'exploitation pour effectuer une validation étendue du fichier. La barre d'outils dispose d'une liste déroulante nommée **Target System**, dans laquelle vous sélectionnerez le système d'exploitation cible pour activer la validation étendue des propriétés. Les options sont : Windows, UNIX, Other (autres) et None (désactive la validation étendue). Au démarrage, la valeur par défaut est Windows.

Pour démarrer Connector Configurator :

- Dans la fenêtre System Manager, sélectionnez **Connector Configurator** dans le menu **Tools**. Connector Configurator s'ouvre.
- En mode autonome, démarrez Connector Configurator.

Pour définir le système d'exploitation et activer la validation étendue du fichier de configuration :

- Cliquez sur la liste déroulante **Target System:** de la barre de menus.
- Sélectionnez le système d'exploitation de votre environnement d'exécution.

Sélectionnez ensuite **File>New>Connector Configuration**. Dans la fenêtre New Connector, entrez le nom du nouveau connecteur.

Vous devez également sélectionner un courtier d'intégration. Le courtier que vous sélectionnez détermine les propriétés qui apparaîtront dans le fichier de configuration. Pour sélectionner un courtier, procédez comme suit :

- Dans la zone **Integration Broker**, sélectionnez ICS, les courtiers de messages WebSphere ou la connectivité WAS.
- Renseignez les zones restantes de la fenêtre **New Connector**, comme décrit précédemment dans ce chapitre.

Création d'un fichier de configuration pour un modèle spécifique au connecteur

Une fois que vous avez créé un modèle spécifique au connecteur, vous pouvez l'utiliser pour créer un fichier de configuration :

1. Indiquez le système d'exploitation pour la validation étendue du fichier de configuration, à l'aide de la liste déroulante **Target System:** de la barre de menus (voir ci-dessus "Création d'un nouveau fichier de configuration").
2. Cliquez sur **File>New>Connector Configuration**.
3. La boîte de dialogue **New Connector** contient les zones suivantes :
   - **Name**

     Entrez le nom du connecteur. Les noms font la différence entre les majuscules et les minuscules. Le nom que vous entrez doit être unique et cohérent avec le nom de fichier d'un connecteur installé sur le système.

     Important : Connector Configurator ne contrôle pas l'orthographe du nom que vous entrez. Vous devez vérifier que le nom est correct.
   - **System Connectivity**

     Cliquez sur ICS, Courtiers de messages WebSphere ou WAS.
   - **Select Connector-Specific Property Template**

     Tapez le nom du modèle conçu pour votre connecteur. Les modèles disponibles s'affichent dans l'écran **Template Name**. Lorsque vous sélectionnez un nom dans l'écran Template Name, l'écran **Property Template Preview** affiche les propriétés spécifiques au connecteur qui ont été définies dans ce modèle.

     Sélectionnez le modèle à utiliser et cliquez sur **OK**.
4. Un écran de configuration apparaît pour le connecteur que vous configurez. La barre de titre contient le nom du courtier d'intégration et du connecteur. Vous pouvez entrer les valeurs de toutes les zones pour terminer la définition maintenant, ou enregistrer le fichier et renseigner les zones ultérieurement.
5. Pour enregistrer le fichier, cliquez sur **File>Save>To File** ou sur **File>Save>To Project**. Pour exécuter un enregistrement dans un projet, System Manager doit être en cours d'exécution.
   Si vous enregistrez un fichier, la boîte de dialogue **Save File Connector** apparaît. Sélectionnez `*.cfg` comme type de fichier, vérifiez dans la zone File Name que le nom est correctement orthographié et que sa casse est correcte, accédez au répertoire dans lequel vous souhaitez enregistrer le fichier et cliquez sur **Save**. L'écran d'état affiché dans le panneau de message de Connector Configurator indique que le fichier de configuration a été créé.

   Important : Le nom et le chemin du répertoire que vous avez définis ici doivent correspondre au nom et au chemin du fichier de configuration du connecteur que vous indiquez dans le fichier de démarrage du connecteur.

6. Pour remplir la définition du connecteur, entrez des valeurs dans les zones de chacun des onglets de la fenêtre Connector Configurator, comme décrit plus loin dans ce chapitre.

Utilisation d'un fichier existant

Vous disposez peut-être d'un fichier existant dans un ou plusieurs des formats suivants :

- Fichier de définition du connecteur.
  Il s'agit d'un fichier texte qui répertorie les propriétés et les valeurs par défaut applicables d'un connecteur spécifique. Certains connecteurs possèdent ce fichier dans un répertoire `\repository` fourni dans leur package d'origine (en général, le fichier a l'extension `.txt` ; par exemple, CN_XML.txt pour le connecteur XML).
- Fichier référentiel ICS.
  Les définitions utilisées dans une implémentation ICS précédente du connecteur peuvent être accessibles dans un fichier référentiel qui a été utilisé pour la configuration de ce connecteur. En général, ce type de fichier a l'extension `.in` ou `.out`.
- Fichier de configuration précédent pour le connecteur.
  En général, ce type de fichier a l'extension *.cfg.

Bien que certaines de ces sources de fichier puissent contenir tout ou partie des propriétés spécifiques au connecteur, le fichier de configuration du connecteur ne sera pas complet tant que vous n'aurez pas ouvert le fichier et défini les propriétés, comme décrit plus loin dans ce chapitre.

Pour utiliser un fichier existant afin de configurer un connecteur, vous devez ouvrir le fichier dans Connector Configurator, réviser la configuration et enregistrer de nouveau le fichier.

Pour ouvrir un fichier *.txt, *.cfg ou *.in dans un répertoire, procédez comme suit :

1. Dans Connector Configurator, cliquez sur **File>Open>From File**.
2. Dans la boîte de dialogue **Open File Connector**, sélectionnez l'un des types de fichier suivants pour afficher les fichiers disponibles :
   - Configuration (`*.cfg`)
   - Référentiel ICS (`*.in`, `*.out`)

     Sélectionnez cette option si vous avez utilisé un fichier référentiel pour configurer le connecteur dans un environnement ICS. Un fichier référentiel peut contenir plusieurs définitions de connecteur, qui apparaissent toutes lorsque vous ouvrez ce fichier.
   - Tous les fichiers (*.*)

     Sélectionnez cette option si un fichier `*.txt` a été fourni dans le package de l'adaptateur pour le connecteur ou qu'un fichier de définition avec une autre extension est disponible.
3. Dans l'écran du répertoire, accédez au fichier de définition du connecteur approprié, sélectionnez-le et cliquez sur **Open**.

Pour ouvrir une configuration de connecteur à partir d'un projet System Manager, procédez comme suit :

1. Démarrez System Manager. Vous pouvez ouvrir une configuration dans System Manager ou l'y enregistrer uniquement si vous avez démarré System Manager.
2. Démarrez Connector Configurator.

3. Cliquez sur **File>Open>From Project**.

Remplissage d'un fichier de configuration

Lorsque vous ouvrez un fichier de configuration ou un connecteur à partir d'un projet, la fenêtre Connector Configurator affiche l'écran de configuration qui contient les valeurs et les attributs courants.

Le titre de l'écran de configuration affiche le courtier d'intégration et le nom du connecteur, comme indiqué dans le fichier. Vérifiez que votre courtier est correct. Dans le cas contraire, modifiez la valeur du courtier avant de configurer le connecteur. Pour ce faire, procédez comme suit :

1. Dans l'onglet **Standard Properties (Propriétés standard)**, sélectionnez la zone de valeur pour la propriété `BrokerType`. Dans le menu déroulant, sélectionnez la valeur `ICS`, `WMQI` ou `WAS`.
2. L'onglet Standard Properties affiche les propriétés de connecteur associées au courtier sélectionné. La table indique les **Property name**, **Value**, **Type**, **Subtype** (si le Type est String), **Description** et **Update Method**.
3. Vous pouvez enregistrer le fichier maintenant ou renseigner les autres zones relatives à la configuration, comme décrit dans «Indication des définitions d'objets métier pris en charge», à la page 102.
4. Une fois la configuration terminée, cliquez sur **File>Save>To Project** ou sur **File>Save>To File**.

   Si vous enregistrez dans un fichier, sélectionnez `*.cfg` comme extension, sélectionnez l'emplacement correct pour le fichier et cliquez sur **Save**.

   Si plusieurs configurations de connecteur sont ouvertes, cliquez sur **Save All to File** pour enregistrer toutes les configurations dans un fichier ou cliquez sur **Save All to Project** pour enregistrer toutes les configurations du connecteur dans un projet System Manager.

   Avant de créer le fichier de configuration, vous utiliserez la liste déroulante **Target System** pour sélectionner le système d'exploitation cible et activer la validation étendue des propriétés.

   Avant d'enregistrer le fichier, Connector Configurator vérifie que vous avez défini des valeurs pour toutes les propriétés standard requises. Si vous n'avez pas défini de valeur pour l'une des propriétés standard requises, Connector Configurator affiche un message indiquant l'échec de la validation. Vous devez attribuer une valeur à la propriété pour pouvoir enregistrer le fichier de configuration.

   Si vous avez activé la validation étendue en sélectionnant Windows, UNIX ou Other dans la liste déroulante **Target System**, le système validera les propriétés de type et de sous-type, et affichera un message en cas d'échec de la validation.

Définition des propriétés d'un fichier de configuration

Lorsque vous créez et que vous nommez un nouveau fichier de configuration du connecteur, ou que vous ouvrez un fichier de configuration existant du connecteur, Connector Configurator affiche un écran de configuration avec des onglets pour les catégories des valeurs de configuration requises.

Connector Configurator requiert des valeurs pour les propriétés dans ces catégories pour les connecteurs s'exécutant sur tous les courtiers :

- Propriétés standard
- Propriétés spécifiques au connecteur

- Objets métier pris en charge
- Valeurs des fichiers journaux/fichiers de trace
- Gestionnaire de données (applicable pour les connecteurs qui utilisent la messagerie JMS avec une livraison des événements garantie)

Remarque : Pour les connecteurs utilisant la messagerie JMS, une catégorie supplémentaire peut s'afficher ; elle est associée à la configuration des gestionnaires de données qui convertissent les données en objets métier.

Pour les connecteurs qui s'exécutent sur **ICS**, des valeurs sont également requises pour ces propriétés :

- Mappes associées
- Ressources
- Messagerie (le cas échéant)
- Sécurité

Important : Connector Configurator accepte que les valeurs des propriétés soient tapées en caractères anglais ou avec d'autres jeux de caractères. Cependant, les noms des propriétés standard et des propriétés spécifiques au connecteur ainsi que les noms des objets métier pris en charge doivent uniquement utiliser le jeu de caractères anglais.

Les différences entre les propriétés standard et les propriétés spécifiques au connecteur sont les suivants :

- Les propriétés standard d'un connecteur sont partagées par le composant spécifique à l'application d'un connecteur et son courtier. Tous les connecteurs ont le même jeu de propriétés standard. Ces propriétés sont décrites dans l'Annexe A de chaque guide de l'adaptateur. Vous pouvez modifier une partie de ces valeurs uniquement.
- Les propriétés spécifiques à l'application s'appliquent uniquement au composant spécifique à l'application d'un connecteur, c'est-à-dire au composant qui interagit directement avec l'application. Chaque connecteur a des propriétés spécifiques à l'application qui sont propres à cette application. Certaines de ces propriétés fournissent des valeurs par défaut, et d'autres non ; vous pouvez modifier certaines des valeurs par défaut. Les chapitres relatifs à l'installation et à la configuration de chaque guide de l'adaptateur décrivent les propriétés spécifiques à l'application et les valeurs recommandées.

Les zones relatives aux **propriétés standard** et aux **propriétés spécifiques au connecteur** sont codées en couleur pour indiquer les éléments configurables :

- Une zone avec un arrière-plan gris indique une propriété standard. Vous pouvez modifier la valeur, mais vous ne pouvez pas modifier le nom ou supprimer la propriété.
- Une zone avec un arrière-plan blanc indique une propriété spécifique à l'application. Ces propriétés varient en fonction des besoins spécifiques de l'application ou du connecteur. Vous pouvez modifier la valeur et supprimer ces propriétés.
- Les zones de valeurs sont configurables.
- La zone **Update Method** s'affiche pour chaque propriété. Elle indique si le redémarrage d'un composant ou d'un agent est nécessaire pour activer les valeurs modifiées. Vous ne pouvez pas configurer ce paramètre.

Définition des propriétés standard du connecteur

Pour modifier la valeur d'une propriété standard, procédez comme suit :

1. Cliquez dans la zone dont vous souhaitez définir la valeur.
2. Entrez une valeur ou sélectionnez-en une dans le menu déroulant le cas échéant.

   Remarque : Si le Type de la propriété est String, la colonne Subtype peut contenir une valeur de sous-type. Ce sous-type sert pour la validation étendue de la propriété.
3. Une fois que vous avez entré toutes les valeurs pour les propriétés standard, vous pouvez exécuter les opérations suivantes :
   - Pour ignorer les modifications, conserver les valeurs d'origine et quitter Connector Configurator, cliquez sur **File>Exit** (ou fermez la fenêtre) et cliquez sur **No** lorsqu'un message vous demande si vous souhaitez enregistrer les modifications.
   - Pour entrer les valeurs des autres catégories dans Connector Configurator, sélectionnez l'onglet relatif à la catégorie. Les valeurs que vous entrez pour la catégories **Standard Properties (Propriétés standard)** (ou n'importe quelle autre catégorie) sont conservées lorsque vous passez à la catégorie suivante. Lorsque vous fermez la fenêtre, vous êtes invité à enregistrer ou à annuler les valeurs que vous avez entrées dans toutes les catégories.
   - Pour enregistrer les valeurs révisées, cliquez sur **File>Exit** (ou fermez la fenêtre) et sur **Yes** lorsqu'un message vous demande si vous souhaitez enregistrer les modifications. Vous pouvez également cliquer sur **Save>To File** dans le menu File ou la barre d'outils.

Pour plus d'informations sur une propriété standard donnée, cliquez sur l'entrée correspondante dans la colonne Description, dans la feuille à onglets Standard Properties. Si Extended Help est installé, un bouton flèche apparaît sur la droite. Un clic sur ce bouton ouvre une fenêtre d'aide, qui affiche des détails concernant la propriété standard.

Remarque : Si le bouton n'apparaît pas, c'est qu'il n'existe pas d'aide étendue pour cette propriété.

Les fichiers Extended Help sont installés dans le répertoire <*ProductDir*>\bin\Data\Std\Help\<*RegionalSetting*>\.

Configuration des propriétés spécifiques au connecteur

Vous pouvez ajouter ou modifier des noms, définir des valeurs, supprimer une propriété spécifique ou la chiffrer. La longueur par défaut d'une propriété est de 255 caractères.

1. Cliquez avec le bouton droit dans la partie supérieure gauche de la grille. Une barre de menus contextuelle apparaît. Cliquez sur **Add** pour ajouter une propriété. Pour ajouter une propriété enfant, cliquez avec le bouton droit sur le numéro de la ligne parent et cliquez sur **Add child**.
2. Entrez une valeur pour la propriété ou la propriété enfant.

   Remarque : Si la propriété est de Type String, vous pouvez sélectionner un sous-type dans la liste déroulante. Ce sous-type sert pour la validation étendue de la propriété.
3. Pour chiffrer une propriété, cochez la case **Encrypt**.

4. Pour plus d'informations sur une propriété donnée, cliquez sur l'entrée correspondante dans la colonne Description. Si Extended Help est installé, un bouton apparaît sur la droite. Un clic sur ce bouton ouvre une fenêtre d'aide, qui affiche des détails concernant la propriété.

   Remarque : Si le bouton n'apparaît pas, c'est qu'il n'existe pas d'aide étendue pour cette propriété.
5. Vous pouvez enregistrer ou ignorer les modifications, comme décrit pour «Définition des propriétés standard du connecteur», à la page 101.

Si les fichiers Extended Help sont installés et que la propriété AdapterHelpName n'est pas renseignée, Connector Configurator pointera sur les fichiers Extended Help spécifiques au connecteur, dans le répertoire <*ProductDir*>\bin\Data\App\Help\<*RegionalSetting*>\. Sinon, Connector Configurator pointera sur les fichiers Extended Help spécifiques au connecteur, dans le répertoire <*ProductDir*>\bin\Data\App\Help\<*AdapterHelpName*>\<*RegionalSetting*>\. Voir la propriété AdapterHelpName, décrite dans l'annexe Propriétés standard.

La zone Update Method affichée pour chaque propriété indique si le redémarrage d'un composant ou d'un agent est nécessaire à l'activation des valeurs modifiées.

Important : La modification du nom prédéfini d'une propriété de connecteur spécifique à l'application peut entraîner l'échec d'un connecteur. Le connecteur peut nécessiter certains noms de propriété pour se connecter à une application ou s'exécuter correctement.

Chiffrement des propriétés du connecteur

Pour chiffrer les propriétés spécifiques à l'application, cochez la case **Encrypt** dans la fenêtre des propriétés spécifiques au connecteur. Pour déchiffrer une valeur, décochez la case **Encrypt**, entrez la valeur appropriée dans la boîte de dialogue **Verification** et cliquez sur **OK**. Si la valeur entrée est correcte, elle est déchiffrée et s'affiche.

Le guide d'utilisateur de l'adaptateur pour chaque connecteur contient la liste et la description de chaque propriété ainsi que sa valeur par défaut.

Si une propriété a plusieurs valeurs, la case **Encrypt** apparaît pour la première valeur de la propriété. Lorsque vous sélectionnez **Encrypt**, toutes les valeurs de la propriété sont chiffrées. Pour déchiffrer plusieurs valeurs d'une propriété, décochez la case **Encrypt** pour la première valeur de la propriété, puis entrez la nouvelle valeur dans la boîte de dialogue **Verification**. Si la valeur entrée est une correspondance, toutes les valeurs multiples sont déchiffrées.

Méthode de mise à jour

Reportez-vous aux descriptions des méthodes de mise à jour, dans l'annexe Propriétés standard, sous «Présentation des valeurs des propriétés de configuration», à la page 66.

Indication des définitions d'objets métier pris en charge

Utilisez l'onglet **Supported Business Objects** dans Connector Configurator pour indiquer les objets métier que le connecteur utilisera. Vous devez indiquer les objets métier génériques et les objets métier spécifiques à l'application, et indiquer les associations pour les mappes entre les objets métier.

Remarque : Certains connecteurs nécessitent que des objets métier soient indiqués comme étant pris en charge pour pouvoir exécuter la notification des événements ou une configuration supplémentaire (à l'aide des méta-objets) avec leurs applications. Pour plus d'informations, voir *Connector Development Guide for C++* ou *Connector Development Guide for Java*.

Si ICS est votre courtier

Pour indiquer qu'une définition d'objet métier est prise en charge par le connecteur ou modifier les paramètres de prise en charge d'une définition d'objet métier existante, cliquez sur l'onglet **Supported Business Objects** et utilisez les zones suivants :

Nom de l'objet métier : Pour indiquer qu'une définition d'objet métier est prise en charge par le connecteur, avec System Manager en cours d'exécution, procédez comme suit :

1. Cliquez dans une zone vide de la liste **Business Object Name**. Une liste déroulante s'affiche, avec toutes les définitions d'objet métier qui existent dans le projet System Manager.
2. Cliquez sur un objet métier pour l'ajouter.
3. Définissez la zone **Agent Support** (décrite plus bas) pour l'objet métier.
4. Dans le menu File de la fenêtre Connector Configurator, cliquez sur **Save to Project**. La définition révisée du connecteur, qui contient la prise en charge sélectionnée pour la définition de l'objet métier ajouté, est enregistrée dans un projet ICL (Integration Component Library) de System Manager.

Pour supprimer un objet métier dans la liste des objets métier pris en charge :

1. Pour sélectionner la zone d'un objet métier, cliquez sur le numéro situé à gauche de l'objet métier.
2. Dans le menu **Edit** de la fenêtre Connector Configurator, cliquez sur **Delete Row**. L'objet métier est supprimé de la liste.
3. Dans le menu **File**, cliquez sur **Save to Project**.

La suppression d'un objet métier dans la liste des objets métier pris en charge modifie la définition du connecteur et rend l'objet métier supprimé inutilisable dans cette implémentation du connecteur. Elle n'affecte pas le code du connecteur et ne supprime pas la définition de l'objet métier dans System Manager.

Prise en charge de l'agent : Si un objet métier dispose de la prise en charge de l'agent, le système tente d'utiliser cet objet métier pour fournir des données à une application via l'agent du connecteur.

En général, les objets métier spécifiques à l'application pour un connecteur sont pris en charge par l'agent de ce connecteur, mais les objets métier génériques ne le sont pas.

Pour indiquer que l'objet métier est pris en charge par l'agent du connecteur, cochez la case **Agent Support**. La fenêtre Connector Configurator ne valide pas vos sélections pour Agent Support.

Niveau de transaction maximum : Le niveau de transaction maximum d'un connecteur correspond au niveau de transaction le plus élevé pris en charge par le connecteur.

Pour la plupart des connecteurs, `Best Effort` est la seule valeur possible.

Vous devez redémarrer le serveur pour que les modifications prennent effet.

Si votre courtier est un courtier de messages WebSphere

Si vous utilisez le mode autonome (non connecté à System Manager), vous devez entrer manuellement le nom de l'objet métier.

Si System Manager est en cours d'exécution, vous pouvez cocher la case située sous la colonne **Business Object Name** dans l'onglet **Supported Business Objects**. Une boîte de dialogue mixte affiche la liste des objets métier disponibles dans le projet Integration Component Library auquel le connecteur appartient. Dans cette liste, sélectionnez l'objet métier de votre choix.

La zone **Message Set ID** est facultative pour WebSphere Business Integration Message Broker 5.0, et sa valeur ne doit pas nécessairement être unique le cas échéant. Cependant, pour WebSphere MQ Integrator et Integrator Broker 2.1, vous devez indiquer un **ID** unique.

Si WAS est votre courtier

Lorsque vous sélectionnez WebSphere Application Server comme type de courtier, Connector Configurator ne nécessite pas les ID d'ensemble de messages. L'onglet **Supported Business Objects** contient la colonne **Business Object Name** pour les objets métier pris en charge uniquement.

Si vous utilisez le mode autonome (non connecté à System Manager), vous devez entrer manuellement le nom de l'objet métier.

Si System Manager est en cours d'exécution, vous pouvez cocher la case située sous la colonne Business Object Name dans l'onglet Supported Business Objects. Une boîte de dialogue mixte affiche la liste des objets métier disponibles dans le projet Integration Component Library auquel le connecteur appartient. Dans cette liste, sélectionnez l'objet métier de votre choix.

Mappes associées (ICS)

Chaque connecteur prend en charge la liste des définitions des objets métier et leurs mappes associées actives dans WebSphere InterChange Server. Cette liste apparaît lorsque vous sélectionnez l'onglet **Associated Maps**.

La liste des objets métier contient l'objet métier spécifique à l'application pris en charge par l'agent et l'objet générique correspondant que le contrôleur envoie à la collaboration de souscription. L'association d'une mappe détermine la mappe qui sera utilisée pour transformer l'objet métier spécifique à l'application en objet métier générique, ou inversement.

Si vous utilisez des mappes uniquement définies pour des objets métier source et cible spécifiques, les mappes sont déjà associées aux objets métier appropriés lorsque vous affichez l'écran, et vous n'avez pas besoin de (ou ne pouvez pas) les modifier.

Si plusieurs mappes sont disponibles pour un objet métier pris en charge, vous devez lier de manière explicite cet objet métier à la mappe qu'il doit utiliser.

L'onglet **Associated Maps** affiche les zones suivantes :

- **Business Object Name**

  Il s'agit des objets métier pris en charge par ce connecteur, comme indiqué dans l'onglet **Supported Business Objects**. Si vous indiquez des objets métier

supplémentaires dans l'onglet Supported Business Objects, ils sont reflétés dans cette liste une fois que vous avez enregistré les modifications en sélectionnant **Save to Project** dans le menu **File** de la fenêtre Connector Configurator.

- **Associated Maps**

  L'écran affiche toutes les mappes installées sur le système à utiliser avec les objets métier pris en charge du connecteur. L'objet métier source pour chaque mappe s'affiche à gauche du nom de la mappe, dans l'écran **Business Object Name**.

- **Liaison explicite**

  Dans certains cas, vous devrez peut-être lier de manière explicite une mappe associée.

  Une liaison explicite est requise uniquement lorsque plusieurs mappes existent pour un objet métier pris en charge spécifique. Lorsque ICS s'amorce, il tente de lier automatiquement une mappe à chaque objet métier pris en charge pour chacun des connecteurs. Si plusieurs mappes prennent le même objet métier comme entrée, le serveur tente de localiser et de lier une mappe qui correspond au sur-ensemble des autres.

  Si aucune mappe n'est le sur-ensemble des autres, le serveur ne peut pas lier l'objet métier à une seule mappe et vous devrez définir la liaison de manière explicite.

  Pour lier une mappe de manière explicite, procédez comme suit :

  1. Dans la colonne **Explicit**, cochez la case correspondant à la mappe à lier.
  2. Sélectionnez la mappe que vous souhaitez associer à l'objet métier.
  3. Dans le menu **File** de la fenêtre Connector Configurator, cliquez sur **Save to Project**.
  4. Déployez le projet jusqu'à ICS.
  5. Réamorcez le serveur pour que les modifications prennent effet.

Ressources (ICS)

L'onglet **Resource** vous permet de définir une valeur qui détermine si l'agent du connecteur gérera plusieurs processus simultanément, et dans quelle mesure, à l'aide du parallélisme de l'agent du connecteur.

Tous les connecteurs ne prennent pas en charge cette fonction. Si vous exécutez un agent de connecteur conçu dans Java pour être multithread, nous vous recommandons de ne pas utiliser cette fonction dans la mesure où il est généralement plus efficace d'utiliser plusieurs unités d'exécution plutôt que plusieurs processus.

Messagerie (ICS)

l'onglet **Messaging** vous permet de configurer les propriétés de messagerie. Les propriétés de messagerie sont disponibles uniquement si vous avez défini MQ comme la valeur de la propriété standard `DeliveryTransport` et ICS comme le type de courtier. Ces propriétés affectent la manière dont le connecteur utilisera les files d'attente.

Validation des files d'attente de messages

Avant de valider une file d'attente de messages, vous devez :

- Vous assurer que WebSphere MQ Series est installé.
- Créer sur la machine hôte une file d'attente de messages avec le canal et le port.
- Configurer une connexion sur la machine hôte.

Pour valider la file d'attente, vous utiliserez le bouton Validate à droite des zones Messaging Type et Host Name, dans l'onglet Messaging.

Sécurité (ICS)

l'onglet **Security** du Connector Configurator permet de définir le niveau de confidentialité d'un message. Cette propriété n'est utilisable que si la propriété DeliveryTransport est définie sur JMS.

Par défaut, Privacy est désactivé. Pour l'activer, cochez la case **Privacy**.

Le **Keystore Target System Absolute Pathname** (Chemin absolu du magasin de clés du système cible) est :

- Pour Windows :
  `<ProductDir>\connectors\security\<connectorname>.jks`
- pour UNIX :
  `opt/IBM/WebSphereAdapters/connectors/security/<connectorname>.jks`

Ce chemin et le fichier qu'il indique doivent être sur le système qui vous servira à démarrer le connecteur, c'est-à-dire le système cible.

Vous pouvez utiliser le bouton Browse à droite, à condition que le système cible soit le seul en cours de fonctionnement. Ce bouton est désactivé jusqu'à ce que **Privacy** soit activé et que le **Target System** de la barre de menus soit défini sur Windows.

Le **Message Privacy Level** peut être défini comme suit pour les trois catégories de messages (All Messages, All Administrative Messages, et All Business Object Messages) :

- La valeur par défaut "" indique qu'aucun niveau de confidentialité n'a été défini pour une catégorie de messages.
- none
  Cette valeur n'a pas le même sens que la valeur par défaut : elle indique le choix délibéré d'attribuer ce niveau de confidentialité à une catégorie de messages.
- integrity
- privacy
- integrity_plus_privacy

La fonction **Key Maintenance** permet de générer, importer et exporter des clés publiques pour le serveur et l'adaptateur.

- La sélection de **Generate Keys** ouvre la boîte de dialogue correspondante, avec les valeurs par défaut pour l'outil qui générera les clés.
- Le magasin de clés est par défaut celui que vous avez indiqué pour **Keystore Target System Absolute Pathname** dans l'onglet Security.
- Lorsque vous cliquez sur OK, les entrées sont validées, le certificat est généré et la sortie est envoyée à la fenêtre de connexion du Connector Configurator.

Avant d'importer un certificat dans le magasin de clés de l'adaptateur, vous devez l'exporter depuis le magasin de clés du serveur. Pour cela, sélectionnez **Export Adapter Public Key**, ce qui ouvre la fenêtre correspondante.

- Par défaut, le certificat exporté prend les valeurs du magasin de clés, à l'exception de son extension qui est <nomdefichier>.cer.

Lorsque vous sélectionnez **Import Server Public Key**, la boîte de dialogue correspondante s'ouvre.

- Par défaut, le certificat importé prend la valeur <*ProductDir*>\bin\ics.cer (si le fichier existe sur le système).
- Le nom de serveur doit être la Certificate Association d'importation. Si un serveur est enregistré, vous pouvez le sélectionner dans la liste déroulante.

La fonction **Adapter Access Control** n'est activée que si DeliveryTransport est définie sur IDL. Par défaut l'adaptateur se connecte avec l'identité d'invité (guest). Si la case **Use guest identity** n'est pas cochée, les zones **Adapter Identity** et **Adapter Password** sont accessibles.

Définition des valeurs du fichier de trace ou du fichier journal

Lorsque vous ouvrez le fichier de configuration ou le fichier de définitions d'un connecteur, Connector Configurator utilise les valeurs de journalisation et de trace de ce fichier comme valeurs par défaut. Vous pouvez modifier ces valeurs dans Connector Configurator.

Pour modifier les valeurs de journalisation et de trace, procédez comme suit :

1. Cliquez sur l'onglet **Trace/Log Files**.
2. Pour la journalisation ou la fonction de trace, vous pouvez écrire des messages à l'un des composants suivants :
   - A la console (STDOUT) :
     Ecrit des messages de journalisation ou de trace à l'écran STDOUT.

     Remarque : Vous pouvez utiliser l'option STDOUT de l'onglet **Trace/Log Files** pour les connecteurs s'exécutant sur la plateforme Windows.
   - A un fichier :
     Ecrit des messages de journalisation ou de trace vers un fichier indiqué. Pour indiquer le fichier, cliquez sur le bouton du répertoire (ellipse), accédez à l'emplacement de votre choix, indiquez un nom de fichier et cliquez sur **Save**. Les messages de journalisation ou de trace sont écrits vers le fichier et l'emplacement indiqués.

     Remarque : Les fichiers de journalisation et de trace sont de simples fichiers texte. Vous pouvez utiliser l'extension de votre choix lorsque vous définissez les noms de fichier. Cependant, pour les fichiers de trace, nous vous recommandons d'utiliser l'extension `.trace` plutôt que l'extension `.trc`, afin d'éviter toute confusion avec les autres fichiers pouvant résider sur le système. Pour les fichiers de journalisation, les extensions classiques sont `.log` et `.txt`.

Gestionnaires de données

La section des gestionnaires de données est disponible pour la configuration uniquement si vous avez indiqué une valeur JMS pour ContainerManagedEvents. Tous les adaptateurs n'utilisent pas les gestionnaires de données.

Pour connaître les valeurs à utiliser pour ces propriétés, reportez-vous aux descriptions sous `ContainerManagedEvents` dans l'Annexe A, Propriétés standard. Pour plus d'informations, voir *Connector Development Guide for C++* ou *Connector Development Guide for Java*.

Enregistrement de votre fichier de configuration

Une fois que vous avez configuré votre connecteur, enregistrez son fichier de configuration. Connector Configurator enregistre le fichier dans le mode courtier que vous avez sélectionné pendant la configuration. La barre de titre de Connector Configurator affiche toujours le mode courtier (ICS, WMQI ou WAS) en cours d'utilisation.

Le fichier est enregistré en tant que document XML. Pour enregistrer le document XML, vous avez trois possibilités :

- dans System Manager, en tant que fichier avec l'extension `*.con` dans le projet ICL ;
- dans un répertoire que vous avez indiqué ;
- en mode autonome, en tant que fichier avec l'extension `*.cfg` dans un répertoire (par défaut, le fichier est enregistré dans `\WebSphereAdapters\bin\Data\App`) ;
- dans un projet WebSphere Application Server, le cas échéant.

Pour plus d'informations sur l'utilisation des projets dans System Manager et sur le déploiement, voir les guides d'implémentation suivants :

- Pour ICS : *Implementation Guide for WebSphere InterChange Server*
- Pour les courtiers de messages WebSphere : *Implementing Adapters with WebSphere Message Brokers*
- Pour WAS : *Implementing Adapters with WebSphere Application Server*

Modification d'un fichier de configuration

Vous pouvez modifier les paramètres du courtier d'intégration pour un fichier de configuration existant. Cela vous permet d'utiliser le fichier comme modèle pour la création d'un nouveau fichier de configuration que vous pouvez utiliser avec un autre courtier.

Remarque : Vous devrez modifier d'autres propriétés de configuration ainsi que la propriété du mode courtier si vous changez de courtiers d'intégration.

Pour modifier votre sélection de courtier dans un fichier de configuration existant (facultatif) :

- Ouvrez le fichier de configuration existant dans Connector Configurator.
- Sélectionnez l'onglet **Standard Properties**.
- Dans la zone **BrokerType** de l'onglet Standard Properties, sélectionnez la valeur appropriée pour votre courtier.
  Lorsque vous modifiez la valeur courante, les onglets disponibles et les sélections de zones de la fenêtre des propriétés changent immédiatement, pour ne montrer que les onglets et zones qui correspondent au courtier que vous avez sélectionné.

Exécution de la configuration

Une fois que vous avez créé un fichier de configuration pour un connecteur et que vous l'avez modifié, assurez-vous que le connecteur peut localiser le fichier de configuration lorsqu'il démarre.

Pour ce faire, ouvrez le fichier de démarrage utilisé pour le connecteur et vérifiez que le nom de fichier et l'emplacement utilisés pour le fichier de configuration du connecteur correspondent exactement au nom attribué au fichier et au répertoire ou au chemin d'accès dans lequel vous l'avez placé.

Utilisation de Connector Configurator dans un environnement globalisé

Connector Configurator est globalisé et peut gérer la conversion des caractères entre le fichier de configuration et le courtier d'intégration. Connector Configurator utilise le codage natif. Lorsqu'il écrit dans le fichier de configuration, il utilise le codage UTF-8.

Connector Configurator prend en charge les caractères qui n'existent pas en anglais dans :

- toutes les zones de valeur ;
- le chemin d'accès au fichier journal et au fichier de trace (indiqué dans l'onglet **Trace/Log files**).

La liste déroulante pour les propriétés de configuration standard `CharacterEncoding` et `Locale` affiche uniquement un sous-ensemble des valeurs prises en charge. Pour ajouter d'autres valeurs à cette liste, vous devez modifier manuellement le fichier `\Data\Std\stdConnProps.xml` dans le répertoire produit.

Par exemple, pour ajouter l'environnement local `en_GB` à la liste des valeurs pour la propriété `Locale`, ouvrez le fichier `stdConnProps.xml` et ajoutez la ligne en caractère gras comme indiqué ci-dessous :

```
<Property name="Locale"
isRequired="true"
updateMethod="component restart">
                <ValidType>String</ValidType>
            <ValidValues>
                                <Value>ja_JP</Value>
                                <Value>ko_KR</Value>
                                <Value>zh_CN</Value>
                                <Value>zh_TW</Value>
                                <Value>fr_FR</Value>
                                <Value>de_DE</Value>
                                <Value>it_IT</Value>
                                <Value>es_ES</Value>
                                <Value>pt_BR</Value>
                                <Value>en_US</Value>
                                <Value>en_GB</Value>
                    <DefaultValue>en_US</DefaultValue>
            </ValidValues>
    </Property>
```

Annexe C. Application Response Measurement

Cet adaptateur est compatible avec l'API Application Response Measurement, qui permet de gérer la disponibilité, les accords de niveau de service et la planification de la capacité des applications. Une application avec appels ARM peut participer à IBM Tivoli Monitoring for Transaction Performance, autorisant ainsi la collecte et la consultation des données relatives aux indicateurs de transaction.

Prise en charge des appels Application Response Measurement

Cet adaptateur est compatible avec l'API Application Response Measurement, qui permet de gérer la disponibilité, les accords de niveau de service et la planification de la capacité des applications. Une application avec appels ARM peut participer à IBM Tivoli Monitoring for Transaction Performance, autorisant ainsi la collecte et la consultation des données relatives aux indicateurs de transaction.

Logiciels requis

En plus des logiciels exigés par l'adaptateur, les applications suivantes doivent être installées pour qu'ARM puisse fonctionner :

- WebSphere Application Server 5.0.1 (contient le serveur IBM Tivoli Monitoring for Transaction Performance). Il ne doit pas forcément être installé sur le même système que l'adaptateur.
- IBM Tivoli Monitoring for Transaction Performance v. 5.2 Fixpack 1. Il doit être installé sur le même système que l'adaptateur, et configuré pour pointer vers la machine qui héberge le serveur IBM Tivoli Monitoring for Transaction Performance.

La prise en charge de Application Response Measurement est possible avec tout courtier d'intégration supporté par cette édition.

Remarque : Les appels Application Response Measurement sont pris en charge par tous les systèmes d'exploitation supportés par la présente édition de IBM WebSphere Business Integration Adapters, *sauf* par HP-UX (toutes versions) et Red Hat Linux 3.0.

Activation de Application Response Measurement

Les appels ARM sont activés en définissant sur "True" la propriété standard `TivoliMonitorTransactionPerformance` de Connector Configurator. Par défaut, la prise en charge ARM est désactivée. (Pour plus d'informations, voir l'annexe "Propriétés standard" de ce document.)

Surveillance des transactions

Lorsque ARM est activé, les transactions surveillées sont les demandes de service et les livraisons d'événements. La transaction est mesurée à partir du début de la demande de service ou de la livraison d'événement et jusqu'à la fin. Le nom de la transaction, affiché sur la console Tivoli Monitoring for Transaction Performance, commencera par `SERVICE REQUEST` ou `EVENT DELIVERY`. La suite du nom sera l'instruction de l'objet métier (comme `CREATE`, `RETRIEVE`, `UPDATE` ou `DELETE`). La fin du nom sera le nom de l'objet métier, comme "`EMPLOYEE`." Par exemple, le nom d'une transaction pour une livraison d'événement correspondant à la création d'un employé pourra être `EVENT DELIVERY CREATE EMPLOYEE` ou `SERVICE REQUEST UPDATE ORDER`.

Les indicateurs ci-après sont collectés par défaut pour chaque type de demande de service ou de livraison d'événement :

- Durée minimale de transaction
- Durée maximale de transaction
- Durée moyenne de transaction
- Total d'exécutions de la transaction

Vous (ou l'administrateur système de WebSphere Application Server) pouvez sélectionner les indicateurs à afficher, et pour quels événements d'adaptateur, en configurant Discovery Policies et Listener Policies pour des transactions données, depuis la console Tivoli Monitoring for Transaction Performance. (Voir «Pour plus d'informations».)

Pour plus d'informations

Pour plus de détails, voir la documentation IBM Tivoli Monitoring for Transaction Performance. Pour obtenir des informations sur la surveillance et la gestion des indicateurs générés par l'adaptateur, voir *IBM Tivoli Monitoring for Transaction Performance User's Guide*.

Annexe D. Common Event Infrastructure

WebSphere Business Integration Server Foundation inclut Common Event Infrastructure Server Application, nécessaire au fonctionnement de Common Event Infrastructure. WebSphere Application Server Foundation peut être installé sur tout système (pas nécessairement sur la même machine que l'adaptateur.)

WebSphere Application Server Application Client contient les bibliothèques nécessaires à l'interaction entre l'adaptateur et Common Event Infrastructure Server Application. Vous devez installer WebSphere Application Server Application Client sur le même système que l'adaptateur. L'adaptateur se connecte à WebSphere Application Server (dans WebSphere Business Integration Server Foundation) par le biais d'une URL configurable.

La prise en charge de Common Event Infrastructure est possible avec tout courtier d'intégration supporté par cette édition.

Logiciels requis

En plus des logiciels exigés pour l'adaptateur, les applications suivantes doivent être installées pour que Common Event Infrastructure puisse fonctionner :

- WebSphere Business Integration Server Foundation 5.1.1
- WebSphere Application Server Application Client 5.0.2, 5.1 ou 5.1.1.

  (WebSphere Application Server Application Client 5.1.1 est fourni avec WebSphere Business Integration Server Foundation 5.1.1.)

Remarque : Common Event Infrastructure n'est pris en charge par aucune plateforme HP-UX ou Linux.

Activation de Common Event Infrastructure

La fonctionnalité Common Event Infrastructure est activée avec les propriétés standard `CommonEventInfrastructure` et `CommonEventInfrastructureContextURL`, configurées avec Connector Configurator. Par défaut, Common Event Infrastructure n'est pas activé. La propriété `CommonEventInfrastructureContextURL` vous permet de configurer l'URL du serveur Common Event Infrastructure. (Pour plus d'informations, voir l'annexe "Propriétés standard" du présent document.)

Obtention d'événements d'adaptateur Common Event Infrastructure

Si Common Event Infrastructure est activé, l'adaptateur génère des événements Common Event Infrastructure qui se mappent sur les événements d'adaptateur suivants :

- Démarrage de l'adaptateur
- Arrêt de l'adaptateur
- Réponse de l'application à un dépassement de délai de l'agent de l'adaptateur
- Tout appel `doVerbFor` émis depuis l'agent de l'adaptateur
- Appel `gotApplEvent` depuis l'agent de l'adaptateur

Pour qu'une autre application (l'"application client") puisse recevoir les événements Common Event Infrastructure générés par l'adaptateur, elle doit

utiliser le catalogue d'événements Common Event Infrastructure pour déterminer les définitions des événements appropriés et leurs propriétés. Pour que l'application client puisse utiliser les événements de l'application d'origine, les événements doivent être définis dans le catalogue d'événements.

L'annexe "Définitions du catalogue des événements Common Event Infrastructure" du présent document contient des métadonnées au format XML. Elles indiquent, pour les adaptateurs WebSphere Business Information, les descripteurs d'événements et les propriétés que l'application client doit rechercher.

Pour plus d'informations

Pour plus d'informations sur Common Event Infrastructure, voir la documentation WebSphere Business Integration Server Foundation, disponible à l'adresse :

http://publib.boulder.ibm.com/infocenter/ws51help

Pour consulter des exemples de métadonnées XML indiquant les propriétés et descripteurs d'événement générés par l'adaptateur qu'une application client doit utiliser, voir «Définitions du catalogue d'événements Common Event Infrastructure».

Définitions du catalogue d'événements Common Event Infrastructure

Le catalogue d'événements Common Event Infrastructure contient des définitions d'événements qui peuvent être interrogées par d'autres applications. Ci-dessous figurent des exemples de définitions d'événements utilisant des métadonnées XML pour des événements d'adaptateur types. Si vous écrivez une autre application, elle peut utiliser les interfaces de catalogue d'événements pour interroger la définition de l'événement. Pour plus d'informations sur les définitions d'événements et savoir comment les interroger, voir la documentation Common Event Infrastructure disponible dans le centre d'information en ligne IBM WebSphere Server Foundation.

Pour les adaptateurs WebSphere Business Integration, les éléments de données étendues qui doivent être définis dans le catalogue des événements sont les clés de l'objet métier. Chaque clé d'objet métier exige une définition d'événement. Par conséquent, pour un adaptateur donné, les divers événements tels que start adapter, stop adapter, timeout adapter et tout événement`doVerbFor` (comme create, update ou delete) doivent avoir une définition dans le catalogue d'événements.

Les sections suivantes contiennent des exemples des métadonnées pour start adapter, stop adapter et la requête ou la livraison d'événement.

Format XML des métadonnées de "start adapter"

```
<eventDefinition name="startADAPTER"
      parent="event">
    <property name ="creationTime" //Commentaire : Par exemple, la valeur peut être
 "2004-05-13T17:00:16.319Z"
         required="true" />
    <property name="globalInstanceId" //Commentaire : Généré automatiquement
 par Common Event Infrastructure
         required="true"/>
    <property name="sequenceNumber"    //Commentaire : Numéro défini par la source
pour les messages qui seront envoyés/triés de façon logique
         required="false"/>
    <property name="version"    //Commentaire : Version de l'événement
```

```
         required="false"
         defaultValue="1.0.1"/>
    <property name="sourceComponentId"
         path="sourceComponentId"
         required="true"/>
    <property name="application"   //Commentaire : Le name#version de
l'application source générant l'événement. Par exemple,  "SampleConnector#3.0.0"
         path="sourceComponentId/application"         required="false"/>
    <property name="component"   //Commentaire : Ce sera le name#version
 du composant source.
         path="sourceComponentId/component"
         required="true"
         defaultValue="ConnectorFrameWorkVersion#4.2.2"/>
    <property name="componentIdType"    //Commentaire : Précise le format
et la signification du composant
          path="sourceComponentId/componentIdType"
          required="true"
          defaultValue="Application"/>
    <property name="executionEnvironment"
 //Commentaire : Identifie l'environnement dans lequel l'application est exécutée
 ...par exemple "Windows 2000#5.0"
          path="sourceComponentId/executionEnvironment"
          required="false" />
     <property name="location"    //Commentaire : Cette valeur correspond au
 nom du serveur... par exemple "WQMI"
          path="sourceComponentId/location"
          required="true"/>
    <property name="locationType"   //Commentaire : Indique le format et
     la signification de l'emplacement
          path="sourceComponentId/locationType"
          required="true"
          defaultValue="Hostname"/>
     <property name="subComponent"     //Commentaire : Distinction supplémentaire
du composant logique
          path="sourceComponentId/subComponent"
          required="true"
          defaultValue="AppSide_Connector.AgentBusinessObjectManager"/>
     <property name="componentType"      //Commentaire : Nom correctement défini
utilisé pour caractériser toutes les instances de ce composant
          path="sourceComponentId/componentType"
          required="true"
          defaultValue="ADAPTER"/>
     <property name="situation"   //Commentaire : Définit le type de
 situation ayant entraîné le signalement de l'événement
          path="situation"
          required="true"/>
     <property name="categoryName="    //Commentaire : Indique le type
de situation pour l'événement
         path="situation/categoryName"
         required="true"
         defaultValue="StartSituation"/>
    <property name="situationType"    //Commentaire : Indique le type
de situation et la disposition de l'événement
         path="situation/situationType"
         required="true"
    <property name="reasoningScope" //Commentaire : Indique
 l'impact de l'événement
         path="situation/situationType/reasoningScope"
         required="true"
         permittedValue="INTERNAL"
         permittedValue="EXTERNAL"/>
    <property name="successDisposition" //Commentaire : Indique la
 réussite de l'événement
         path="situation/situationType/successDisposition"
         required="true"
         permittedValue="SUCCESSFUL"
         permittedValue="UNSUCCESSFUL" />
```

```
    <property name="situationQualifier"  //Commentaire : Indique les
 qualificatifs de la situation pour cet événement
         path="situation/situationType/situationQualifier"
         required="true"
         permittedValue="START_INITIATED"
         permittedValue="RESTART_INITIATED"
         permittedValue="START_COMPLETED" />
</eventDefinition>
```

Format XML des métadonnées de ″stop adapter″

Les métadonnées de “stop adapter” sont identiques à celles de “start adapter” aux exceptions suivantes près :

- La valeur par défaut de la propriété `categoryName` est `StopSituation` :

```
<property name="categoryName="
 //Commentaire : Indique le type
 de situation pour l'événement
              path="situation/categoryName"
              required="true"
              defaultValue="StopSituation"/>
```

- Les valeurs autorisées pour la propriété `situationQualifier` sont différentes et sont les suivantes pour “stop adapter” :

```
<property name="situationQualifier"
 //Commentaire : Précise les qualificatifs de la situation pour cet événement
           path="situation/situationType/situationQualifier"
           required="true"
           permittedValue="STOP_INITIATED"
           permittedValue="ABORT_INITIATED"
           permittedValue="PAUSE_INITIATED"
           permittedValue="STOP_COMPLETED"
 />
```

Format XML des métadonnées de “timeout adapter”

Les métadonnées de “timeout adapter” sont identiques à celles de “start adapter” et “stop adapter” aux exceptions suivantes près :

- La valeur par défaut de la propriété `categoryName` est `ConnectSituation` :

```
<property name="categoryName="
 //Commentaire : Indique le type
 de situation pour l'événement
              path="situation/categoryName"
              required="true"
              defaultValue="ConnectSituation"/>
```

- Les valeurs autorisées pour la propriété `situationQualifier` sont différentes et sont les suivantes pour “timeout adapter” :

```
<property name="situationQualifier"  //Commentaire : Précise
 les qualificatifs de la situation pour cet événement
           path="situation/situationType/situationQualifier"
           required="true"
           permittedValue="IN_USE"
           permittedValue="FREED"
           permittedValue="CLOSED"
           permittedValue="AVAILABLE"
 />
```

Format XML des métadonnées de ″request″ ou ″delivery″

A la fin de ce format XML figurent les éléments de données étendues. Les éléments de données étendues des événements de livraison et de requête d'adaptateur sont les données de l'objet métier en cours de traitement. Ces données incluent le nom de l'objet métier, sa clé (étrangère ou locale) et les objets métier qui sont des enfants d'objets métier parents. Les objets métier enfants sont ensuite décomposés dans les mêmes données que le parent (nom, clé et tout objet métier enfant). Ces données sont représentées dans un élément de données étendues de la définition de l'événement. Ces données changeront selon l'objet métier, les clés et les objets métier enfants traités. Les données étendues de cette définition d'événement sont un simple exemple et représentent un objet métier nommé `Employee` avec une clé `EmployeeId` et un objet métier enfant `EmployeeAddress` avec une clé `EmployeeId`. Ce modèle peut s'appliquer à toutes les données d'un objet métier particulier.

```
<eventDefinition name="createEmployee"     //Commentaire : Ce
 nom d'extension est toujours l'instruction de l'objet métier suivi de
 son nom
          parent="event">
    <property name ="creationTime"  //Commentaire : Par exemple, la valeur peut être
"2004-05-13T17:00:16.319Z"
         required="true" />
    <property name="globalInstanceId" //Commentaire : Généré automatiquement
 par Common Event Infrastructure
         required="true"/>
    <property name="localInstanceId"    //Commentaire : La valeur est égale à
 l'instruction de l'objet métier+nom de l'objet métier+#+nom app+ identificateur de
 l'objet métier         required="false"/>
    <property name="sequenceNumber"    //Commentaire : Numéro défini par la source
pour les messages qui seront envoyés/triés de façon logique
         required="false"/>
    <property name="version"  //Commentaire : Version de l'événement... la valeur est
 définie sur 1.0.1
         required="false"
         defaultValue="1.0.1"/>
    <property name="sourceComponentId"
         path="sourceComponentId"
         required="true"/>
    <property name="application"   //Commentaire : Le name#version de
 l'application source qui génère l'événement... par exemple
 "SampleConnector#3.0.0"
         path="sourceComponentId/application"
         required="false"/>
    <property name="component"   //Commentaire : Ce sera le name#version
du composant source.
         path="sourceComponentId/component"
         required="true"
         defaultValue="ConnectorFrameWorkVersion#4.2.2"/>
    <property name="componentIdType"    //Commentaire : Précise le format
 et la signification du composant
         path="sourceComponentId/componentIdType"
         required="true"
         defaultValue="Application"/>
    <property name="executionEnvironment" //Commentaire : Identifie le
 environment#version dans lequel l'application est exécutée...
 par exemple "Windows 2000#5.0"
         path="sourceComponentId/executionEnvironment"
         required="false" />
    <property name="instanceId" //Commentaire : La valeur est égale à l'instruction
  de l'objet métier+nom de l'objet métier+#+nom app+identificateur de l'objet métier
          path="sourceComponentId/instanceId"
          required="false"
    <property name="location"    //Commentaire : Cette valeur correspond au
nom du serveur... par exemple "WQMI"
          path="sourceComponentId/location"
```

```
          required="true"/>
     <property name="locationType"   //Commentaire : Indique le format et
 la signification de l'emplacement
          path="sourceComponentId/locationType"
          required="true"
          defaultValue="Hostname"/>
     <property name="subComponent"  //Commentaire : Distinction supplémentaire du
 composant logique (dans ce cas, la valeur est égale au nom de l'objet
 métier)
          path="sourceComponentId/subComponent"
          required="true"/>
     <property name="componentType"      //Commentaire : Nom correctement défini utilisé
 pour caractériser toutes les instances de ce composant
          path="sourceComponentId/componentType"
          required="true"
          defaultValue="ADAPTER"/>
     <property name="situation"   //Commentaire : Définit le type de
situation ayant entraîné le signalement de l'événement
          path="situation"
          required="true"/>
    <property name="categoryName"    //Commentaire : Indique le type
 de situation pour l'événement
         path="situation/categoryName"
         required="true"
         permittedValue="CreateSituation"
         permittedValue="DestroySituation"
         permittedValue="OtherSituation" />
    <property name="situationType"    //Commentaire : Indique le type
de situation et la disposition de l'événement
         path="situation/situationType"
         required="true"
    <property name="reasoningScope" //Commentaire : Indique
l'impact de l'événement
         path="situation/situationType/reasoningScope"
         required="true"
         permittedValue="INTERNAL"
         permittedValue="EXTERNAL"/>
    <property name="successDisposition" //Commentaire : Indique la
 réussite de l'événement
         path="situation/situationType/successDisposition"
         required="true"
         permittedValue="SUCCESSFUL"
         permittedValue="UNSUCCESSFUL" />
    <extendedDataElements name="Employee" //Commentaire : Nom de l'objet métier
 lui-même
             type="noValue"
             <children name="EmployeeId"
                  type="string"/>  //Commentaire : Le type est l'une des
 valeurs autorisées dans la documentation Common Event Infrastructure
             <children name="EmployeeAddress"
                  type="noValue"/>
                    <children name="EmployeeId"
                        type="string"/>
                     -
                     -
                     -
    </extendedDataElements
</eventDefinition>
```

Index

Informations légales

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 d'échange IBM. Toute référence à un produit, logiciel ou service IBM n'implique pas que seul ce produit, logiciel ou service puisse être utilisé. Tout autre élément fonctionnellement équivalent peut être utilisé, s'il n'enfreint aucun droit d'IBM. Il est de la responsabilité de l'utilisateur d'évaluer et de vérifier lui-même les installations et applications réalisées avec des produits, logiciels ou services non expressément référencés par IBM.

IBM peut détenir des brevets ou des demandes de brevet couvrant les produits mentionnés dans le présent document. La remise de ce document ne vous donne aucun droit de licence sur ces brevets ou demandes de brevet. Si vous désirez recevoir des informations concernant l'acquisition de licences, veuillez en faire la demande par écrit à l'adresse suivante :

IBM EMEA Director of Licensing
IBM Europe Middle-East Africa
Tour Descartes
La Défense 5
2, avenue Gambetta
92066 - Paris-La Défense CEDEX
France

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 à l'adresse suivante :

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032,
Japan

Le paragraphe suivant ne s'applique ni au Royaume-Uni, ni dans aucun pays dans lequel il serait contraire aux lois locales.

LE PRESENT DOCUMENT EST LIVRE EN L'ETAT, 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, 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
577 Airport Blvd., Suite 800
Burlingame, CA 94010
U.S.A.

Ces informations peuvent être soumises à des conditions particulières, prévoyant notamment le paiement d'une redevance.

Le logiciel sous licence décrit dans ce document et tous les éléments sous licence disponibles s'y rapportant sont fournis par IBM conformément aux dispositions de l'ICA, des Conditions internationales d'utilisation des logiciels IBM ou de tout autre accord équivalent.

Les données de performance indiquées dans ce document ont été déterminées dans un environnement contrôlé. Par conséquent, les résultats peuvent varier de manière significative selon l'environnement d'exploitation utilisé. Certaines mesures évaluées sur des systèmes en cours de développement ne sont pas garanties sur tous les systèmes disponibles. En outre, elles peuvent résulter d'extrapolations. Les résultats peuvent donc varier. Il incombe aux utilisateurs de ce document de vérifier si ces données sont applicables à leur environnement d'exploitation.

Les informations concernant des produits non IBM ont été obtenues auprès des fournisseurs de ces produits, par l'intermédiaire d'annonces publiques ou via d'autres sources disponibles. IBM n'a pas testé ces produits et ne peut confirmer l'exactitude de leurs performances ni leur compatibilité. Elle ne peut recevoir aucune réclamation concernant des produits non IBM. Toute question concernant les performances de produits non IBM doit être adressée aux fournisseurs de ces produits.

Toute instruction relative aux intentions d'IBM pour ses opérations à venir est susceptible d'être modifiée ou annulée sans préavis, et doit être considérée uniquement comme un objectif.

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.

LICENCE DE COPYRIGHT :

Le présent logiciel contient des exemples de programmes en langage source, destinés à illustrer les techniques de programmation sur différentes platesformes d'exploitation. Vous avez le droit de copier, de modifier et de distribuer ces exemples de programmes sous quelque forme que ce soit et sans paiement d'aucune redevance à IBM, à des fins de développement, d'utilisation, de vente ou de distribution de programmes d'application conformes aux interfaces de programmation des plateformes pour lesquels ils ont été écrits ou aux interfaces de programmation IBM. Ces exemples de programmes n'ont pas été rigoureusement testés dans toutes les conditions. Par conséquent, IBM ne peut garantir expressément ou implicitement la fiabilité, la maintenabilité ou le fonctionnement de ces programmes.

Si vous visualisez ces informations en ligne, il se peut que les photographies et illustrations en couleur n'apparaissent pas à l'écran.

Informations sur les interfaces de programmation

Les informations sur les interfaces de programmation ont pour objectif de vous aider à créer des logiciels d'application à l'aide de ce programme.

Les interfaces de programmation génériques vous permettent de créer des logiciels d'application qui obtiennent les services des outils de ce programme.

Cependant, ces informations peuvent également contenir des informations sur le diagnostic, la modification et le réglage. Ces informations vous permettent d'exécuter le débogage de votre logiciel d'application.

Avertissement : N'utilisez pas ces informations sur le diagnostic, la modification et le réglage comme interface de programmation car elles sont susceptibles de changer.

Marques et marques de service

Les termes qui suivent sont des marques d'International Business Machines Corporation aux Etats-Unis et/ou dans certains autres pays :

i5/OS
IBM
le logo IBM
AIX
AIX 5L
CICS
CrossWorlds
DB2
DB2 Universal Database
Domino
HelpNow
IMS
Informix
iSeries
Lotus

Lotus Notes
MQIntegrator
MQSeries
MVS
Notes
OS/400
Passport Advantage
pSeries
Redbooks
SupportPac
WebSphere
z/OS

Java et toutes les marques incluant Java sont des marques de Sun Microsystems, Inc. aux Etats-Unis et/ou 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.

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 ou marques déposées de Intel Corporation ou de ses filiales, aux Etats-Unis et dans d'autres pays.

UNIX est une marque enregistrée de The Open Goup aux Etats-Unis et/ou dans certains autres pays.

Linux est une marque de Linus Torvalds aux Etats-Unis et/ou dans certains autres pays.

Les autres noms de sociétés, de produits et de services peuvent appartenir à des tiers.

Ce produit inclut un logiciel développé par Eclipse Project (http://www.eclipse.org/).

WebSphere Business Integration Adapter Framework, version 2.6.0.3