WebSphere Business Integration Adapters

Adapter for JMS - Guide d'utilisation

Adapter Version 2.8.x

WebSphere Business Integration Adapters

Adapter for JMS - Guide d'utilisation

Adapter Version 2.8.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 117.

Remarque

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

LE PRESENT DOCUMENT EST LIVRE ″EN L'ETAT″. IBM DECLINE TOUTE RESPONSABILITE, EXPRESSE OU IMPLICITE, RELATIVE AUX INFORMATIONS QUI Y SONT CONTENUES, Y COMPRIS EN CE QUI CONCERNE LES GARANTIES DE QUALITE MARCHANDE OU D'ADAPTATION A VOS BESOINS. Certaines juridictions n'autorisent pas l'exclusion des garanties implicites, auquel cas l'exclusion ci-dessus ne vous sera pas applicable.

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 v

A propos de ce document vii
Public visé vii
Prérequis pour ce document. vii
Documents connexes vii
Conventions typographiques viii

Nouveautés de la présente édition xi
Nouveautés de l'édition 2.8.x xi
Nouveautés de l'édition 2.7 xi
Nouveautés de l'édition 2.6 xii
Nouveautés de l'édition 2.5 xii
Nouveautés de l'édition 2.4.x xii
Nouveautés de l'édition 2.3.x xii
Nouveautés de l'édition 2.2.x xiii
Nouveautés de l'édition 2.1.x xiii
Nouveautés de l'édition 1.3.x xiii
Nouveautés de l'édition 1.2.x xiv
Nouveautés de l'édition 1.1.x xiv

Chapitre 1. Présentation de Adapter for JMS 1
Environnement de l'adaptateur pour JMS. 1
Terminologie de l'adaptateur pour JMS 4
Architecture de connecteur pour JMS 4
Traitement de message 4

Chapitre 2. Installation et configuration de l'adaptateur 17
Tâches d'installation 17
Installation de l'adaptateur et des fichiers associés 17
Structure de fichiers installée 17
Configuration du connecteur 19
Configuration des propriétés du connecteur 19
Configuration des méta-objets 31
Configuration des scripts de démarrage 43
Création de plusieurs instances de connecteur 43
Démarrage du connecteur 44
Arrêt du connecteur 46

Chapitre 3. Création ou modification d'objets métier 47
Structure des objets métier de connecteur 47

Chapitre 4. Résolution des incidents 49
Gestion des erreurs 49
Traçage 50
Résolution des incidents de démarrage 51

Annexe A. Propriétés de configuration standard pour les connecteurs 53
Nouvelles propriétés 53
Présentation des propriétés de connecteur standard 53
Référence rapide des propriétés standard 55
Propriétés standard 61

Annexe B. Connector Configurator 79
Présentation de Connector Configurator 79
Démarrage de Connector Configurator 80
Exécution de Connector Configurator à partir de System Manager 81
Création d'un modèle de propriétés spécifiques au connecteur 81
Création d'un fichier de configuration 84
Utilisation d'un fichier existant 86
Remplissage d'un fichier de configuration 87
Définition des propriétés d'un fichier de configuration 87
Enregistrement de votre fichier de configuration 96
Modification d'un fichier de configuration 96
Exécution de la configuration 97
Utilisation de Connector Configurator dans un environnement globalisé 97

Annexe C. Didacticiel 99
Présentation du didacticiel 99
Configuration de votre environnement 100
Exécution des scénarios 102

Annexe D. Configuration des messageries basées sur des rubriques et sur des files d'attente 105
Configuration pour la messagerie basée sur des files d'attente 105
Configuration de la messagerie basée sur des rubriques 106

Annexe E. Common Event Infrastructure 107
Logiciels requis 107
Activation de Common Event Infrastructure 107
Obtention d'événements d'adaptateur Common Event Infrastructure 107
Pour plus d'informations 108
Définitions du catalogue d'événements Common Event Infrastructure 108
Format XML des métadonnées de "start adapter" 108
Format XML des métadonnées de "stop adapter" 110
Format XML des métadonnées de "timeout adapter" 110
Format XML des métadonnées de "request" ou "delivery" 111

Annexe F. Application Response Measurement 113
Prise en charge des appels Application Response Measurement 113

Index 115

Informations légales 117
Informations sur les interfaces de programmation 119
Marques et marques de service 119

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 existants et 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.

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

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, développeurs et administrateurs système qui prennent en charge et gèrent le système d'intégration WebSphere sur des sites clients.

Prérequis pour ce document

Les utilisateurs doivent bien connaître le système d'intégration WebSphere, le développement des objets métier et collaborations et l'application JMS. Pour consulter des liens, voir «Documents connexes».

Documents connexes

La documentation complète qui accompagne ce produit présente les caractéristiques et les fonctions communes à toutes les installations d'adaptateur WebSphere, 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 un adaptateur, pour apprendre à utiliser des adaptateurs avec des courtiers de messages WebSphere (WebSphere MQ Integrator, WebSphere MQ Integrator Broker, WebSphere Business Integration Message Broker) et utiliser des adaptateurs avec WebSphere Application Server :
  - http://www.ibm.com/websphere/integration/wbiadapters/infocenter
- Pour utiliser des adaptateurs avec InterChange Server:
  - http://www.ibm.com/websphere/integration/wicserver/infocenter

– http://www.ibm.com/websphere/integration/wbicollaborations/infocenter

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

Remarque : Des informations importantes relatives à ce produit peuvent être disponibles dans les flashes de support technique (Technical Support Flashes), après la publication de ce document. Elles sont disponibles sur le site Web de WebSphere Business Integration Support Web, à l'adresse http://www.ibm.com/software/integration/websphere/support/. Sélectionnez la zone qui vous intéresse et parcourez les sections Technotes et Flashes. Des informations complémentaires sont également disponibles sur IBM Redbooks à l'adresse http://www.redbooks.ibm.com/.

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, *`italique`*	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`.
ProductDir	Correspond au répertoire où le produit est installé.
/, \	Dans ce document, les barres obliques inverses (\) sont utilisées comme convention pour les chemins de répertoire. 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.
UNIX:/Windows:	Les paragraphes qui commencent par l'un de ces éléments indiquent des remarques sur les différences qui existent entre les systèmes d'exploitation.
u	Ce symbole indique la fin d'un paragraphe **UNIX/Windows**. Il indique également la fin d'une note à plusieurs paragraphes.

%texte% et *$texte*	Du texte placé entre des signes pourcentage (%) indique la valeur de la variable système `texte` Windows ou la variable utilisateur. Le symbole équivalent dans un environnement UNIX est *$texte*, indiquant la valeur de la variable d'environnement UNIX *texte*.

Nouveautés de la présente édition

Nouveautés de l'édition 2.8.x

Mise à jour de septembre 2005. L'édition de ce document pour la version 2.7.x de l'adaptateur contient les nouveautés ou modifications suivantes.

Prise en charge supplémentaire de AIX 5.3

Prise en charge supplémentaire de la gestion de l'adaptateur JMS par IBM Tivoli License Manager (ITLM)

Prise en charge du texte bidirectionnel

Prise en charge étendue de plusieurs unités d'exécution pour l'interrogation d'événement, par le biais de la propriété facultative de configuration de connecteur WorkerThreadCount

Prise en charge étendue de messages objets. L'adaptateur prend désormais en charge les messages de type objet Java.

Appel optimisé du gestionnaire de données via la propriété facultative de configurateur de connecteur DataHandlerPoolSize

Formatage facultatif permettant de préciser les propriétés de connecteur ConnectionFactoryName et ArchivalConnectionFactoryName

Nouveautés de l'édition 2.7

Mise à jour de septembre 2004. L'édition de ce document pour la version 2.7.x de l'adaptateur contient les nouveautés ou modifications suivantes.

Cette édition ajoute la prise en charge des plateformes suivantes :

- Solaris 9 : cet adaptateur prend en charge une JVM 32 bits sur une plateforme 64 bits
- Pour AIX 5.1 et 5.2, déjà pris en charge : cet adaptateur prend en charge le JVM 32 bits sur plateforme 64 bits
- Microsoft Windows 2003
- Linux RedHat AS 3.0, ES 3.0 et WS 3.0
- SUSE Linux Standard Server 8.1 et Enterprise Server 8.1 SP3
- IBM JRE/JDK 1.4.2

Cette édition prend en charge le niveau de trace 5 pour afficher la trace printStackTrace() sur des exceptions levées par l'adaptateur.

Nouveautés de l'édition 2.6

Trois propriétés spécifiques au connecteur ont été ajoutées : EnableMessageProducerCache, SessionPoolSizeForRequests et ArchivalConnectionFactoryName. Pour plus d'informations, voir «Configuration des propriétés spécifiques au connecteur», à la page 20.

A partir de la version 2.6.x, l'adaptateur n'étant pas pris en charge sous Solaris 7, les références à cette plateforme ont été supprimées de ce guide.

Nouveautés de l'édition 2.5

L'adaptateur peut désormais utiliser WebSphere Integration Message Broker comme courtier d'intégration. Pour plus d'informations, voir «Compatibilité du courtier», à la page 2.

A partir de la version 2.5, l'adaptateur pour JMS n'est pas pris en charge sous Microsoft Windows NT.

Les informations relatives à l'installation de l'adaptateur ont été déplacées. Voir le chapitre 2 pour savoir où sont ces informations.

L'adaptateur prend désormais en charge la messagerie de publication/souscription (basée sur les rubriques) définie par le standard JMS, ainsi que l'interface de messagerie point à point (basée sur les files d'attente). Une même instance de l'adaptateur n'accepte qu'un seul type de messagerie. Rubriques et files d'attente ne peuvent être précisées dans la même configuration. Toutefois, les deux types de messagerie peuvent être pris en charge en exécutant plusieurs instances de l'adaptateur, dont certaines implémentent la messagerie basée sur les rubriques et d'autre la messagerie basée sur des files d'attente.

Nouveautés de l'édition 2.4.x

L'adaptateur peut désormais utiliser WebSphere Application Server comme courtier d'intégration. Pour plus d'informations, voir «Compatibilité du courtier», à la page 2.

Le connecteur s'exécute désormais sur les plateformes suivantes :

- Microsoft Windows NT 4.0 Service Pack 6A ou Windows 2000
- Solaris 7, 8 ou AIX 5.1, 5.2 ou HP UX 11.i

Nouveautés de l'édition 2.3.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 à présent "System Manager," et "CrossWorlds InterChange Server" est devenu "WebSphere InterChange Server."

Vous pouvez maintenant associer un gestionnaire de données à une file d'attente de données. Pour plus d'informations, voir «Mappage des gestionnaires de données sur les destinations d'entrée», à la page 37.

La fonctionnalité de livraison garantie d'événement a été étendue. Pour plus d'informations, voir *Connector Development Guide for Java*.

Nouveautés de l'édition 2.2.x

La file d'attente InProgress n'est plus obligatoire et peut être désactivée. Pour plus d'informations, voir «InProgressDestination», à la page 27.

La propriété ReplyToQueue peut à présent être déterminée par le méta-objet enfant dynamique, plutôt que par la propriété de connecteur ReplyToQueue. Pour plus d'informations, voir «En-têtes JMS et attributs de méta-objet enfant dynamique», à la page 41.

Vous pouvez utiliser un sélecteur de message pour identifier, filtrer et contrôler la façon dont l'adaptateur identifie le message de réponse pour une requête données. Cette capacité JMS s'applique uniquement au traitement de requête synchrone. Pour plus d'informations, voir «Traitement synchrone», à la page 13.

Nouveautés de l'édition 2.1.x

Le connecteur a été internationalisé. Pour plus d'informations, voir «Données dépendant des paramètres nationaux», à la page 3 et Annexe A, «Propriétés de configuration standard pour les connecteurs», à la page 53.

Le présent guide fournit des informations sur l'utilisation de l'adaptateur avec ICS.

Remarque : Pour utiliser la fonction de livraison garantie d'événements, vous devez installer l'édition 4.1.1.2 de ICS.

Nouveautés de l'édition 1.3.x

IBM WebSphere Business Integration Adapter for JMS inclut le connecteur pour JMS. L'adaptateur fonctionne avec le courtier d'intégration InterChange Server (ICS). 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. L'adaptateur comprend :

- Un composant d'application spécifique à JMS
- Des exemples d'objets métier
- IBM WebSphere Adapter Framework, composé de :
  - Connector Framework
  - Des outils de développement (y compris Business object Designer et IBM CrossWorlds System Manager)
  - Des API (y compris CDK)

Le présent manuel fournit des informations sur l'utilisation de l'adaptateur avec ICS.

Important : Le connecteur n'ayant pas été internationalisé, ne l'exécutez pas avec ICS version 4.1.1 si vous n'êtes pas sûr de ne traiter que des données ISO Latin-1.

Le connecteur est à présent activé pour AIX 4.3.3 Patch Level 7.

Nouveautés de l'édition 1.2.x

Dans les précédentes versions du connecteur pour JMS, le gestionnaire de données qui servait à convertir les données entre messages JMS et objets métier IBM CrossWorlds était déterminé par les propriétés de connecteur `DataHandlerConfigMO` et `DataHandlerMimeType`. Ce procédé exigeait plusieurs instances du connecteur pour traiter différents formats de données. Dans l'édition 1.2.x, le connecteur vous permet si vous le souhaitez de préciser ces propriétés dans l'objet métier statique du connecteur ou dans le méta-objet enfant dynamique d'un objet métier de requête. Pour plus d'informations, voir «Configuration des méta-objets», à la page 31.

Nouveautés de l'édition 1.1.x

La présente édition du document contient des informations sur les nouvelles fonctionnalités et améliorations suivantes :

- Le connecteur reconnaît et lit les propriétés de conversion à partir d'un méta-objet dynamique ajouté en tant qu'enfant à l'objet métier transmis au connecteur. Les valeurs d'attribut du méta-objet enfant dynamique dupliquent les propriétés de conversion que vous pouvez préciser par le biais du méta-objet statique utilisé pour configurer le connecteur. La propriété de connecteur indiquant le méta-objet statique n'est plus requise, mais peut toujours être utilisée. Vous pouvez utiliser un méta-objet enfant dynamique indépendamment du méta-objet statique et vice-versa.
- Le connecteur accepte plusieurs noms de files d'attente pour la propriété de connecteur *`InputQueue`*. Le connecteur interroge les files d'attente en boucle et extrait un nombre maximum *`pollQuantity`* de messages depuis chacune d'entre elles. Le caractère séparateur des noms de files d'attente est le point-virgule.

Chapitre 1. Présentation de Adapter for JMS

- «Environnement de l'adaptateur pour JMS»
- «Terminologie de l'adaptateur pour JMS», à la page 4
- «Architecture de connecteur pour JMS», à la page 4
- «Traitement de message», à la page 4

Le connecteur pour JMS est un composant d'exécution de IBM WebSphere Business Integration Adapter for JMS. Il permet aux courtiers d'intégration IBM WebSphere d'échanger des objets métier avec les applications qui envoient ou reçoivent des données sous forme de messages JMS.

Le JMS est une API à standard ouvert qui permet d'accéder aux systèmes de messagerie d'entreprise. Il est conçu pour permettre aux applications d'envoyer et recevoir des données et événements.

Les connecteurs sont constitués d'un composant spécifique à l'application et de l'architecture de connecteur. Le composant propre à l'application contient des codes adaptés à une application ou une technologie spécifique. 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 spécifique à l'application. 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.

Ce document contient des informations sur le composant spécifique à l'application et sur l'architecture de connecteur. Il fait référence à ces deux éléments comme étant le connecteur.

Remarque : Tous les adaptateurs d'intégration WebSphere fonctionnent avec un courtier d'intégration. Le connecteur pour JMS fonctionne avec :

- le courtier d'intégration InterChange Server, décrit dans le document *Technical Introduction to IBM WebSphere InterChange Server*
- les courtiers de messages WebSphere MQ, décrits dans le document *Implementing Adapters with WebSphere Message Brokers*
- le courtier d'intégration WebSphere Application Server (WAS), décrit dans la documentation *Implementing Adapters with WebSphere Application Server*

Environnement de l'adaptateur pour JMS

Avant d'installer, de configurer et d'utiliser l'adaptateur, vous devez connaître les caractéristiques nécessaires à son environnement:

- «Compatibilité du courtier», à la page 2
- «Standards de l'adaptateur», à la page 2
- «Platesformes d'adaptateur», à la page 2
- «Dépendances de l'adaptateur», à la page 3
- «Données dépendant des paramètres nationaux», à la page 3

Compatibilité du courtier

L'architecture qu'utilise l'adaptateur doit être compatible avec la version du courtier d'intégration avec lequel l'adaptateur communique. Cet adaptateur s'exécute avec WebSphere Business Integration Adapter Framework version 2.6 et exige une des applications suivantes :

- WebSphere InterChange Server V 4.2.2 ou V 4.3
- WebSphere MQ Integrator V 2.1
- WebSphere MQ Integrator Broker V 2.1
- WebSphere Business Integration Message Broker V 5.0.1
- WebSphere Application Server Enterprise V 5.0.2, avec WebSphere Studio Application Developer Integration Edition V 5.0.1
- WebSphere Business Integration Server Foundation V 5.1 ou V 5.1.1

Voir les Notes d'édition pour connaître les exceptions.

Les caractéristiques matérielles et logicielles requises sont indiquées dans le Techdoc suivant

http://www.ibm.com/support/docview.wss?uid=swg27006249

Standards de l'adaptateur

L'adaptateur est écrit conformément au standard JMS 1.0.2. La prise en charge des autres versions du standard n'a pas été vérifiée mais rien ne laisse actuellement penser qu'elle pourrait poser problème.

L'adaptateur prend en charge les interfaces de messagerie point à point (PTP) et de publication/souscription (Pub/Sub) définies par le standard JMS. Ces styles sont souvent désignés respectivement par les termes messagerie basée sur des files d'attente et messagerie basée sur des rubriques. Une instance donnée d'adaptateur ne prend en charge qu'un seul style de messagerie à la fois (rubriques et files d'attente ne peuvent être mélangées dans la configuration). Toutefois, les deux styles de messagerie peuvent être pris en charge en exécutant plusieurs instances de l'adaptateur en parallèle, configurées pour PTP ou Pub/Sub.

Platesformes d'adaptateur

Outre le courtier, cet adaptateur exige l'un des systèmes d'exploitation suivants :

Remarque : Tous les systèmes d'exploitation exigent le compilateur Java (IBM JDK 1.4.2 for Windows 2000) pour compiler les adaptateurs personnalisés.

- Microsoft Windows 2000 (Professional, Server, ou Advanced Server) avec Service Pack 4
- Windows XP avec SP1A, pour WebSphere Business Integration Adapter Framework (uniquement pour les outils d'administration)
- Windows 2003 (Standard Edition ou Enterprise Edition)
- Solaris 8 (2.8) avec le Solaris Patch Cluster du 11 février 2004 ou ultérieur.
- Solaris 9 (2.9) avec le Solaris Patch Cluster du 11 février 2004. Cet adaptateur prend en charge la JVM 32 bits sur une plateforme 64 bits
- AIX 5.2 avec Maintenance Level 1 ou AIX 5.3. Cet adaptateur prend en charge la JVM 32 bits sur une plateforme 64 bits
- HP-UX 11i (11.11) avec les regroupements GOLDBASE11i et GOLDAPPS11i de juin 2003

- RedHat Enterprise Linux AS 3.0 avec Update 1
- RedHat Enterprise Linux ES 3.0 avec Update 1
- RedHat Enterprise Linux WS 3.0 avec Update 1
- SUSE Linux Enterprise Server x86 8.1 avec SP3
- SUSE Linux Standard Server x86 8.1 avec SP3 et Enterprise Server 8.1 SP3

  Remarque : Le composant TMTP (Tivoli Monitoring for Transaction Performance) de WebSphere Business Integration Adapter Framework V 2.6 n'est pas pris en charge sur Linux Red Hat.
- IBM JRE/JDK 1.4.2

Dépendances de l'adaptateur

L'adaptateur n'utilise et ne dépend d'aucune base de données. Toutes les bibliothèques client requises par le fournisseur JMS et le fournisseur JNDI doivent être incluses dans le chemin de classe de l'adaptateur. Ces bibliothèques sont différentes d'un fournisseur à l'autre.

Common Event Infrastructure

Cet adaptateur est compatible avec IBM Common Event Infrastructure, un standard de gestion des événements qui autorise l'interopérabilité avec d'autres applications IBM WebSphere qui génèrent des événements. Si la prise en charge Common Event Infrastructure est activée, les événements produits par l'adaptateur peuvent être reçus (ou utilisés) par une autre application compatible Common Event Infrastructure.

Pour plus d'informations, voir l'annexe Common Event Infrastructure du présent guide.

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 la collecte et la consultation des données relatives aux indicateurs de transaction.

Pour plus d'informations, voir l'annexe Application Response Measurement du présent guide.

Données dépendant des paramètres nationaux

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 entre deux emplacements qui utilisent des jeux différents de codes de caractères, il effectue la conversion des caractères afin de préserver le sens des informations.

Cet adaptateur prend en charge l'écriture bidirectionnelle (bi-di) pour l'arabe et l'hébreu, lorsqu'il est exécuté dans un environnement Windows. Cette fonctionnalité n'est pas prise en charge dans les autres environnements. Pour utiliser la fonctionnalité bidirectionnelle, vous devez configurer les propriétés standard bidirectionnelles. Pour plus d'informations, voir les propriétés standard de configuration des connecteurs dans l'Annexe A, «Propriétés de configuration standard pour les connecteurs», à la page 53.

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 connus (à la fois sur un ou plusieurs octets). La plupart des composants du système WebSphere Business Integration sont écrits en Java. Par conséquent, lorsque des données sont transférées entre la plupart des composants d'intégration, 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 `Locale` pour votre environnement. Pour plus d'informations sur les propriétés de configuration, voir l'Annexe A, «Propriétés de configuration standard pour les connecteurs», à la page 53.

Terminologie de l'adaptateur pour JMS

- **Fournisseur JMS** Un système de messagerie qui implémente JMS
- **Messages** Requêtes et événements contenant des données utilisées par les applications d'entreprise.
- **PTP** Messagerie de style point à point basée sur des files d'attente
- **Pub/Sub** Messagerie de type publication/souscription ou basée sur des rubriques
- **Destination JMS** représente la source ou la cible d'un message. Dans la messagerie PTP, une destination est une file d'attente. Dans la messagerie Pub/Sub, une destination est une rubrique. Ce terme est utilisé largement dans la spécification des deux descriptions et les noms de propriété réels lorsqu'une file d'attente ou une rubrique peut s'appliquer à une situation donnée.
- **ASI** Application-specific information—Métadonnées qui apparaissent sous la forme de paires nom=valeur, délimitées par des points-virgules, dans les objets métier et méta-objets.

Architecture de connecteur pour JMS

Les messages, dans le contexte de cet adaptateur, sont des requêtes et événements contenant des données utilisées par les applications d'entreprise. Les produits Message Oriented Middleware (MOM) permettent aux applications d'entreprise de s'envoyer et de recevoir des messages de façon asynchrone. L'API Java Message Service (JMS) a été établie pour standardiser la façon dont les programmes Java communiquent avec ces systèmes de messagerie. Par le passé, un client de messagerie était souvent écrit de façon à fonctionner avec un seul système MOM. Les clients JMS, tels que l'adaptateur, peuvent généralement tirer parti de tout système de messagerie prenant en charge JMS. L'adaptateur WBI pour JMS permet l'intégration à un nombre croissant de systèmes de messagerie d'entreprise prenant en charge le standard JMS.

Traitement de message

L'adaptateur prend en charge deux opérations principales :

1. L'extraction des messages à partir d'une destination JMS
2. La livraison d'un message à une destination JMS

L'adaptateur réalise ces opérations en établissant une connexion auprès d'un fournisseur JMS (tel que WebSphere MQ) puis en utilisant l'API JMS pour :

- Interroger et extraire les messages présents sur une destination JMS

- Générer et livrer de nouveaux messages demandés par le courtier

Ces deux opérations sont décrites en détail dans «Traitement des messages d'événement» et «Traitement des messages de requête», à la page 11.

Traitement des messages d'événement

Le connecteur vérifie régulièrement si des messages ont été livrés à une ou plusieurs destinations JMS. A chaque cycle d'interrogation, le connecteur :

1. Utilise l'API JMS pour extraire tout message en attente.
2. Appelle un gestionnaire de données configuré pour convertir le contenu du message en un objet métier.
3. Livre ou publie l'objet métier d'événement au courtier d'intégration configuré, afin qu'il soit traité par tout processus métier souscripteur.

Ces étapes sont illustrées dans la figure 1 et décrites en détail dans :

- «Détection d'événement»
- «Statut de l'événement et reprise sur incident», à la page 6
- «Extraction des événements», à la page 8

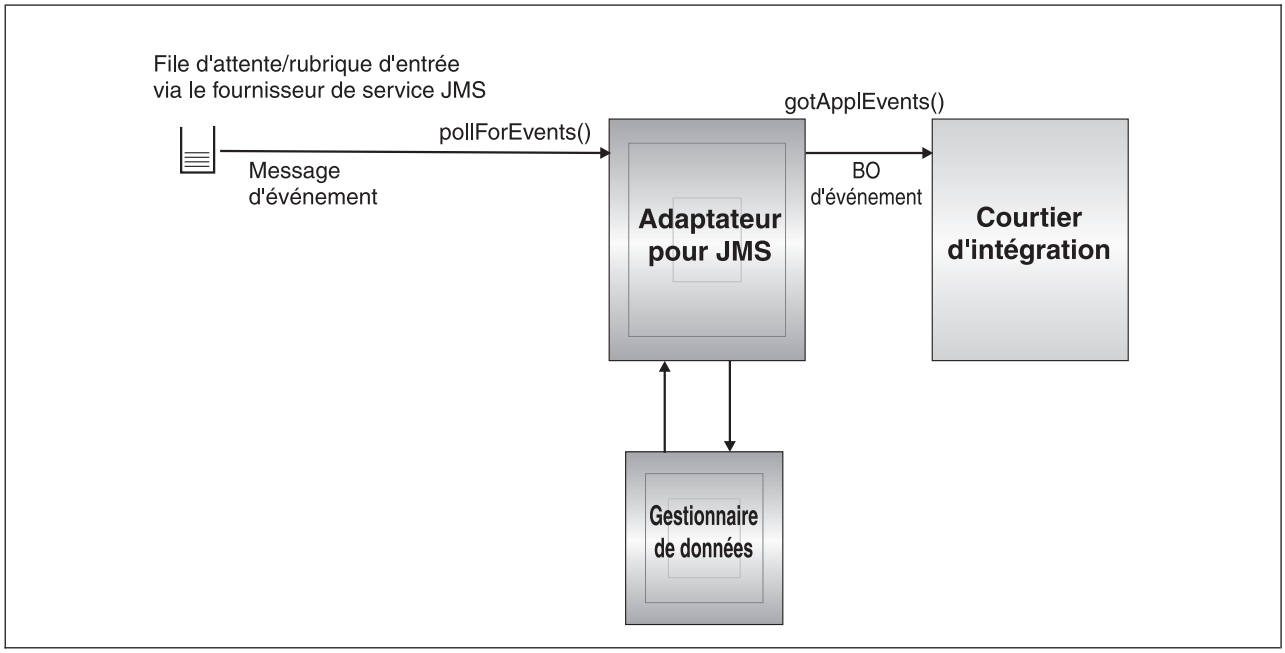


Figure 1. Flot de message d'événement

Détection d'événement

Pendant chaque cycle d'interrogation d'événement, le connecteur procède à une lecture non bloquante des messages, sur la destination précisée par la propriété de connecteur InputDestination (pour plus d'informations sur les propriétés du connecteur, voir «Configuration des propriétés du connecteur», à la page 19). Le connecteur extrait les messages puis les publie sur le courtier.

Le connecteur utilise la méthode `pollForEvents()` pour détecter d'éventuels messages, à intervalles réguliers. Pour chaque cycle d'interrogation, l'extraction de message est limitée au nombre maximal indiqué par la propriété de connecteur PollQuantity. S'il extrait tous les messages disponibles avant d'atteindre le

maximum indiqué, le connecteur n'attend pas de messages supplémentaires mais revient immédiatement du cycle d'interrogation.

Si plusieurs destinations sont précisées dans la propriété de connecteur InputDestination, le connecteur interroge chacune de façon circulaire. Il extrait et publie sur le courtier un nombre maximum *PollQuanity* de messages à partir de chaque destination. S'il vide toutes les destinations avant d'atteindre le maximum indiqué par PollQuantity, le connecteur revient immédiatement du cycle d'interrogation.

Par exemple, dans un scénario où

- le connecteur est configuré avec PollQuantity égale à 2 et les files d'attente d'entrée A, B, et C
- le file d'attente A contient 2 messages, la file d'attente B contient 1 message et la file d'attente C en contient 5

l'adaptateur extrairait les messages dans l'ordre suivant au cours d'un cycle d'interrogation unique :

1. Le premier message de la file d'attente A (laissant 1 message restant)
2. Le premier message de la file d'attente B (non vide)
3. Le premier message de la file d'attente C (laissant 4 messages restants)
4. Le message suivant de la file d'attente A (à présent vide)
5. Le connecteur contrôle la file d'attente B, mais elle est toujours vide.
6. Le message suivant de la file d'attente C (laissant 3 messages restants)

L'adaptateur revient ensuite du cycle d'interrogation car il a à présent interrogé un maximum de 2 messages dans chaque file d'attente (tel que défini par PollQuanity).

Statut de l'événement et reprise sur incident

L'extraction d'un message d'événement fait partie d'une transaction. Si le connecteur s'arrête de façon inattendue avant de valider la transaction, celle-ci est annulée et le message d'origine est restauré. L'architecture du connecteur ne prenant actuellement pas en charge les transactions réparties, le connecteur peut publier un événement sur le courtier mais s'arrête de façon inattendue ou perd la communication avant réception d'un accusé de réception de la part du courtier. Dans de tels cas, le connecteur ne dispose d'aucune information permettant de déterminer si l'événement a été reçu ou non par le courtier. Pour éviter la perte de messages d'événement, le connecteur ne valide la transaction qu'une fois que le connecteur a reçu une réponse du courtier confirmant la réception de l'événement. En cas de panne entre le moment où le connecteur publie un événement et celui où il reçoit un accusé de réception, la transaction est annulée automatiquement et le message d'origine est restauré. Puisqu'il est impossible de savoir si le message a été traité par le courtier ou non, ces événements sont désignés comme étant des événements en attente de validation.

Au redémarrage, le connecteur commencera à traiter les messages émis par la destination d'entrée et soumettra de nouveau l'événement en attente de validation. Bien que cette stratégie élimine tout risque de perte d'un événement, elle ne peut pas empêcher le fait que le même événement soit publié deux fois.

Il existe deux moyens de réduire ou d'éliminer le risque de livraison d'événement en double : utiliser une destination en cours (voir «Reprise sur incident avec une destination en cours», à la page 7) ou la livraison d'événement garantie (voir «Recovery with guaranteed event delivery», à la page 7).

Reprise sur incident avec une destination en cours : Pour contrôler la gestion des événements en attente de validation, vous pouvez créer une destination temporaire séparée en définissant la propriété de connecteur InProgressDestination.

Remarque : La reprise sur incident avec une destination en cours n'est pas prise en charge par la messagerie Pub/Sub.

Avant de publier un événement sur le courtier, le connecteur déplace le message d'événement depuis la destination d'entrée vers la destination en cours. Une fois qu'il a reçu l'accusé de réception du courtier, le connecteur supprime le message de la destination en cours. Ceci a pour effet d'isoler les messages en attente de validation qui n'ont pas été traités. Au démarrage, si le connecteur trouve des messages dans la destination en cours, il peut supposer en toute sécurité que ceux-ci proviennent d'une instance précédente du connecteur qui s'est arrêté de façon inattendue. Vous pouvez préciser différentes actions prises par le connecteur sur de tels messages (si la notification d'événement en double est inacceptable). Pour cela, indiquez l'une des quatre options suivantes pour la propriété de configuration InDoubtEvents :

- **Fail on startup** S'il trouve des messages dans la destination en cours pendant l'initialisation, le connecteur consigne une erreur et s'arrête immédiatement. Vous, ou un administrateur système, examinez ensuite le message et effectuez l'action appropriée : supprimer totalement ces messages ou les déplacer dans un emplacement différent.
- **Reprocess** S'il trouve des messages dans la destination en cours pendant l'initialisation, le connecteur traite ces messages en premier pendant les interrogations suivantes. Lorsque tous les messages de la destination en cours ont été traités, le connecteur commence à traiter les messages provenant de la destination d'entrée.
- **Ignore** S'il trouve des messages dans la destination en cours pendant l'initialisation, le connecteur les ignore et ne s'arrête pas.
- **Log error** Avec l'option de consignation d'erreur, si le connecteur trouve un message dans la destination en cours pendant l'initialisation, il consigne une erreur et ne s'arrête pas.

Pour plus d'informations, voir «InDoubtEvents», à la page 27.

Recovery with guaranteed event delivery : La fonctionnalité de livraison garantie d'événement permet à l'architecture du connecteur de s'assurer que les événements ne sont jamais perdus ni envoyés deux fois. L'architecture du connecteur prend en charge la livraison d'événement garantie via deux mécanismes : les événements gérés par le conteneur (CME) et l'élimination d'événements en double (DEE).

Evénements gérés par le conteneur (CME) : Vous pouvez utiliser le mécanisme CME lorsque le connecteur est configuré pour la messagerie PTP. Pour utiliser CME, WebSphere MQ doit être votre fournisseur JMS et les files d'attente source et de destination doivent être sur un gestionnaire de files d'attente.

Remarque : Lorsqu'il est configuré pour la messagerie Pub/Sub, le connecteur ne prend pas en charge CME. Pour plus d'informations sur le fonctionnement de cette méthode de livraison d'événement garantie, voir *Connector Development Guide for Java*. Pour plus d'informations sur la propriété de connecteur ContainerManagedEvents, voir «ContainerManagedEvents», à la page 64.

Elimination des événements en double (DEE) : DEE est l'approche recommandée pour mettre en oeuvre la livraison d'événement garantie pour l'adaptateur JMS. DEE est également la seule approche prise en charge par la messagerie Pub/Sub.

Avec DEE, un connecteur inclut un ID unique à chaque événement publié sur le courtier. L'architecture vérifie que le connecteur ne soumet pas plusieurs fois de suite le même ID d'événement. Si cela se produit, l'architecture suppose que le connecteur publie le même événement deux fois et élimine la deuxième soumission. Pour la messagerie PTP, DEE réduit la surcharge importante résultant de la copie des messages depuis et vers une destination en cours.

Ce connecteur inclut l'ID de message pour tous les événements lors de la publication d'objets métier sur le courtier. Si le connecteur ne parvient pas à poster un événement sur le courtier, en raison d'un problème de communication ou d'un arrêt inattendu, le message d'origine est remis dans la file d'attente d'entrée, tel que décrit précédemment. Au redémarrage, le connecteur commence à soumettre de nouveau les événements depuis la file d'attente, y compris les messages en attente de validation. Si DEE est activé, tout message en attente de validation qui a bien atteint le courtier dans le passé sera annulé. Ceci garantit que chaque message soit soumis une fois et une seule au courtier.

Avec DEE, évitez de changer l'ordre des messages dans les destinations pendant que le connecteur est hors ligne. DEE n'enregistre que le dernier ID de message extrait par l'adaptateur. DEE échouera dans le cas où, par exemple, de nouveaux messages de priorité supérieure font reculer le message en attente de validation dans la file d'attente, avant que l'adaptateur ne puisse redémarrer.

Pour plus d'informations sur DEE et sur son activation, voir *Connector Development Guide for Java*. Pour plus d'informations sur la propriété de connecteur DuplicateEventElimination, voir «DuplicateEventElimination», à la page 68.

Extraction des événements

L'extraction d'événements couvre le traitement général des événements par le connecteur. Elle commence avec la détection d'un événement entrant et s'achève avec sa conversion dans un format adapté à l'application cible, et sa livraison réussie au courtier d'intégration désigné. Le connecteur au courtier livre tous les événements de façon asynchrone ("en aveugle").

Les sections qui suivent traitent de l'extraction des événements :

- «Métadonnées et méta-objets»
- «Mappage d'objet métier», à la page 9
- «Présentation du mappage de l'en-tête du message», à la page 10
- «Archivage», à la page 10
- «Reprise après erreur», à la page 10

Métadonnées et méta-objets : Pour que le connecteur parvienne à convertir des messages en objets métier et vice-versa, il a besoin d'informations supplémentaires appelées métadonnées. Les métadonnées décrivent de quelle façon les données d'un objet, d'un message ou d'une application sont représentées ou doivent être traitées. Les métadonnées indiquent notamment l'objet métier à créer si le connecteur extrait un message d'une destination XYZ, ou le gestionnaire de données à utiliser pour sérialiser un objet métier de requête de type `Customer` avec l'instruction `Create`.

Attributs, propriétés, instructions et informations spécifiques à l'application constituent les métadonnées d'une définition d'objet métier. De plus, vous pouvez préciser un ou plusieurs méta-objets pouvant contenir des méta-données concernant les destinations, formats de données, gestionnaires de données, etc.

Il existe deux types de méta-objets : statiques et dynamiques. Vous créez un méta-objet statique pendant l'implémentation. Il contient des attributs qui fournissent des méta-données pour chaque type d'objet métier devant être pris en charge par le connecteur. Le méta-objet statique est spécifié dans les propriétés propres au connecteur. Il est lu par celui-ci pendant l'initialisation. Pour une présentation des propriétés du méta-objet et pour savoir comment elles affectent la transformation des messages, voir «Mappage d'objet métier» et «Présentation du mappage de l'en-tête du message», à la page 10.

Le second type de méta-objet est dynamique. Il permet de modifier les métadonnées utilisées par l'adaptateur pour traiter un objet métier requête par requête, pendant le traitement des requêtes. Pendant le traitement des événements, le méta-objet dynamique agit en tant que conteneur pour proposer les informations de transfert relatives à l'événement (par exemple, message, ID, priorité, etc.) afin que les processus métier en aval puissent utiliser les informations dans leur logique métier. Le méta-objet dynamique est représenté en tant qu'objet enfant spécialement marqué dans l'objet de niveau supérieur (ou la requête) de l'événement.

Vous pouvez choisir d'utiliser un ou les deux types de méta-objets dans la même implémentation. Les valeurs fournies dans un méta-objet dynamique sont généralement prioritaires sur les valeurs fournies dans le méta-objet statique. Pour plus d'informations sur les métadonnées, voir *Connector Development Guide for Java*. Pour plus d'informations sur la configuration de méta-objets statiques et dynamiques, voir «Configuration des méta-objets», à la page 31.

Mappage d'objet métier : Lors de l'extraction d'un message, le connecteur tente de déterminer sur quel objet métier le message doit être mappé.

Par défaut, le connecteur autorise le gestionnaire de données, configuré dans les propriétés du connecteur, à déterminer le type d'objet métier. Il transmettra le corps du message au gestionnaire de données et publiera l'objet métier retourné au courtier par le gestionnaire de données. Si le gestionnaire de données ne peut déterminer l'objet métier approprié, le connecteur fait échouer l'événement.

Si un méta-objet statique est précisé pour la propriété de configuration du connecteur ConfigurationMetaObject, le connecteur examine cet objet pour trouver une règle correspondant au message en termes de format et de destination d'entrée. Si la règle précisée dans le méta-objet précise à la fois le format et la destination de l'entrée, le connecteur observe cette règle, mais uniquement si le message correspond aux deux propriétés. Si l'une des deux propriétés est manquante, le connecteur utilise uniquement la propriété spécifiée.

Par exemple, un message dont le format d'entrée est `Cust_In` et la destination d'entrée est `MyInputDest` correspond aux règles de méta-objet statique :

1. `InputFormat=Cust_In;InputDestination=MyInputDest`
2. `InputFormat=Cust_In`
3. `InputDestination=MyInputDest`

S'il peut faire correspondre le message d'événement à une seule règle, le connecteur détermine l'objet métier en créant une nouvelle instance de cet objet métier et en la transmettant avec le corps du message, au gestionnaire de données précisé dans la règle. Si aucun gestionnaire de données n'est précisé dans la règle, le connecteur utilisera le gestionnaire de données par défaut, indiqué dans les propriétés de configuration du connecteur.

Si l'adaptateur peut faire correspondre le message d'événement avec plusieurs règles ou aucune, le connecteur autorise le gestionnaire de données à déterminer le type d'objet métier, en transmettant uniquement le corps du message au gestionnaire de données indiqué dans les propriétés de configuration du connecteur.

Présentation du mappage de l'en-tête du message : Pour transformer un message d'événement en objet métier, le connecteur compare les métadonnées relatives à l'objet métier avec les métadonnées du message, en les mappant les unes sur les autres. Comme décrit dans «Métadonnées et méta-objets», à la page 8, les métadonnées relatives aux objets métier résident dans les définitions d'objet métier (les informations spécifiques à l'application et les méta-objets enfants dynamiques), les propriétés du connecteur et les méta-objets statiques. Les méta-données du message sont contenues dans les en-têtes.

Pour accéder aux informations de l'en-tête du message spécifiques au transfert, et pour obtenir plus d'informations et de contrôle sur le transfert, vous pouvez ajouter des attributs à un méta-objet dynamique qui est un enfant d'une définition d'objet métier. Le fait d'ajouter ces attributs vous permet de lire et éventuellement d'écrire dans les en-têtes des messages et de modifier ainsi leurs métadonnées. De telles changements peuvent inclure la modification des propriétés JMS, le contrôle de la destination sur une base par requête (plutôt que d'utiliser la destination par défaut précisée dans les propriétés de l'adaptateur), le reciblage d'un CorrelationID de message, etc. Lorsque vous précisez de telles propriétés dans un méta-objet dynamique qui est un enfant d'une définition d'objet métier, le connecteur contrôle leurs équivalents dans les en-têtes de messages et renseigne un méta-objet dynamique basé sur le contenu de l'en-tête du message. Vous pouvez définir un ou tous les attributs de méta-objet dynamiques pris en charge, le connecteur renseignera le méta-objet en conséquence. Pour plus d'informations, y compris pour une liste des propriétés d'en-têtes de message que vous pouvez lire et écrire, voir «Alimentation du méta-objet enfant dynamique pendant l'interrogation», à la page 41.

Archivage : Si vous précisez la propriété spécifique au connecteur ArchiveDestination, le connecteur place une copie de tous les messages dont le traitement a réussi dans cette destination. Si ArchiveDestination n'est pas défini, les messages dont le traitement a réussi sont éliminés. Pour plus d'informations, voir «Configuration des propriétés spécifiques au connecteur», à la page 20.

Reprise après erreur : S'il rencontre des erreurs pendant la lecture depuis les destinations d'entrée, le connecteur retourne immédiatement la constante APPRESPONSETIMEOUT au courtier, entraînant l'arrêt et l'éventuel redémarrage du connecteur. De telles erreurs irrémédiables sont généralement dues à une perte de connexion avec le fournisseur JMS, ou à des erreurs internes rapportées par le fournisseur JMS et que le connecteur ne reconnaît pas, ou reconnaît mais juge irrémédiables (par exemple, un échec de transaction).

S'il rencontre des erreurs lors de la conversion du message entrant en un objet métier d'événement (par exemple, si le gestionnaire de données fait état d'un format de message incorrect), le connecteur fait échouer l'événement et consigne un message d'erreur approprié expliquant le motif. Si la propriété du connecteur ErrorDestination est définie et valide, le connecteur place une copie du message qui a échoué dans cette destination d'erreur. Dans le cas contraire, le message est éliminé.

Si le courtier rapporte une erreur une fois que le connecteur a publié l'objet métier d'événement, le connecteur fait échouer l'événement et consigne le message d'erreur rapporté par le courtier. Si la propriété du connecteur ErrorDestination est définie et valide, le connecteur place une copie du message qui a échoué dans cette destination. Dans le cas contraire, le message est éliminé.

Si le connecteur ne parvient pas à déterminer un objet métier pour un message, ou s'il publie un message sur un courtier et que le courtier rapporte que le message n'est pas pris en charge, le connecteur le considère comme non souscrit. Si la propriété du connecteur, UnsubscribedDestination est définie et valide, le connecteur place une copie du message non souscrit dans cette destination. Dans le cas contraire, le message est éliminé.

Traitement des messages de requête

Lorsqu'une requête d'objet métier est envoyée au connecteur, il crée un nouveau message dans la destination cible. L'en-tête du message est complété par une combinaison de valeurs définies par l'utilisateur et précisées dans les méta-objets de requête et les paramètres par défaut indiqués par les propriétés du connecteur. Le corps du message est complété par le contenu résultant du passage de l'objet métier de requête par le gestionnaire de données configuré.

La figure 2 illustre une communication de requête de message. Lorsque la méthode `doVerbFor()` reçoit un objet métier d'un courtier, le connecteur transmet l'objet métier au gestionnaire de données. Le gestionnaire de données convertit l'objet métier en un message convenable, qui est ensuite émis en tant que message vers une destination.

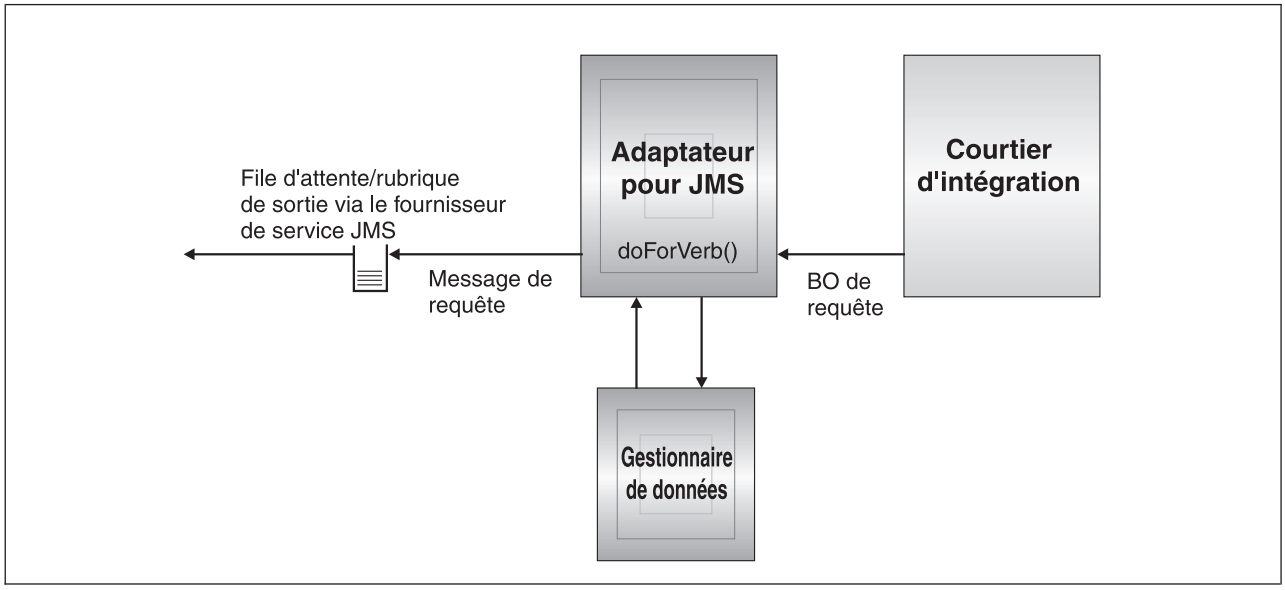


Figure 2. Flot de requête

Le connecteur peut avoir deux comportements pendant le traitement de la requête. Dans le premier, décrit ci-dessous en tant que traitement asynchrone, le connecteur place un message dans la destination cible et revient en indiquant sa réussite. Ce fonctionnement est dit "en aveugle". Dans le second, décrit ci-dessous en tant que traitement synchrone, le connecteur place de nouveau un message dans l'application cible mais attend qu'elle lui retourne une réponse.

Le mode de traitement est déterminé par la propriété numérique ResponseTimeout, précisée dans le méta-objet dynamique ou statique de la requête d'objet métier. Si cette propriété n'est pas définie ou est égale à -1, le connecteur livre la requête de façon asynchrone. Si cette propriété est supérieure ou égale à 0, l'adaptateur traite la requête de façon synchrone et attend au moins ce nombre de millisecondes pour qu'un message de réponse soit retourné par l'application cible. Le traitement de requête illustré dans la figure 2 est décrit en détail dans :

- «Prise en charge de l'instruction»
- «Traitement asynchrone»
- «Traitement synchrone», à la page 13

Prise en charge de l'instruction

Le connecteur n'accorde aucune valeur sémantique à l'instruction précisée dans l'objet métier de la requête. Il exécute la même action, plaçant un message dans une destination JMS, quelle que soit l'instruction spécifiée.

Traitement asynchrone

Pendant un traitement asynchrone, le connecteur convertit l'objet métier de requête en message, place celui-ci dans la destination cible puis retourne immédiatement au courtier. La réussite ou l'échec de la requête dépend entièrement de la capacité du connecteur à placer le message dans la destination JMS. Notez que la réussite de cette livraison n'implique pas que l'application cible a reçu ou recevra le message. En raison de la nature asynchrone des systèmes de messagerie, un message peut rester indéfiniment sur le fournisseur JMS, jusqu'à ce que l'application cible soit capable de le traiter ou jusqu'à ce qu'il expire (selon la configuration).

Le connecteur transforme d'abord l'objet métier de requête, en texte à l'aide du gestionnaire de données configuré. Le connecteur utilise le gestionnaire de données indiqué, de préférence dans l'ordre suivant

1. le méta-objet dynamique
2. le méta-objet statique
3. les propriétés de configuration du connecteur

Le connecteur crée un nouveau message, avec les données d'objet métier mises en série comme corps du message. Il renseigne les en-têtes de messages comme indiqué dans le tableau ci-après. Dans tous les cas où une propriété peut être précisée dans le méta-objet dynamique ou statique, la valeur indiquée dans le méta-objet dynamique est prioritaire sur toute valeur définie dans le méta-objet statique. Pour obtenir une description et une liste des propriétés possibles dans les méta-objets, voir «Configuration des méta-objets», à la page 31.

Tableau 1. Renseignement de l'en-tête de message JMS pendant le traitement de requête asynchrone

Propriété de méta-objet	Action par défaut si la propriété n'est pas définie	Action entreprise si la propriété est définie
`OutputFormat`	Le connecteur ne précise pas le format du message	Le connecteur indique cette valeur comme format du message.
`CorrelationID`	Le connecteur laisse cette valeur vierge dans l'en-tête du message.	Le connecteur précise cette valeur pour l'ID de corrélation, dans l'en-tête du message de requête.
`ReplyToDestination`	Le connecteur laisse cette valeur vierge dans l'en-tête du message.	Le connecteur précise cette valeur pour la destination de la réponse dans l'en-tête du message de la requête.

Tableau 1. Renseignement de l'en-tête de message JMS pendant le traitement de requête asynchrone (suite)

`Priority (Priorité)`	Le connecteur autorise le fournisseur JMS à utiliser la priorité par défaut.	Le connecteur définit une priorité de message numérique à l'aide de cette valeur.
`JMSProperties`	Aucune	Le connecteur mappe les propriétés JMS précisées sur celles de l'en-tête de message.

Les attributs suivants du méta-objet déterminent le mode de livraison du message :

Tableau 2. Livraison asynchrone à la destination

Propriété de méta-objet	**Action par défaut si la propriété n'est pas définie**	**Action entreprise si la propriété est définie**
`OutputDestination`	Il est indispensable d'indiquer une valeur.	Destination cible du message.
`DeliveryMode`	Le connecteur autorise le fournisseur JMS à définir la persistance du message.	Le connecteur écrit le message de façon persistante/non persistante, comme indiqué par l'utilisateur.

En fonction de la capacité du connecteur à réussir à livrer le message de requête à la destination de sortie (cible), un des codes suivants est retourné au courtier :

Tableau 3. Codes de retour asynchrones

Action du connecteur	**Code de retour vers le courtier**
A réussi à livrer le message à une destination cible.	SUCCEED
Ne parvient pas à livrer le message en raison d'erreurs remédiables telles que des méta-données incorrectes ou incomplètes, un incident sur le gestionnaire de données, ou des problèmes généraux de traitement.	FAIL
Ne parvient pas à livrer le message en raison d'erreurs irrécupérables rapportées par le fournisseur JMS, telles qu'un incident de connexion	APPRESPONSETIMEOUT

Traitement synchrone

Dans le traitement synchrone, le connecteur fournit la requête à la destination cible puis attend un message de réponse sur la seconde destination. La création du message de requête est identique à celle décrite dans le traitement asynchrone. Toutefois, le connecteur vérifie également la présence d'attributs supplémentaires dans le méta-objet :

Tableau 4. Propriétés de méta-objet synchrone

Propriété de méta-objet	**Action par défaut si la propriété n'est pas définie**	**Action entreprise si la propriété est définie**
`ResponseTimeout`	Il est indispensable d'indiquer une valeur.	Durée minimale (exprimée en millisecondes) pendant laquelle l'adaptateur doit attendre le retour d'un message de réponse.
`TimeoutFatal`	S'il ne reçoit pas de réponse dans le délai précisé par `ResponseTimeout`, le connecteur retourne APPRESPONSETIMEOUT au courtier, ce qui entraîne généralement l'arrêt du connecteur.	S'il ne reçoit pas de réponse, le connecteur fait échouer la requête (il retourne FAIL au courtier) mais ne s'arrête pas.

La livraison du message à la destination cible est identique à celle décrite pour le traitement asynchrone, à l'exception des points suivants :

Tableau 5. Livraison synchrone à la destination

Propriété de méta-objet	Action par défaut si la propriété n'est pas définie	Action entreprise si la propriété est définie
`ReplyToDestination`	Même chose que pour le traitement asynchrone.	Le connecteur complète cette zone du message de requête avec la valeur de la propriété spécifique au connecteur ReplyToDestination.

Le connecteur attend un message de réponse émis par l'application cible précisée par ReplyToDestination, pendant une durée au moins égale au temps indiqué par l'attribut de méta-objet ResponseTimeout. Si une réponse n'est pas retournée dans cet intervalle, le connecteur indique une erreur.

Critères de réponse : Le connecteur ne suppose pas que le premier message de la destination de la réponse est le message de réponse correct. A la place, il suit les conventions de réponse aux requêtes JMS et examine le premier message dont l'ID de corrélation correspond à l'ID de message de la requête. En d'autres termes, l'application qui reçoit le message de requête doit créer un message de réponse dont l'ID de corrélation est identique à l'ID de message de requête. Elle doit ensuite placer ce message dans la destination de réponse précisée par le message de requête.

Toutes les applications n'appliquent pas la convention d'utilisation de l'ID de corrélation pour mapper les messages de requête et de réponse. Le connecteur accepte alors des critères personnalisés permettant d'identifier un message de réponse.

Dès réception d'un objet métier de traitement de requête asynchrone, le connecteur vérifie la présence de la paire nom-valeur `response_selector=` dans les informations spécifiques à l'application de l'instruction. Si aucune paire nom-valeur n'existe, le connecteur identifie le message de réponse à l'aide de l'ID de corrélation de message, tel qu'indiqué ci-dessus.

Si une paire nom-valeur de sélecteur de réponse est définie, le connecteur considère que la valeur représente une chaîne de sélecteur de message JMS capable d'identifier le message de réponse. Voici quelques exemples d'utilisation. Pour plus d'informations sur la syntaxe du sélecteur de message JMS, voir la spécification de l'API JMS. Notez que le connecteur ne procède pas à l'analyse syntaxique du sélecteur de message JMS. Par contre, la syntaxe est comprise par le fournisseur JMS. Le connecteur rend le sélecteur disponible au fournisseur JMS, en tant que moyen de filtrage des messages (un peu comme une requête de base de données).

Par exemple, des informations spécifiques à l'application contenant la paire nom-valeur

```
response_selector=JMSType = 'xmlResponse'
```

informent le connecteur que le message de réponse doit correspondre à la chaîne de sélecteur `JMSType = 'xmlResponse'`. Le connecteur transmet ce sélecteur au fournisseur JMS, puis retourne le premier message fourni à la destination de réponse, où la zone de type JMS du message est égale à `xmlResponse`.

Dans tous les cas, la chaîne du sélecteur de message doit être capable d'identifier de façon unique une réponse et une seule. Si plusieurs messages répondant aux critères du sélecteur de réponse ont été livrés à la destination de réponse, l'adaptateur n'extraira que le premier. Tout autre message de réponse correspondant aux critères sera ignoré.

Pour autoriser un sélecteur de message unique lors de l'exécution, le connecteur prend en charge les remplacements dynamiques de valeurs d'attribut dans le sélecteur de message lui-même. Pour ce faire, vous devez indiquer dans le sélecteur de réponse un paramètre blanc, sous forme d'un nombre entier entre accolades (″{1}″). Il doit être suivi d'un caractère ″:″ et d'une liste d'attributs séparés par des virgules à utiliser pour la substitution. Le nombre entier de la marque de réserve a un rôle d'index utilisé par l'attribut pour le remplacement.

Par exemple, le sélecteur de message suivant

```
response_selector=JMSCorrelationID LIKE '{1}':MyDynamicMO.CorrelationID
```

indique au connecteur de remplacer le repère {1} par la valeur de l'attribut `CorrelationID`, dans l'objet enfant `MyDynamicMO`. Si l'attribut `CorrelationID` avait la valeur `123ABC`, le connecteur générerait et utiliserait le sélecteur de message

```
JMSCorrelation LIKE '123ABC'
```

Vous pouvez également préciser plusieurs substitutions, comme indiqué ci-dessous :

```
response_selector=Name LIKE '{1}'AND Zip LIKE '{2}':PrimaryID,Address[4].AddressID
```

Dans cet exemple, le connecteur remplace `'{1}'` par la valeur de l'attribut `PrimaryID` à partir de l'objet métier de niveau supérieur, et `'{2}'` par la valeur de `AddressID` à partir de la cinquième position (base 0) de l'objet de conteneur enfant `Address`. Grâce à cette approche, vous pouvez référencer n'importe quel attribut de l'objet métier et méta-objet du sélecteur de message de réponse.

Pour préciser la valeur littérale ″{″ dans le sélecteur de message, utilisez ″{{″. Par exemple, le sélecteur suivant

```
response_selector=PrimaryID LIKE {{1}
```

sera reconnu par l'adaptateur en tant que la valeur littérale

```
PrimaryID LIKE {1}
```

Dans ce cas, le connecteur n'effectuera aucune substitution sur la valeur '{1}'.

Lorsque le connecteur rencontre dans des valeurs d'attributs des caractères spéciaux tels que ″{″, ″}″, ″:″ ou ″;″, ils sont insérés directement dans la chaîne de requête. Ceci vous permet d'inclure dans une chaîne de requête des caractères spéciaux, servant également de délimiteurs d'informations spécifiques à l'application. Par exemple, le sélecteur suivant

```
Response_selector=PrimaryID = '{1}':Foo
```

lorsque l'attribut `Foo` a une valeur égale à `{A:B};{C:D}`, est converti en un sélecteur de message littéral tel que

```
PrimaryID = '{A:B};{C:D}'
```

Traitement des réponses : Pour savoir comment réagir à réception du message de réponse, le connecteur contrôle la propriété de résultat JMS précisée par la propriété de connecteur MessageResponseResultProperty. En fonction de la valeur de cette propriété JMS, le connecteur s'attend à ce que le corps du message de réponse contienne un objet métier ou un message d'erreur (voir tableau ci-dessous). Dans tous les cas, le connecteur renvoie au courtier le code de retour correspondant. Par exemple, si la propriété de résultat JMS est égale à `VALCHANGE` dans le message, le connecteur agit comme indiqué pour `VALCHANGE`, et retourne au courtier la valeur numérique correspondant à la constante de courtier `VALCHANGE`.

Tableau 6. Traitement des messages de réponse

Valeur de la propriété de résultat JMS	Action du connecteur
SUCCESS	Ne modifie pas l'objet métier de requête et le retourne simplement au courtier.
VALCHANGE MULTIPLE_HITS *undefined value*	Renseigne de nouveau l'objet métier de requête avec le contenu du corps du message de réponse. Si le corps du message de réponse est vide, l'objet métier de requête est laissé inchangé. Renseigne de nouveau le méta-objet dynamique de l'objet métier de requête avec les zones de l'en-tête JMS du message de réponse.
FAIL FAIL_RETRIEVE_BY_CONTENT BO_DOES_NOT_EXIST UNABLE_TO_LOGIN VALDUPES	Si la réponse est renseignée, le connecteur suppose qu'il s'agit d'un message d'erreur et le retourne au courtier. Si le corps du message de réponse est vide, le connecteur retourne un message d'erreur générique au courtier.
APPRESPONSETIMEOUT	Même chose que ci-dessus, sauf que le retour de APPRESPONSETIMEOUT au courtier provoque l'arrêt de l'agent de l'adaptateur.
unrecognized value	Le connecteur fait échouer la requête.

Gestion des erreurs : Si le connecteur rencontre des erreurs lorsqu'il lit ou écrit le message de requête dans la destination cible ou contrôle le message de réponse (le cas échéant), il retourne immédiatement APPRESPONSETIMEOUT au courtier. Ceci entraîne l'arrêt ou le redémarrage possible de l'adaptateur. De telles erreurs irrémédiables sont généralement dues à une perte de connexion au fournisseur JMS ou à des erreurs internes rapportées par le fournisseur JMS et que le connecteur ne reconnaît pas, ou reconnaît mais juge irrémédiables (par exemple, un échec de transaction).

S'il rencontre des erreurs lors de la conversion d'un objet métier en message ou vice-versa (par exemple, si le gestionnaire de données rapporte un format de message incorrect), le connecteur fait échouer la requête et consigne un message d'erreur expliquant le motif.

Pour plus d'informations, et pour consulter des scénarios d'incident sur événement, voir «Gestion des erreurs», à la page 49.

Chapitre 2. Installation et configuration de l'adaptateur

- «Tâches d'installation»
- «Installation de l'adaptateur et des fichiers associés»
- «Structure de fichiers installée»
- «Configuration des propriétés du connecteur», à la page 19
- «Configuration du style de message», à la page 30
- «Configuration de JNDI», à la page 30
- «Configuration des méta-objets», à la page 31
- «Configuration des scripts de démarrage», à la page 43
- «Création de plusieurs instances de connecteur», à la page 43
- «Démarrage du connecteur», à la page 44
- «Arrêt du connecteur», à la page 46

Ce chapitre explique comment installer et configurer le connecteur, et comment configurer les flots de messages pour l'utiliser.

Tâches d'installation

Pour installer l'adaptateur pour JMS, procédez comme suit :

- **Installation du courtier d'intégration** Cette tâche, qui comprend l'installation du système d'intégration WebSphere et le démarrage du courtier d'intégration, est décrite dans la documentation d'installation relative à votre courtier et à votre système d'exploitation.
- **Installation de l'adaptateur et des fichiers associés** Cette tâche comprend l'installation des fichiers de l'adaptateur à partir du module logiciel de votre système d'exploitation. Voir «Installation de l'adaptateur et des fichiers associés».

Installation de l'adaptateur et des fichiers associés

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

Structure de fichiers installée

Les sections ci-dessous décrivent les chemins et noms des fichiers du produit après l'installation.

Structure de fichiers du connecteur sous Windows

L'installation du connecteur copie sur votre système les fichiers standard associés.

L'utilitaire installe l'agent du connecteur dans le répertoire *ProductDir*\connectors\JMS et ajoute un raccourci pour l'agent dans le menu Démarrer. Notez que *ProductDir* représente le répertoire dans lequel IBM WebSphere Business Integration Adapters est installé. La variable d'environnement contient le chemin du répertoire *ProductDir*, par défaut IBM\WebSphereAdapters.

Le tableau 7 décrit la structure de fichiers utilisée sous Windows par le connecteur, et indique les fichiers qui sont automatiquement installés lorsque vous utilisez le programme d'installation.

Tableau 7. Structure de fichiers installée sous Windows pour le connecteur

Sous-répertoire de *ProductDir*	**Description**
`connectors\JMS\CWJMS.jar`	Classes utilisées par le connecteur JMS
`connectors\JMS\start_JMS.bat`	Script de démarrage du connecteur (NT/2000)
`connectors\messages\JMSConnector.txt`	Fichier de message du connecteur
`bin\Data\App\JMSConnectorTemplate`	Modèle de fichier pour la définition du connecteur
`connectors\JMS\Samples\JMSConnector.cfg`	Exemple de fichier de configuration du connecteur
`connectors\JMS\Samples\PortConnector.cfg`	Exemple de fichier de configuration du connecteur de port
`connectors\JMS\Samples\Sample_JMS_Contact.xsd`	Exemple de fichier de référentiel d'objet métier
`connectors\JMS\Samples\Sample_JMS_MO_Config.xsd`	Exemple de méta-objet
`connectors\JMS\Samples\Sample_JMS_MO_DataHandler.xsd`	Exemple de méta-objet de gestionnaire de données
`connectors\JMS\Samples\Sample_JMS_MO_DataHandler_DelimitedConfig.xsd`	Exemple de méta-objet de gestionnaire de données délimitées
`connectors\JMS\Samples\Sample_JMS_DynMO.xsd`	Exemple de méta-objet dynamique
`connectors\JMS\Samples\JMSPropertyPairs.xsd`	Exemple d'objet métier enfant de propriétés JMS pour méta-objet dynamique

Remarque : Tous les noms de chemin des produits sont relatifs au répertoire dans lequel le produit est installé sur votre système.

Structure de fichiers du connecteur sous UNIX

L'installation du connecteur copie sur votre système les fichiers standard associés.

L'utilitaire installe l'agent du connecteur dans le répertoire *ProductDir*`/connectors/JMS`.

Le tableau 8 décrit la structure de fichiers utilisée sous UNIX par le connecteur, et indique les fichiers qui sont automatiquement installés lorsque vous utilisez le programme d'installation.

Tableau 8. Structure de fichiers installée sous UNIX pour le connecteur

Sous-répertoire de *ProductDir*	**Description**
`connectors/JMS/CWJMS.jar`	Contient les classes utilisées par le connecteur JMS

Tableau 8. Structure de fichiers installée sous UNIX pour le connecteur (suite)

Sous-répertoire de *ProductDir*	Description
connectors/JMS/start_JMS.sh	Script de démarrage système pour le connecteur. Il est appelé depuis le script du gestionnaire de connecteurs générique. Lorsque vous cliquez sur l'écran de configuration du connecteur dans System Manager, l'installation crée un encapsuleur personnalisé pour ce script de gestionnaire de connecteur. Vous utiliserez cet encapsuleur personnalisé pour démarrer et arrêter le connecteur.
connectors/messages/JMSConnector.txt	Fichier de message du connecteur
binData/App/JMSConnectorTemplate	Modèle de fichier pour la définition du connecteur
connectors/JMS/Samples/JMSConnector.cfg	Exemple de fichier de configuration du connecteur
connectors/JMS/Samples/PortConnector.cfg	Exemple de fichier de configuration du connecteur de port
connectors/JMS/Samples/Sample_JMS_Contact.xsd	Exemple de fichier de référentiel d'objet métier
connectors/JMS/Samples/Sample_JMS_MO_Config.xsd	Exemple de méta-objet
connectors/JMS/Samples/Sample_JMS_MO_DataHandler.xsd	Exemple de méta-objet de gestionnaire de données
connectors/JMS/Samples/Sample_JMS_MO_DataHandler_DelimitedConfig.xsd	Exemple de méta-objet de gestionnaire de données délimitées
connectors/JMS/Samples/Sample_JMS_DynMO.xsd	Exemple de méta-objet dynamique
connectors/JMS/Samples/JMSPropertyPairs.xsd	Exemple d'objet métier enfant de propriétés JMS pour méta-objet dynamique

Remarque : Tous les noms de chemin des produits sont relatifs au répertoire dans lequel le produit est installé sur votre système.

Configuration du connecteur

Une fois l'adaptateur installé, vous devez le configurer. Pour ce faire, effectuez les tâches décrites dans les sections suivantes :

- «Configuration des propriétés du connecteur»
- «Configuration du style de message», à la page 30
- «Configuration de JNDI», à la page 30
- «Configuration des méta-objets», à la page 31
- «Configuration des scripts de démarrage», à la page 43

Configuration des propriétés du connecteur

Les connecteurs ont deux types de propriétés de configuration, décrits dans les sections suivantes :

- «Configuration des propriétés standard du connecteur», à la page 20
- «Configuration des propriétés spécifiques au connecteur», à la page 20

Vous devez définir les valeurs de ces propriétés avant d'exécuter le connecteur.

Utilisez Connector Configurator pour configurer les propriétés du connecteur :

- Pour obtenir une description de Connector Configurator et des procédures étape par étape, voir l'Annexe B, «Connector Configurator», à la page 79.
- Pour une description des propriétés standard du connecteur, voir «Configuration des propriétés standard du connecteur» puis l'Annexe A, «Propriétés de configuration standard pour les connecteurs», à la page 53.
- Pour une description des propriétés spécifiques au connecteur, voir «Configuration des propriétés spécifiques au connecteur».

Configuration des propriétés standard du connecteur

Les propriétés standard de configuration fournissent des informations utilisées par tous les connecteurs. Voir l'Annexe A, «Propriétés de configuration standard pour les connecteurs», à la page 53 pour plus d'informations sur ces propriétés. Pour une procédure étape par étape de définition de ces propriétés, voir l'Annexe B, «Connector Configurator», à la page 79.

Remarque : Lorsque vous définissez des propriétés de configuration dans Connector Configurator, précisez votre courtier à l'aide de la propriété BrokerType. Une fois définies, les propriétés applicables à votre courtier apparaissent dans la fenêtre Connector Configurator.

Configuration des propriétés spécifiques au connecteur

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

Le tableau 9 contient les propriétés de configuration spécifiques au connecteur. Pour obtenir une explication des propriétés, voir les sections suivantes.

Tableau 9. Propriétés de configuration spécifiques au connecteur

Nom	Valeurs possibles	Valeur par défaut	Obligatoire
ArchivalConnectionFactoryName	Nom de l'objet du magasin JNDI à extraire et utiliser pour les événements d'archivage ; prend en charge les styles de publication PTP (basé sur les files d'attente) et Pub/Sub (basé sur les rubriques).		Non
ArchiveDestination	*Destination des copies des messages dont l'envoi a réussi*		Non
ConfigurationMetaObject	*Méta-objet de configuration*		Voir la description de la propriété
ConnectionFactoryName	*Fabrique de connexion de file d'attente ou rubrique JMS définie dans le magasin JNDI.*		Oui
CTX_Authoritative	*Constante contenant le nom de la propriété d'environnement permettant d'indiquer l'autorité du service requis.*		Non
CTX_Batchsize	*Constante contenant le nom de la propriété d'environnement permettant d'indiquer la taille de traitement par lot à utiliser pour retourner des données via le protocole du service.*		Non

Tableau 9. Propriétés de configuration spécifiques au connecteur (suite)

Nom	Valeurs possibles	Valeur par défaut	Obligatoire
CTX_DNS_URL	*Constante contenant le nom de la propriété d'environnement permettant d'indiquer les noms de domaine et d'hôte DNS à utiliser pour le contexte de l'URL JNDI (par exemple "dns://somehost/wiz.com").*		Non
CTX_InitialContextFactory	*Nom de la classe de fabrique à utiliser pour établir un contexte JNDI initial.*		Oui
CTX_Language	*Constante contenant le nom de la propriété d'environnement permettant d'indiquer la langue à utiliser de préférence avec le service.*		Non
CTX_ObjectFactories	*Constante contenant le nom de la propriété d'environnement permettant d'indiquer la liste des fabriques d'objets à utiliser.*		Non
CTX_ProviderURL	*URL identifiant le contexte JNDI dans lequel se situe la fabrique de connexion.*		Oui
CTX_Referral	*Constante contenant le nom de la propriété d'environnement permettant d'indiquer de quelle façon les référenceurs rencontrés par le fournisseur de service doivent être traités.*		Non
CTX_SecurityAuthentication	*Constante contenant le nom de la propriété d'environnement permettant d'indiquer le niveau de sécurité à utiliser.*		Non
CTX_SecurityCredentials	*Constante contenant le nom de la propriété d'environnement permettant d'indiquer les accréditations de l'entité pour authentifier l'appelant au service.*		Non
CTX_SecurityPrincipal	*Constante contenant le nom de la propriété d'environnement permettant d'indiquer l'identité de l'entité pour authentifier l'appelant au service.*		Non
CTX_SecurityProtocol	*Constante contenant le nom de la propriété d'environnement permettant d'indiquer le protocole de sécurité à utiliser.*		Non
CTX_URLPackagePrefixes	*Constante contenant le nom de la propriété d'environnement permettant d'indiquer la liste des préfixes de modules à utiliser lors du chargement dans des fabriques de contexte d'URL.*		Non
DataHandlerClassName	*Nom de la classe du gestionnaire de données à instancier.*		Voir la description de la propriété
DataHandlerConfigMO	*Nom du méta-objet du gestionnaire de données contenant des informations de configuration pour DataHandlerMimeType*	`MO_DataHandler_ Valeur par défaut`	Voir la description de la propriété
DataHandlerMimeType	*Type mime à utiliser lors de la sélection du gestionnaire de données par défaut*	`text/delimited`	Voir la description de la propriété
DataHandlerPoolSize	*Nombre maximum d'instances DataHandler à mettre en mémoire cache pour un type particulier de DataHandler*	`30`	Non

Tableau 9. Propriétés de configuration spécifiques au connecteur (suite)

Nom	Valeurs possibles	Valeur par défaut	Obligatoire
DefaultVerb	*Indique l'instruction à définir dans un objet métier entrant*	`Create`	Non
EnableMessageProducerCache	`true` ou `false`	`true`	Non
ErrorDestination	*Destination des messages non traités*		Non

InDoubtEvents	`FailOnStartup`
	`Reprocess`
	`Ignore`
	`LogError`
		`Reprocess`	Non

InProgressDestination	*Destination de stockage temporaire*		Non
InputDestination	*Nom des destinations d'interrogation*		Non
LookupDestinationsUsingJNDI	`true` ou `false`	`false`	Non
MessageFormatProperty	*Nom de propriété précisant le format du message*	`JMSType`	Non
MessageResponseResultProperty	*Propriété du message de réponse indiquant le résultat de l'opération de requête*	`WBI_Result`	Oui, pour un traitement synchrone.
PollQuantity	*Nombre de messages à extraire de chaque destination indiquée dans la propriété* `InputDestination`	1	Non
ReplyToDestination	*Destination des fichiers de réponse lorsque le connecteur émet des requêtes*		Oui, pour un traitement synchrone.
SessionPoolSizeForRequests	*Taille de pool maximale pour la mise en mémoire cache des sessions utilisées au cours du traitement de la requête.*	10	Non
UnsubscribeOnTerminate	*Indique les rubriques supprimées de InputDestination.*		Non

UseDefaults	`true`
	ou
	`false`
		`false`	Non

UseDurableSubscriptions	`true`
	ou
	`false`
		`false`	Non

WorkerThreadCount	*Nombre maximum d'unités d'exécution parallèles à interroger.*	1	Non

ArchivalConnectionFactoryName

Cette propriété permet au connecteur de prendre en charge l'archivage des événements dans des styles point à point ou basés sur les rubriques. La propriété nomme la file d'attente JMS ou l'objet de fabrique de connexion de rubrique défini dans le magasin JNDI, et que le connecteur doit extraire et utiliser pour établir une connexion auprès du fournisseur JMS. Cet objet de connexion est ensuite utilisé pour créer des références de publication pour les destinations d'archive. Les propriétés de connecteur qui définissent les définitions d'archivage sont les suivantes :

- InProgressDestination
- ErrorDestination
- UnsubscribedDestination
- ArchiveDestination

Si cette propriété n'est pas définie, le connecteur utilise la fabrique définie dans la propriété ConnectionFactoryName pour créer des références aux destinations d'archivage.

Cette propriété est indiquée sous la forme

```
connection_factory_name:<msg_system_type>
```

où connection_factory_name est obligatoire et le second paramètre (<msg_system_type>) facultatif, avec la valeur Topics ou Queues. Le second paramètre indique à l'adaptateur de déterminer le type de système de message en fonction de la configuration de l'utilisateur.

Par exemple, pour préciser une messagerie pub/sub, configurez ConnectionFactoryName comme suit :

```
<connection_factory_name>:Topics
```

Pour une messagerie point à point, configurez ConnectionFactoryName comme suit :

```
<connection_factory_name>:Queues
```

Si vous ne mentionnez pas le second paramètre (Topics ou Queues), l'adaptateur tentera d'identifier le système de messagerie en fonction de l'instance de l'objet de fabrique.

Valeur par défaut = none.

ArchiveDestination

Destination des copies des messages dont l'envoi a réussi.

La valeur par défaut est `CWLD_ARCHIVE`.

ConfigurationMetaObject

Nom du méta-objet statique contenant les informations de configuration du connecteur.

Il n'y a pas de valeur par défaut.

ConnectionFactoryName

Nom de l'objet de fabrique de connexion de file d'attente ou de rubrique JMS, défini dans le magasin JNDI, et que le connecteur doit extraire et utiliser pour établir une connexion auprès du fournisseur JMS. Pour rechercher ce nom, le connecteur utilise le contexte JNDI initial établi par les propriétés CTX_InitialContextFactory et CTX_ProviderURL.

Cette propriété est indiquée sous la forme

```
connection_factory_name:<msg_system_type>
```

où connection_factory_name est obligatoire et le second paramètre (<msg_system_type>) facultatif, avec la valeur Topics ou Queues. Le second paramètre indique à l'adaptateur de déterminer le type de système de message en fonction de la configuration de l'utilisateur.

Par exemple, pour préciser une messagerie pub/sub, configurez ConnectionFactoryName comme suit :

```
<connection_factory_name>:Topics
```

Pour une messagerie point à point, configurez ConnectionFactoryName comme suit :

```
<connection_factory_name>:Queues
```

Si vous ne mentionnez pas le second paramètre (Topics ou Queues), l'adaptateur tentera d'identifier le système de messagerie en fonction de l'instance de l'objet de fabrique.

Valeur par défaut = none.

CTX_Authoritative

Constante contenant le nom de la propriété d'environnement permettant d'indiquer l'autorité du service requis. Si la propriété est définie sur la chaîne "true", l'accès est la source qui dispose des droits les plus élevés (prioritaire sur la mémoire cache et les répliques). Pour toutes les autres valeurs, la source n'est pas nécessairement celle qui a les droits les plus élevés (même si elle peut l'être).

Par défaut, la propriété est définie sur "false".

CTX_Batchsize

Constante contenant le nom de la propriété d'environnement permettant d'indiquer la taille de traitement par lot à utiliser pour retourner des données via le protocole du service. Ceci indique au fournisseur de retourner les résultats des opérations dans des lots de la taille précisée, pour qu'il puisse optimiser ses performances et l'utilisation des ressources. La propriété est indiquée par une chaîne qui représente un entier. En l'absence d'indication, la taille du traitement par lot est déterminée par le fournisseur.

CTX_DNS_URL

Constante contenant le nom de la propriété d'environnement permettant d'indiquer les noms de domaine et d'hôte DNS à utiliser pour le contexte de l'URL JNDI (par exemple, "dns://somehost/wiz.com"). Si la propriété n'est pas précisée dans l'environnement, la propriété système portant le même nom est utilisée. Si elle n'est pas non plus précisée dans une propriété système et si le programme tente d'utiliser une URL JNDI contenant le nom DNS, une exception ConfigurationException est levée.

CTX_InitialContextFactory

Nom de la classe de fabrique utilisé pour établir un contexte JNDI initial.

Valeur par défaut = none.

CTX_Language

Constante contenant le nom de la propriété d'environnement permettant d'indiquer la langue à utiliser de préférence avec le service. La valeur de la propriété est une liste de balises de langues définies dans RFC 1766 et séparées par le caractère ":". En l'absence d'indication, la langue est déterminée par le fournisseur de service.

CTX_ObjectFactories
Constante contenant le nom de la propriété d'environnement permettant d'indiquer la liste des fabriques d'objets à utiliser. La valeur de la propriété doit être une liste de noms complets de classes de fabriques, séparés par le caractère ":", qui créeront un objet d'après des informations le concernant. Si la propriété n'est pas précisée dans l'environnement, la propriété système portant le même nom est utilisée. Si elle n'est pas précisée non plus en tant que propriété système, NamingManager.getObjectInstance() tentera de créer l'objet.

CTX_ProviderURL
URL complète identifiant le contexte JNDI dans lequel se situe le facteur de connexion. Cette valeur est transmise au facteur de contexte.

Valeur par défaut = none.

CTX_Referral
Constante contenant le nom de la propriété d'environnement permettant d'indiquer de quelle façon les référenceurs rencontrés par le fournisseur de service doivent être traités. La valeur de la propriété est l'une des chaînes suivantes :

- follow
- follow referrals automatically
- ignore
- ignore referrals
- throw
- throw ReferralException when a referral is encountered
- En l'absence d'indication, la valeur par défaut est déterminée par le fournisseur.

CTX_SecurityAuthentication
Constante contenant le nom de la propriété d'environnement permettant d'indiquer le niveau de sécurité à utiliser. La valeur est l'une des chaînes "none", "simple" et "strong". En l'absence d'indication, le comportement est déterminé par le fournisseur de service.

CTX_SecurityCredentials
Constante contenant le nom de la propriété d'environnement permettant d'indiquer les accréditations de l'entité pour authentifier l'appelant au service. La valeur de la propriété dépend de la méthode d'authentification. Par exemple, il peut s'agir d'un mot de passe haché, d'un mot de passe en texte clair, d'une clé, d'un certificat, etc. En l'absence d'indication, le comportement est déterminé par le fournisseur de service.

CTX_SecurityPrincipal
Constante contenant le nom de la propriété d'environnement permettant d'indiquer l'identité de l'entité pour authentifier l'appelant au service. La valeur de la propriété dépend de la méthode d'authentification. En l'absence d'indication, le comportement est déterminé par le fournisseur de service.

CTX_SecurityProtocol
Constante contenant le nom de la propriété d'environnement permettant d'indiquer le protocole de sécurité à utiliser. Sa valeur est une chaîne déterminée par le fournisseur de service (par exemple "ssl"). En l'absence d'indication, le comportement est déterminé par le fournisseur de service.

CTX_URLPackagePrefixes
Constante contenant le nom de la propriété d'environnement permettant d'indiquer la liste des préfixes de modules à utiliser lors du chargement dans des fabriques de contexte d'URL. La valeur de la propriété doit être une liste (séparée par le caractère ":") de préfixes de modules pour le nom de classe de la classe de fabrique qui créera une fabrique de contexte d'URL. Si la propriété n'est pas précisée dans l'environnement, la propriété système portant le même nom est utilisée. Le préfixe com.sun.jndi.url est toujours ajouté à la fin de la liste potentiellement vide des préfixes de modules.

DataHandlerClassName
Classe de gestionnaire de données à utiliser lors de la conversion de messages depuis et vers des objets de données. Précisez à la fois DataHandlerConfigMO et DataHandlerMimeType, ou uniquement DataHandlerClassName. N'indiquez pas les trois propriétés à la fois.

Remarque : Une valeur DataHandlerClassName définie dans un méta-objet statique ou dynamique prévaut sur celle précisée dans cette propriété de configuration du connecteur. Si vous ne fournissez pas de valeur DataHandlerClassName dans un méta-objet, alors le connecteur se procure la valeur auprès de cette propriété de configuration du connecteur.

Valeur par défaut = none.

DataHandlerConfigMO
Nom du méta-objet contenant des informations de configuration pour le type mime précisé dans la propriété DataHandlerMimeType. Fournit des informations de configuration pour le gestionnaire de données. Précisez à la fois DataHandlerConfigMO et DataHandlerMimeType, ou uniquement DataHandlerClassName. N'indiquez pas les trois propriétés à la fois.

Remarque : Une valeur DataHandlerConfigMO définie dans un méta-objet statique ou dynamique prévaut sur celle précisée dans cette propriété de configuration du connecteur. Si vous ne fournissez pas de valeur DataHandlerConfigMO dans un méta-objet, le connecteur se procure la valeur auprès de cette propriété de configuration du connecteur.

La valeur par défaut est `MO_DataHandler_Default`.

DataHandlerMimeType
Permet de lancer une requête sur un gestionnaire de données en fonction d'un type MIME particulier. Précisez à la fois DataHandlerConfigMO et DataHandlerMimeType, ou uniquement DataHandlerClassName. N'indiquez pas les trois propriétés à la fois.

Remarque : Une valeur DataHandlerMimeType définie dans un méta-objet statique ou dynamique prévaut sur celle précisée dans cette propriété de configuration du connecteur. Si vous ne fournissez pas de valeur DataHandlerMimeType dans un méta-objet, le connecteur se procure la valeur auprès de cette propriété de configuration du connecteur.

Valeur par défaut = `text/delimited`.

DataHandlerPoolSize
Définissez cette propriété facultative si vous souhaitez que l'adaptateur mette en pool les instances de gestionnaires de données précédemment créées, en vue d'une utilisation ultérieure. Nombre maximum d'instances DataHandler à mettre en mémoire cache pour un type particulier de DataHandler.

Valeur par défaut =30

DefaultVerb
Indique l'instruction à définir dans une objet métier entrant, si elle n'a pas été définie par le gestionnaire de données pendant l'interrogation.

Valeur par défaut = Create

EnableMessageProducerCache
Propriété booléenne permettant de préciser que l'adaptateur doit activer la mémoire cache d'un producteur de message pour envoyer des messages de requête.

Valeur par défaut = true

ErrorDestination
Destination des copies des messages entrants lorsque le connecteur rencontre des erreurs de traitement.

La valeur par défaut est CWLD_ERROR.

InDoubtEvents
Indique comment gérer les événements en cours qui ne sont pas totalement traités en raison d'une panne inattendue de connecteur. Choisissez l'une des quatre actions à entreprendre si des événements figurent dans la file d'attente en cours pendant l'initialisation :

- FailOnStartup – Consigner l'erreur et arrêter immédiatement.
- Reprocess – Traiter d'abord les événements restants, puis traiter les messages de la file d'attente d'entrée.
- Ignore – Ignorer les messages de la file d'attente en cours.
- LogError – Consigner l'erreur mais ne pas arrêter

La valeur par défaut est Reprocess.

Remarque : Si vous configurez la propriété InProgressDestination, vous devez donner une valeur à cette propriété.

InProgressDestination
Destination temporaire contenant les messages pendant le traitement.

Valeur par défaut = none.

InputDestination
Destinations qui seront interrogées par le connecteur pour détecter les nouveaux messages. Le connecteur accepte plusieurs noms, séparés par un point-virgule. Par exemple, pour interroger les trois files d'attente suivantes dans une configuration de ce type : MyQueueA, MyQueueB et MyQueueC, la propriété de configuration du connecteur *InputQueue* doit être définie sur : MyQueueA;MyQueueB;MyQueueC.

Si la propriété InputDestination n'est pas indiquée, le connecteur ne sera pas interrogé.

Valeur par défaut = none.

LookupDestinationsUsingJNDI

Si cette propriété a la valeur true, le connecteur cherche tous les noms de destination JMS du magasin JNDI. Pour cela, toute destination précisée est définie dans le magasin JNDI.

Par défaut, le connecteur saute cette étape et permet au fournisseur JMS de résoudre le nom sur la destination appropriée lors de l'exécution.

Valeur par défaut = false.

MessageFormatProperty

Zone d'un message JMS contenant le format d'entrée ou de sortie du message. Par défaut, le connecteur contrôle la zone JMSType des messages entrants pour connaître le format du message, et il l'indique dans la zone JMSType des messages sortants.

Valeur par défaut =JMSType.

MessageResponseResultProperty

Obligatoire pour le traitement synchrone des requêtes, cette propriété précise dans un message JMS de réponse la zone que le connecteur doit contrôler pour déterminer le résultat de la requête. Cette propriété n'est pas utilisée pour le traitement asynchrone.

Si la propriété de l'en-tête JMS spécifiée par MessageResponseResultProperty n'existe pas, le connecteur interprète le code de retour en tant que VALCHANGE, transmet le contenu du message de réponse au gestionnaire de données puis met à jour l'objet métier.

La valeur par défaut est WBI_Result.

PollQuantity

Nombre maximum de messages à extraire de chaque destination précisée par la propriété InputDestination pendant un cycle pollForEvents.

La valeur par défaut est 1.

ReplyToDestination

Destination des fichiers de réponse lorsque le connecteur émet des requêtes. Il s'agit par défaut de la destination utilisée par le connecteur pour coordonner l'échange des messages de requête avec l'application cible. Ne précisez cette propriété que pour le traitement synchrone.

Valeur par défaut = none.

SessionPoolSizeForRequests

Taille de pool maximale pour la mise en mémoire cache des sessions utilisées au cours du traitement de la requête.

Valeur par défaut = 10

UnsubscribedDestination
Destination des copies des messages entrants si le message n'est pas reconnu ou si l'objet métier auquel il est mappé n'est pas pris en charge. Si cette propriété est définie et valide, le connecteur place une copie des messages non souscrits dans cette destination. Dans le cas contraire, le message est éliminé.

Valeur par défaut = none.

UnsubscribeOnTerminate
N'est utilisé que si UserDurableSubscriptions est défini sur `true`. L'utilisation de souscriptions durables crée un problème si vous supprimez des rubriques de la configuration du connecteur. Le fournisseur JMS continuera à conserver des messages pour les souscriptions durables, même si le connecteur ne contrôle plus jamais ces souscriptions.

Lorsque vous supprimez des rubriques de la liste précisée dans InputDestination, indiquez-les (en les séparant par un point-virgule) pour cette valeur de propriété. Pour détruire les souscriptions durables, procédez comme suit :

1. Déplacez depuis InputDestination vers UnsubscribeOnTerminate les noms des rubriques auxquelles vous ne voulez plus souscrire.
2. Démarrez et arrêtez le connecteur (ceci détruit les souscriptions durables).
3. Effacez les rubriques précisées pour UnsubscribeOnTerminate.

Cette action ne modifie aucune valeur de InputDestination.

A défaut d'effectuer la procédure ci-dessus, le connecteur n'est pas perturbé mais le fournisseur JMS de messages stockera des messages inutiles.

Valeur par défaut = none.

UseDefaults
Dans une opération Create, si UseDefaults est défini sur `true`, le connecteur contrôle si une valeur valide ou une valeur par défaut est fournie pour chaque attribut d'objet métier isRequired. Si une valeur est indiquée, l'opération Create réussit. Si le paramètre est défini sur false, le contrôleur cherche uniquement une valeur valide et entraîne l'échec de l'opération Create s'il n'en trouve pas.

Valeur par défaut = `false`.

UseDurableSubscriptions
N'utilisez cette propriété qu'avec les messages de type rubrique Pub/Sub. Si cette propriété est définie sur `true`, le connecteur agit comme un souscripteur durable pour les destinations applicables. Au prix d'une surcharge supérieure, le connecteur indiquera au fournisseur JMS de conserver tous les messages des rubriques auxquelles il souscrit, même lorsque le connecteur est hors ligne. Une fois ramené en ligne, le connecteur traitera de nouveau les messages publiés qu'il a manqués.

Valeur par défaut = `false`.

WorkerThreadCount
Nombre maximum d'unités d'exécution parallèles à interroger. Pendant le traitement simultané des événements, l'adaptateur est dans l'incapacité de soumettre des événements au courtier dans l'ordre dans lequel il les a reçus. Si la séquence doit être maintenue, WorkerThreadCount doit toujours être défini sur 1.

Configuration du style de message

L'adaptateur prend en charge les interfaces de messagerie point à point (PTP) et de publication/souscription (Pub/Sub) définies par le standard JMS. Le style de messagerie utilisé par l'adaptateur est déterminé par le type d'objet administré précisé par l'utilisateur dans la propriété ConnectionFactoryName spécifique au connecteur. Voir «ConnectionFactoryName», à la page 23 avant de poursuivre avec les procédures suivantes :

- «Configuration du style de message PTP»
- «Configuration du style Pub/Sub»

Configuration du style de message PTP

Pour configurer une instance de l'adaptateur dans le style de message PTP :

1. Ouvrez Connector Configurator.
2. Cliquez sur l'onglet Connector-Specific Properties.
3. Précisez un nom pour le ConnectionFactoryName qui se mappe sur une instance d'un *Queue*`ConnectionFactory` de votre magasin JNDI. L'adaptateur fonctionnera en style PTP et supposera que toutes les propriétés de connecteur et de méta-objet identifiant des destinations (par exemple la propriété OutputDestination) représentent des files d'attente.

Configuration du style Pub/Sub

Pour configurer une instance de l'adaptateur dans le style de message Pub/Sub :

1. Ouvrez Connector Configurator.
2. Cliquez sur l'onglet Connector-Specific Properties.
3. Précisez un nom pour le ConnectionFactoryName qui se mappe sur une instance d'un *Topic*`ConnectionFactory` de votre magasin JNDI. L'adaptateur fonctionnera en style publication/souscription et supposera que toutes les propriétés de connecteur et de méta-objet identifiant des destinations (par exemple la propriété OutputDestination) représentent des rubriques.

Configuration de JNDI

Pour établir une connexion avec le fournisseur JMS, le connecteur doit accéder à une fabrique de connexion JMS. JMS définit l'interface de la fabrique mais chaque fournisseur JMS doit fournir sa propre implémentation. Une fois que le connecteur a une référence à cette implémentation de fabrique, il peut établir une connexion et communiquer avec le fournisseur JMS sans connaître les protocoles propriétaires ni même l'identité du fournisseur.

Pour être portable, le connecteur a besoin que la fabrique de connexions réside dans un magasin JNDI. Pendant l'implémentation, l'utilisateur ou l'administrateur système doit créer et configurer une fabrique de connexions et la placer dans le magasin JNDI, sous un nom défini par l'utilisateur. Lors de l'exécution, le connecteur se connectera au magasin JNDI, recherchera la fabrique de connexions et l'utilisera pour établir une connexion auprès du fournisseur JMS.

Certains fournisseurs JMS apportent leurs propres implémentations JNDI contenant les fabriques et connexions et autres objets JMS administrés que vous créez. Cette approche simplifie la configuration de l'adaptateur JMS. Pour d'autres fournisseurs JMS, les utilisateurs devront peut-être installer et configurer un fournisseur JNDI externe, créer la fabrique de connexions et la mettre à la disposition de l'adaptateur. Pour plus d'informations, voir la documentation sur le fournisseur JNDI.

Pour plus d'informations sur la configuration et les variables de l'environnement JNDI, voir www.javasoft.com. Pour plus d'informations sur la configuration de JNDI avec le module de correction MA88, voir «Configuration de JNDI avec les bibliothèques du client WebSphere MQ Java».

Configuration de JNDI avec les bibliothèques du client WebSphere MQ Java

Pour accéder à un didacticiel indiquant comment configurer JNDI avec les bibliothèques du client WebSphere MQ Java, voir «Configuration pour la messagerie basée sur des files d'attente», à la page 105 et «Configuration de la messagerie basée sur des rubriques», à la page 106.

Configuration des méta-objets

Le connecteur pour JMS peut reconnaître et lire deux types de méta-objets :

- un méta-objet de connecteur statique
- un méta-objet enfant dynamique

Les valeurs d'attribut du méta-objet enfant dynamique dupliquent et supplantent celles du méta-objet statique. Pour une présentation des métadonnées et de la différence entre méta-objets statiques et dynamiques, voir «Métadonnées et méta-objets», à la page 8.

Pour déterminer le méta-objet le mieux adapté à votre implémentation, tenez compte des points suivants :

- **Méta-objet statique**
  - Utile si toutes les métadonnées des différents messages sont fixes et peuvent être précisées lors de la configuration.
  - Vous n'avez qu'à indiquer les valeurs par type d'objet métier. Par exemple, tous les objets de type Customer doivent être envoyés à la même destination.
- **Méta-objet dynamique**
  - Permet aux processus métier d'accéder aux informations des en-têtes de messages
  - Permet aux processus métier de modifier le traitement des messages lors de l'exécution, quel que soit le type métier. Par exemple, un méta-objet dynamique vous permettrait de spécifier une destination différente pour chaque type d'objet Customer envoyé à l'adaptateur.
  - Exige de modifier la structure des objets métier pris en charge. De tels changements peuvent nécessiter de modifier les mappes et processus métier.
  - Exige de modifier les gestionnaires de données personnalisés.

Propriétés du méta-objet

Le tableau 10 fournit une liste complète des propriétés prises en charge dans les méta-objets. Reportez-vous à ces propriétés lors de l'implémentation de méta-objets.

Toutes les propriétés ne sont pas disponibles pour les deux types d'objets. Elles n'ont pas non plus toutes la possibilité d'être lues ou écrites dans l'en-tête de message. Pour déterminer comment une propriété spécifique est interprétée et utilisée par le connecteur, voir les sections appropriées sur le traitement des requêtes et des événements au Chapitre 1, «Présentation de Adapter for JMS», à la page 1.

Tableau 10. Propriétés de méta-objet JMS

Nom de la propriété	Définissable dans un méta-objet statique	Définissable dans un méta-objet dynamique	Description
DataHandlerConfigMO	Oui	Oui	Méta-objet transmis au gestionnaire de données pour fournir des informations de configuration. Si elle est indiquée dans le méta-objet statique, cette propriété remplacera la valeur précisée dans la propriété du connecteur DataHandlerConfigMO. Utilisez cette propriété de méta-objet statique lorsque différents gestionnaires de données sont requis pour traiter différents types d'objet métier. Utilisez le méta-objet enfant dynamique pour traiter la requête lorsque le format des données peut dépendre des données métier réelles. L'objet métier précisé doit être pris en charge par l'agent du connecteur. Voir la description dans «Configuration des propriétés spécifiques au connecteur», à la page 20.
DataHandlerMimeType	Oui	Oui	Permet de lancer une requête sur un gestionnaire de données, en fonction d'un type MIME particulier. Si elle est indiquée dans le méta-objet statique, cette propriété remplacera la valeur précisée dans la propriété du connecteur DataHandlerMimeType. Utilisez cette propriété de méta-objet statique lorsque différents gestionnaires de données sont requis pour traiter différents types d'objet métier. Utilisez le méta-objet enfant dynamique pour traiter la requête lorsque le format des données peut dépendre des données métier réelles. L'objet métier précisé dans DataHandlerConfigMO doit avoir un attribut correspondant à la valeur de cette propriété. Voir la description dans «Configuration des propriétés spécifiques au connecteur», à la page 20.
DataHandlerClassName	Oui	Oui	Voir la description dans «Configuration des propriétés spécifiques au connecteur», à la page 20.
InputFormat	Oui	Oui	Format ou type des messages (événement) entrants. Cette valeur aide à identifier le contenu du message. Elle est précisée par l'application qui a généré le message. La zone que le connecteur considère comme définissant le format dans le message peut être définie par l'utilisateur via la propriété MessageFormatProperty spécifique au connecteur.
OutputFormat	Oui	Oui	Format à renseigner dans les messages sortants. Si OutputFormat n'est pas précisé, le format d'entrée est utilisé, s'il est disponible.

Tableau 10. Propriétés de méta-objet JMS (suite)

Nom de la propriété	Définissable dans un méta-objet statique	Définissable dans un méta-objet dynamique	Description
`InputDestination`	Oui	Oui	Cette propriété est utilisée pour faire correspondre les messages entrants et les objets métier uniquement. A l'inverse, la propriété InputDestination spécifique au connecteur définit les destinations interrogées par l'adaptateur. C'est la seule propriété utilisée par l'adaptateur pour déterminer les destinations à interroger. Dans l'objet géré, les propriétés InputDestination et InputFormat peuvent servir de critère pour que l'adaptateur mappe un message donné sur un objet métier spécifique. Pour mettre en oeuvre cette fonctionnalité, utilisez des propriétés spécifiques au connecteur afin de configurer plusieurs destinations d'entrées, et éventuellement de mapper différents gestionnaires de données sur chacune en fonction des formats d'entrée des messages entrants. Ne définissez pas cette propriété à l'aide des propriétés de conversion par défaut. C'est sa valeur qui est utilisée.
`OutputDestination`	Oui	Oui	Destination où sont écrits les messages sortants.
`ResponseTimeout`	Oui	Oui	Indique la durée d'attente d'un réponse avant expiration (en millisecondes), dans un traitement de requête synchrone. Si cette propriété n'est pas précisée ou si elle contient une valeur inférieure à zéro, le connecteur retourne SUCCESS immédiatement, sans attendre de réponse.
`DataEncoding`	Oui	Oui	`DataEncoding` indique si les données du messages sont traitées par l'adaptateur comme étant de type texte, binaire ou objet. Si cette propriété n'est pas spécifiée dans le méta-objet statique, le connecteur tente de lire les messages sans utiliser de code spécifique. La propriété `DataEncoding` définie dans un méta-objet enfant dynamique remplace la valeur définie dans le méta-objet statique. La valeur par défaut est `Text`. Le format de la valeur de cet attribut est `messageType[:enc]`. Par exemple `Text:ISO8859_1`, `Text:UnicodeLittle`, `Text`, `Object` ou `Binary`. Cette propriété est liée en interne à la propriété InputFormat : vous précisez un et un seul `DataEncoding per InputFormat`. Pour plus d'informations sur DataEncoding, voir «DataEncoding», à la page 34.
Ci-dessous figurent des zones mappées spécifiquement sur l'en-tête du message JMS. Pour obtenir des explications précises, savoir comment interpréter les valeurs et etc., voir les spécifications de l'API JMS. Il est possible que les fournisseurs JMS interprètent certaines zones de façon différente. Par conséquent, consultez également la documentation de votre fournisseur JMS pour connaître les éventuelles différences.			
ReplyToDestination		Oui	Destination à laquelle doit être envoyé un message de réponse à une requête.
Type		Oui	Type de message. Généralement défini par l'utilisateur, en fonction du fournisseur JMS.
MessageID		Oui	ID unique du message (spécifique au fournisseur JMS).
CorrelationID	Oui	Oui	Utilisé dans les messages de réponse pour indiquer l'ID du message de requête qui a émis cette réponse.

Tableau 10. Propriétés de méta-objet JMS (suite)

Nom de la propriété	Définissable dans un méta-objet statique	Définissable dans un méta-objet dynamique	Description
Delivery Mode	Oui	Oui	Indique si le message est conservé ou non dans le système MOM. Valeurs acceptées : 1 = non persistant 2 = persistant D'autres valeurs peuvent exister selon le fournisseur JMS.
Priority (Priorité)		Oui	Priorité numérique du message. Valeurs acceptées : 0 à 9 inclus (priorité faible à élevée).
Destination		Oui	Emplacement en cours ou dernier emplacement (en cas de suppression) dans un système MOM.
Expiration		Oui	Durée de vie du message.
Redelivered		Oui	Indique que le fournisseur JMS a très probablement déjà tenté de livrer le message au client, mais qu'aucun accusé de réception n'a été reçu.
Timestamp		Oui	Heure à laquelle le message a été distribué au fournisseur JMS.
UserID		Oui	Identité de l'émetteur du message.
AppID		Oui	Identité de l'application qui envoie le message.
DeliveryCount		Oui	Nombre de tentatives de livraison.
GroupID		Oui	Identité du groupe de messages.
GroupSeq		Oui	Séquence de ce message dans le groupe de messages indiqué par GroupID.
JMSProperties		Oui	Voir «Propriétés JMS», à la page 41.

DataEncoding

Le standard JMS définit différents types de messages permettant de conserver différents types de données. Par exemple, un message texte peut contenir un document XML, tandis que des données binaires peuvent contenir une image. Le fait de préciser de façon explicite le codage attendu des données, par le biais de la propriété de méta-objet `DataEncoding`, aide l'adaptateur à savoir quels types de messages il doit attendre et comment il doit les traiter.

Par défaut, l'adaptateur suppose que tous les messages sont de type texte. Toutefois, l'adaptateur pour JMS prend en charge les messages de types texte, binaire et objet Java. Pendant le traitement de la requête, l'adaptateur utilise le gestionnaire de données configuré pour convertir l'objet métier de la requête en texte, qu'il livre ensuite sous forme de message texte. Pendant la notification d'événement, l'adaptateur extrait le texte d'un message et le transmet au gestionnaire de données pour créer l'objet métier de l'événement. Si l'adaptateur reçoit un message binaire, il convertit le corps binaire en texte à l'aide du codage par défaut du JVM, avant de transmettre le contenu au gestionnaire de données.

Dans certains cas, ces procédures ne sont pas appropriés. Si vous voulez traiter des messages binaires et qie votre gestionnaire de données est conçu pour les données binaires, vous devez utiliser la propriété `DataEncoding`. Cette propriété accepte une des trois valeurs possibles suivantes :

- `text` : Valeur par défaut.
- `binary` : Si vous choisissez `binary`, l'adaptateur a un comportement différent :
  1. Pendant le traitement de la requête, l'adaptateur transmet l'objet métier aux méthodes `binary` du gestionnaire de données et fournit un message en octets.

2. Pendant la notification d'événement, l'adaptateur extrait les octets du message binaire et les transmet au gestionnaire de données en tant qu'instance Java `InputStream` (octets).
3. Si l'adaptateur reçoit un message texte, il convertit le corps du texte en binaire à l'aide du codage par défaut du JVM, avant de transmettre le contenu au gestionnaire de données.

Pour les messages binaires, vous devez définir `DataEncoding=binary`

Remarque : Tous les gestionnaires de données ne prennent pas en charge les données binaires. Vérifiez donc votre gestionnaire de données configuré pour connaître les limites de prise en charge.

- `object` : Si vous choisissez `object`, l'adaptateur a un comportement différent :
  1. Pendant le traitement de la requête, l'adaptateur transmet l'objet métier à la méthode `getStreamFromBO` du gestionnaire de données, pour obtenir un `InputStream` .
  2. Pendant la notification d'événement, l'adaptateur extrait les objets Java du message objet et les transmet au gestionnaire de données, en tant que `InputGetStream`.

  Pour les messages de type objet, définissez `DataEncoding=object`

  Remarque : Tous les gestionnaires de données ne prennent pas en charge les données d'objet Java. Vérifiez donc votre gestionnaire de données configuré pour connaître les limites de prise en charge.

Lorsque c'est possible, il est recommandé d'envoyer les données de contexte dans des messages texte JMS plutôt que d'obliger l'adaptateur à gérer la conversion du format texte au format binaire (ou vice-versa). Un codage incorrect pouvant altérer subtilement les données, il est difficile de le détecter ou de le diagnostiquer. Toutefois, il est possible de préciser comment vous souhaitez que l'adaptateur effectue la conversion entre les format binaire et texte (ou vice-versa), à l'aide d'un modificateur supplémentaire indiqué par la propriété `DataEncoding`. Si vous ajoutez un caractère ″:″ au nom d'un codage Java pris en charge, l'adaptateur utilisera le codage précisé pour la conversion. Le format est `text:encoding`.

Par exemple, `DataEncoding=text;ISO8859_1` informe l'adaptateur qu'en cas de réception d'un message binaire, il doit convertir le corps du message en texte à l'aide du codage `ISO8859_1`, avant de le transmettre au gestionnaire de données. Si le type de message correspond au codage des données, cette valeur n'aura aucun effet (par exemple, si le message binaire et le codage binaire indiquent binaire ou texte, et si le codage des données indique un type texte).

Conception du gestionnaire de données pour les données binaires : Si vous envisagez de concevoir et d'utiliser un gestionnaire de données personnalisé avec l'adaptateur, tenez compte des points suivants :

- Vous devez fournir une implémentation pour la méthode `getStreamFromBO` (BusinessObject, Object).
- Vous devez remplacer et fournir une implémentation pour les méthodes `getBO`(InputStream, Object) et `getBO` (InputStream, BusinessObject, Object).
- Des implémentations par défaut de ces méthodes sont fournies dans la classe DataHandler de base, mais elle ne font que convertir toute donnée binaire en texte, avant d'appeler les méthodes de texte correspondantes sur votre sous-classe de gestionnaire de données personnalisée (ce qui va à l'encontre du but de l'opération). Vous devez fournir vos propres implémentations pour que la prise en charge des données binaires fonctionne.

Configuration d'un méta-objet statique

Le méta-objet statique de configuration JMS contient une liste de propriétés de conversion définies pour différents objets métier. Pour afficher un exemple de méta-objet statique, lancez Business Object Designer et ouvrez l'exemple suivant fourni avec l'adaptateur : `connectors\JMS\Samples\Sample_JMS_MO_Config.xsd`.

A un moment donné, le connecteur ne prend en charge qu'un seul méta-objet statique. Vous implémentez un méta-objet statique en indiquant son nom dans la propriété du connecteur ConfigurationMetaObject.

La structure du méta-objet statique est telle que chaque attribut représente une combinaison unique d'objet métier et d'instruction, ainsi que toutes les méta-données associées au traitement de cet objet. Le nom de chaque attribut doit être le nom du type d'objet métier et l'instruction, séparés par un caractère souligné, par exemple `Customer_Create`. Les informations spécifiques à l'application de l'attribut doivent consister en une ou plusieurs paires nom-valeur, séparées par un caractère ";", et représentant les propriétés de méta-données que vous souhaitez préciser pour cette combinaison objet-instruction unique.

Tableau 11. Structure de méta-objet statique

Nom de l'attribut	Texte spécifique à l'application
`<business object type>_<verb>`	`property=value;property=value;...`
`<business object type>_<verb>`	`property=value;property=value;...`

Prenons l'exemple du méta-objet suivant

Tableau 12. Exemple de structure de méta-objet statique

Nom de l'attribut	Informations spécifiques à l'application
`Customer_Create`	`OutputFormat=CUST;OutputDestination=QueueA`
`Customer_Update`	`OutputFormat=CUST;OutputDestination=QueueB`
`Order_Create`	`OutputFormat=ORDER;OutputDestination=QueueC`

Le méta-objet de cet exemple informe le connecteur que lorsqu'il reçoit un objet métier de requête de type `Customer` avec une instruction `Create`, il doit le convertir en un message de format `CUST`, et le placer dans la destination `QueueA`. Si par contre l'objet client avait une instruction `Update`, le message serait placé dans `QueueB`. Si le type d'objet était `Order` et l'instruction `Create`, le connecteur convertirait l'objet et le livrerait à `QueueC`, au format `ORDER`. Tout autre objet métier transmis au connecteur serait traité comme étant non souscrit.

Si vous le souhaitez, vous pouvez nommer un attribut `Default` et lui affecter une ou plusieurs propriétés dans l'ASI. Pour tous les attributs contenus dans le méta-objet, les propriétés de l'attribut par défaut sont combinées avec celles des attributs objet-instruction spécifiques. Ceci s'avère utile lorsque vous appliquez une ou plusieurs propriétés de façon universelle (quelle que soit la combinaison objet-instruction).

Dans l'exemple suivant, le connecteur considérerait les combinaisons objet-instruction `Customer_Update` et `Order_Create` comme ayant `OutputDestination=QueueA`, en plus de leurs propriétés de méta-données individuelles :

Tableau 13. Exemple de structure de méta-objet statique

Nom de l'attribut	Informations spécifiques à l'application
`Valeur par défaut`	`OutputDestination=QueueA`
`Customer_Update`	`OutputFormat=CUST`
`Order_Create`	`OutputFormat=ORDER`

Voir tableau 10, à la page 32 dans «Propriétés du méta-objet», à la page 31, qui décrit les propriétés disponibles en tant qu'informations spécifiques à l'application dans le méta-objet statique.

Pour implémenter un méta-objet statique :

1. Lancez Business Object Designer. Pour plus d'informations, voir *Business Object Development Guide*.
2. Ouvrez l'exemple de méta-objet `connectors\JMS\Samples\Sample_JMS_MO_Config.xsd`.

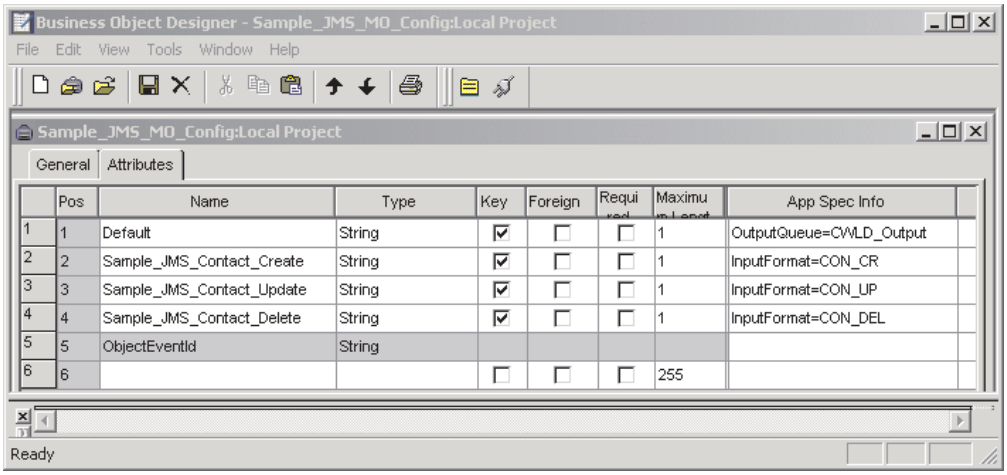


Figure 3. Exemple de méta-objet statique

3. Modifiez les attributs et l'ASI en fonction de vos besoins, en consultant le tableau 10, à la page 32, puis enregistrez le fichier du méta-objet.
4. Précisez le nom de ce fichier de méta-objet comme étant la valeur de la propriété du connecteur ConfigurationMetaObject.

Mappage des gestionnaires de données sur les destinations d'entrée

Vous pouvez utiliser la propriété InputDestination dans l'ASI du méta-objet statique, pour associer un gestionnaire de données à une destination d'entrée. Cette fonctionnalité est utile pour travailler avec plusieurs partenaires d'échange dont les exigences en matière de format et de conversion diffèrent.

Pour mapper un gestionnaire de données sur une destination d'entrée :

1. Lancez Connector Configurator. Pour plus d'informations, voir l'Annexe B, «Connector Configurator», à la page 79.

2. Utilisez les propriétés spécifiques au connecteur (voir «InputDestination», à la page 27) pour configurer une ou plusieurs destinations d'entrée. Les noms de destination doivent être séparés par un caractère ";".
3. Pour chaque destination d'entrée, précisez dans l'ASI la destination (gestionnaire de file d'attente si vous implémentez une messagerie PTP) et un nom de destination d'entrée, ainsi que le nom de classe du gestionnaire de données et le type mime.

Par exemple, l'attribut suivant dans un méta-objet statique associe un gestionnaire de données à un InputDestination nommé CompReceipts :

```
[Attribute]
Name = Customer_Create
Type = String
Cardinality = 1
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo =
InputDestination=//queue.manager/CompReceipts;DataHandlerClassName=com.crossworlds.
DataHandlers.MQ.disposition_notification;DataHandlerMimeType=message/
disposition_notification
IsRequiredServerBound = false
[End]
```

Configuration d'un méta-objet enfant dynamique

S'il est difficile ou infaisable de préciser les méta-données nécessaires via un méta-objet statique, le connecteur a la possibilité d'accepter les métadonnées livrées lors de l'exécution pour chaque instance d'objet métier.

Les méta-objet dynamiques vous permettent de modifier les méta-données utilisées par le connecteur pour traiter un objet métier en fonction de chaque requête, lors du traitement de celle-ci, et d'extraire des informations sur un message d'événement pendant le traitement de l'événement.

La structure du méta-objet dynamique est telle que chaque attribut représente une propriété et une valeur de méta-données uniques :`meta-object property name =meta-object property value`

Pour implémenter un méta-objet dynamique, ajoutez-le en tant qu'enfant à votre objet de niveau supérieur et incluez la paire nom-valeur `cw_mo_conn=<MO attribute>` à votre ASI d'objet de niveau supérieur, où *<MO attribute>* est le nom de l'attribut de votre objet de niveau supérieur représentant le méta-objet dynamique. Par exemple :

```
Customer (ASI = cw_mo_conn=MetaData)
  |-- Id
  |-- FirstName
  |-- LastName
  |-- ContactInfo
  |-- MetaData
        |-- OutputFormat = CUST
        |-- OutputDestination = QueueA
```

Dès réception d'une requête renseignée comme indiqué ci-dessus, le connecteur convertit l'objet `Customer` en un message au format `CUST` , puis le place dans la file d'attente `QueueA`.

Les objets métier peuvent utiliser des méta-objets dynamiques identiques ou différents, ou n'en utiliser aucun.

Remarque : Tous les gestionnaires de données IBM WebSphere standard sont conçus pour ignorer cet attribut de méta-objet dynamique en reconnaissant la balise `cw_mo_`. Vous devez procéder de la même façon lors du développement de gestionnaires de données personnalisés à utiliser avec l'adaptateur.

Le connecteur reconnaît et lit les propriétés de conversion à partir d'un méta-objet dynamique ajouté en tant qu'enfant à l'objet métier de niveau supérieur transmis au connecteur. Les valeurs d'attribut du méta-objet enfant dynamique dupliquent les propriétés de conversion que vous pouvez préciser par le biais du méta-objet statique utilisé pour configurer le connecteur.

Comme les propriétés de méta-objet enfant dynamique remplacent celles des méta-objets statiques, si vous spécifiez un méta-objet enfant dynamique, vous devez inclure une propriété de connecteur qui indique le méta-objet statique. Par conséquent, vous pouvez utiliser un méta-objet enfant dynamique indépendamment du méta-objet statique et vice-versa.

Voir tableau 10, à la page 32 dans «Propriétés du méta-objet», à la page 31, qui décrit les propriétés disponibles en tant qu'informations spécifiques à l'application dans le méta-objet dynamique.

Les attributs suivants, qui reflètent les propriétés de l'en-tête JMS, sont reconnues dans le méta-objet dynamique.

Tableau 14. Attributs d'en-tête du méta-objet dynamique

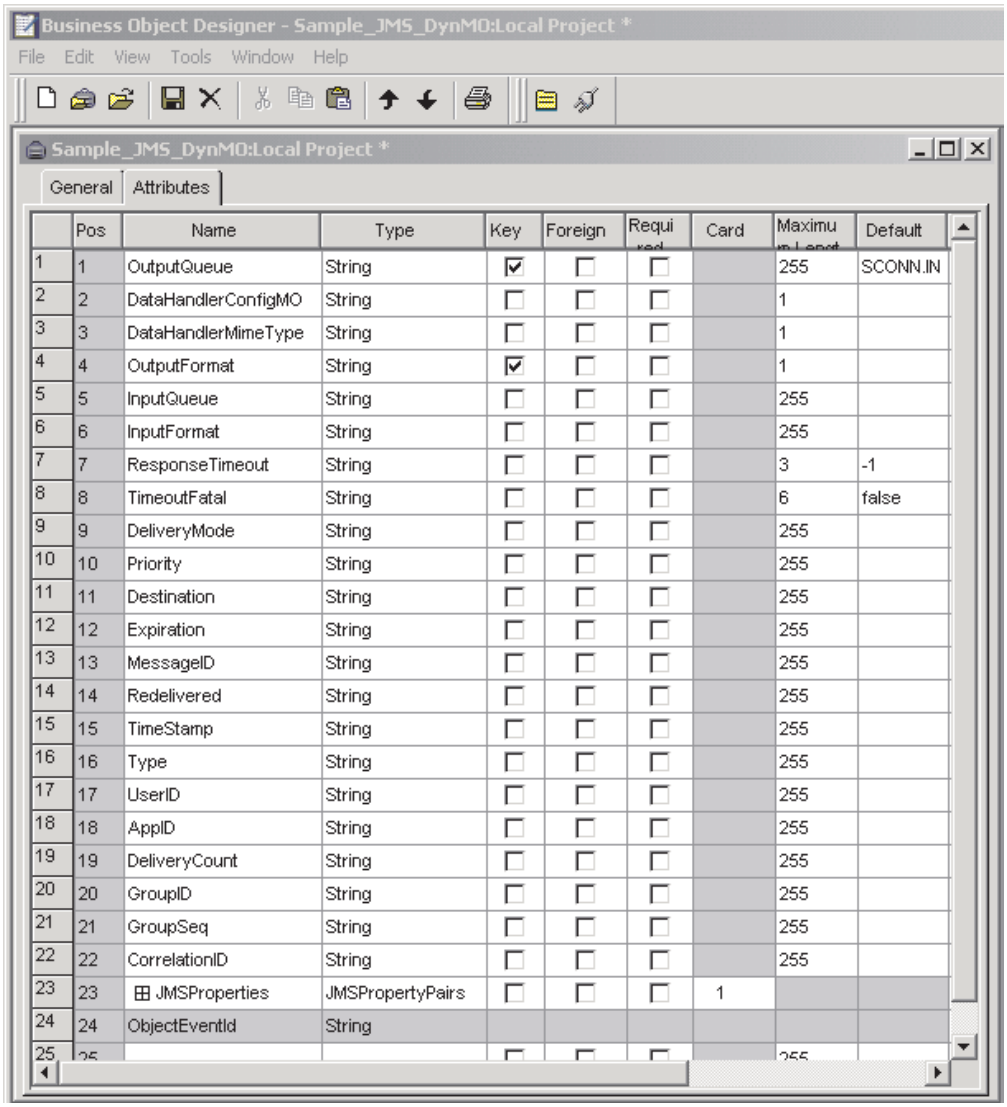
Nom de l'attribut d'en-tête	Mode	En-tête JMS correspondant
`CorrelationID`	Lecture/écriture	`JMSCorrelationID`
`ReplyToQueue`	Lecture/écriture	`JMSReplyTo`
`DeliveryMode`	Lecture/écriture	`JMSDeliveryMode`
`Priority (Priorité)`	Lecture/écriture	`JMSPriority`
`Destination`	Lecture	`JMSDestination`
`Expiration`	Lecture	`JMSExpiration`
`MessageID`	Lecture	`JMSMessageID`
`Redelivered`	Lecture	`JMSRedelivered`
`TimeStamp`	Lecture	`JMSTimeStamp`
`Type`	Lecture	`JMSType`
`UserID`	Lecture	`JMSXUserID`
`AppID`	Lecture	`JMSXAppID`
`DeliveryCount`	Lecture	`JMSXDeliveryCount`
`GroupID`	Lecture	`JMSXGroupID`
`GroupSeq`	Lecture	`JMSXGroupSeq`
`JMSProperties`	Lecture/écriture	

Les attributs en lecture seule sont lus à partir d'un en-tête de message pendant la notification d'événement, et écrits dans le méta-objet dynamique. Ces propriétés renseignent également le MO dynamique lorsqu'un message de réponse est émis

au cours du traitement d'une requête.Les attributs en lecture/écriture sont définis dans les en-tête de messages créés au cours du traitement de la requête. Pendant la notification d'événement, les attributs en lecture/écriture sont lus à partir des en-têtes de messages pour renseigner le méta-objet dynamique.

Pour implémenter un méta-objet dynamique :

1. Lancez Business Object Designer. Pour plus d'informations, voir *Business Object Development Guide*.
2. Ouvrez l'exemple de méta-objet `connectors\JMS\Samples\Sample_JMS_DynMO.xsd`.



	Pos	Name	Type	Key	Foreign	Requi	Card	Maximu	Default
1	1	OutputQueue	String	☑	☐	☐		255	SCONN.IN
2	2	DataHandlerConfigMO	String	☐	☐	☐		1	
3	3	DataHandlerMimeType	String	☐	☐	☐		1	
4	4	OutputFormat	String	☑	☐	☐		1	
5	5	InputQueue	String	☐	☐	☐		255	
6	6	InputFormat	String	☐	☐	☐		255	
7	7	ResponseTimeout	String	☐	☐	☐		3	-1
8	8	TimeoutFatal	String	☐	☐	☐		6	false
9	9	DeliveryMode	String	☐	☐	☐		255	
10	10	Priority	String	☐	☐	☐		255	
11	11	Destination	String	☐	☐	☐		255	
12	12	Expiration	String	☐	☐	☐		255	
13	13	MessageID	String	☐	☐	☐		255	
14	14	Redelivered	String	☐	☐	☐		255	
15	15	TimeStamp	String	☐	☐	☐		255	
16	16	Type	String	☐	☐	☐		255	
17	17	UserID	String	☐	☐	☐		255	
18	18	AppID	String	☐	☐	☐		255	
19	19	DeliveryCount	String	☐	☐	☐		255	
20	20	GroupID	String	☐	☐	☐		255	
21	21	GroupSeq	String	☐	☐	☐		255	
22	22	CorrelationID	String	☐	☐	☐		255	
23	23	⊞ JMSProperties	JMSPropertyPairs	☐	☐	☐	1		
24	24	ObjectEventId	String						
25				☐	☐	☐		255	

Figure 4. Exemple de méta-objet dynamique

3. Modifiez les attributs et propriétés en fonction de vos besoins pour cet objet métier, et enregistrez-le.
4. Ajoutez le méta-objet dynamique en tant qu'enfant à votre objet de niveau supérieur, et insérez une paire nom-valeur `cw_mo_conn=<MO attribute>` dans votre ASI d'objet de niveau supérieur, où *<MO attribute>* est le nom de l'attribut dans votre objet de niveau supérieur représentant le méta-objet dynamique.

Alimentation du méta-objet enfant dynamique pendant l'interrogation

Pour fournir des collaborations dotées d'informations plus nombreuses sur les messages extraits lors de l'interrogation, le connecteur renseigne des attributs spécifiques du méta-objet dynamique, s'ils ont déjà été définis pour l'objet métier créé.

Le tableau 15 montre comment structurer un méta-objet enfant dynamique en vue de l'interrogation.

Tableau 15. Structure de méta-objet enfant dynamique JMS pour interrogation

Nom de l'attribut	Exemple de valeur
InputFormat	CUST_IN
InputQueue	MYInputQueue
OutputFormat	CxIgnore
OutputQueue	CxIgnore
ResponseTimeout	CxIgnore
TimeoutFatal	CxIgnore

Comme indiqué au tableau 15, vous pouvez définir dans un méta-objet enfant dynamique les attributs supplémentaires Input_Format et Inputdestination. L'attribut Input_Format est renseigné avec le format du message extrait, tandis que l'attribut InputDestination contient le nom de la destination d'où a été extrait un message donné. Si ces propriétés ne sont pas définies dans le méta-objet enfant, elles ne sont pas renseignées.

Exemple de scénario :

- Le connecteur extrait un message dont le format est CUST_IN, depuis la file d'attente MyInputQueue.
- Le connecteur convertit ce message en un objet métier Customer et vérifie le texte spécifique à l'application pour déterminer si un méta-objet est défini.
- Si c'est le cas, le connecteur crée une instance de ce méta-objet et complète les attributs InputDestination et InputFormat en conséquence, puis publie l'objet métier dans les collaborations disponibles.

En-têtes JMS et attributs de méta-objet enfant dynamique

Vous pouvez ajouter des attributs à un méta-objet dynamique pour obtenir plus d'informations sur le transfert des messages, ainsi qu'un meilleur contrôle. La présente section décrit ces attributs et la façon dont ils affectent la notification d'événement et le traitement des requêtes.

Propriétés JMS : Contrairement aux autres attributs du méta-objet dynamique, JMSProperties doit définir un objet enfant à cardinalité unique. Chaque attribut de cet objet enfant doit définir une propriété unique, afin qu'elle soit lue/écrite dans la partie variable de l'en-tête de message JSM, de la façon suivante :

1. Le nom de l'attribut n'a aucune valeur sémantique.
2. Le type de l'attribut doit toujours être String, quel que soit le type de propriété JMS.
3. Les informations spécifiques à l'application de l'attribut doivent contenir deux paires nom-valeur, définissant le nom et le format de la propriété du message JMS auquel l'attribut est mappé. Le nom est défini par l'utilisateur. La valeur doit appartenir à un des types suivants :
   - Boolean

- String
- Int
- Float
- Double
- Long
- Short
- Byte

Le tableau ci-dessous montre les propriétés des informations spécifiques à l'application que vous devez définir pour les attributs de l'objet `JMSProperties`.

Tableau 16. Informations spécifiques à l'application des attributs de propriété JMS

Attribut	Valeurs possibles	ASI	Commentaires
Nom	Tout nom de propriété JMS valide (valide = compatible avec le type défini dans l'ASI)	`name=<`*`JMS property name`*`>;type=<`*`JMS property type`*`>`	Certains fournisseurs réservent certaines propriétés pour apporter une fonctionnalité étendue. En général, les utilisateurs ne doivent pas définir de propriétés personnalisées qui commencent par `JMS`, à moins qu'ils ne cherchent à accéder à ces fonctionnalités spécifiques au fournisseur.
Type	`String`	type=<*voir commentaires*>	Type de la propriété JMS. L'API JMS propose plusieurs méthodes pour définir des valeurs dans le message JMS : `setIntProperty`, `setLongProperty`, `setStringProperty`, etc. Le type de propriété JMS indiqué ici détermine laquelle de ces méthodes est utilisée pour paramétrer la valeur de propriété dans le message.

Dans l'exemple ci-dessous, un objet enfant JMSProperties est défini pour que l'objet Customer autorise l'accès aux zones définies par l'utilisateur de l'en-tête du message :

```
Customer (ASI = cw_mo_conn=MetaData)
  |-- Id
  |-- FirstName
  |-- LastName
  |-- ContactInfo
  |-- MetaData
```

```
|-- OutputFormat = CUST
|-- OutputDestination = QueueA
|-- JMSProperties
     |-- RoutingCode = 123 (ASI= name=RoutingCode;type=Int)
     |-- Dept = FD (ASI= name=RoutingDept;type=String)
```

Pour illustrer un autre exemple, la figure 5 montre l'attribut `JMSProperties` dans le méta-objet dynamique ainsi que des définitions pour quatre propriétés dans l'en-tête de message JMS : ID, GID, RESPONSE et RESPONSE_PERSIST. Les informations spécifiques à l'application de ces attributs définissent le nom et le type de chacun. Par exemple, l'ID d'attribut se mappe sur la propriété JMS `ID` de type `String`.

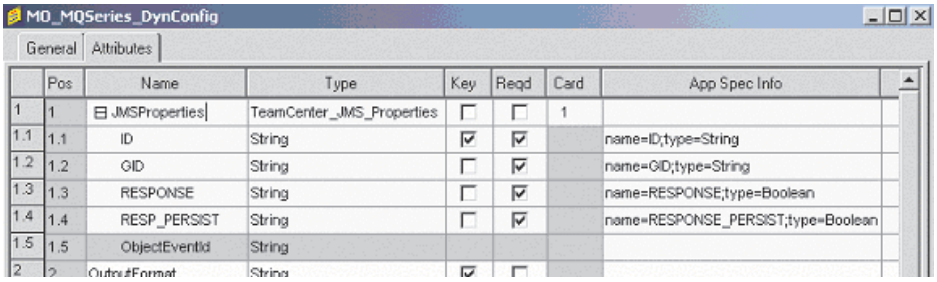


Figure 5. Attribut de propriétés JMS dans un méta-objet dynamique

Configuration des scripts de démarrage

Le connecteur est fourni avec les scripts de démarrage—`JMS.bat` ou `JMS.sh`, selon le système d'exploitation. Pour plus d'informations sur le démarrage et l'arrêt d'un connecteur, ainsi que sur le fichier journal de démarrage temporaire, voir le chapitre relatif au démarrage dans le *System Installation Guide* de votre plateforme.

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 43.
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 17.

Tableau 17. 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. Création ou modification d'objets métier

• «Structure des objets métier de connecteur»

Le connecteur est livré avec des exemples d'objets métier uniquement. L'intégrateur système, le consultant ou le client doit concevoir les objets métier utilisés.

Le présent chapitre décrit les exigences du connecteur en matière d'objets métier. Vous pouvez utiliser ces informations pour vous aider à implémenter de nouveaux objets métier.

Structure des objets métier de connecteur

Après avoir installé le connecteur, vous devez créer des objets métier. La structure des objets métier n'impose aucune exigence autre que celles dictées par le gestionnaire de données configuré. Les objets métier traités par le connecteur peuvent porter n'importe quel nom autorisé par le courtier d'intégration. Pour plus d'informations, consultez les conventions de dénomination applicables à votre courtier d'intégration.

Le connecteur extrait des messages d'une destination et tente de renseigner un objet métier (défini par le méta-objet) avec le contenu du message. Le connecteur n'a aucun contrôle ni aucune influence sur la structure de l'objet métier. Ces fonctions sont assurées par les définitions de méta-objet ainsi que par les exigences du gestionnaire de données du connecteur. En fait, il n'existe aucun texte d'application au niveau de l'objet métier. Par contre, le rôle principal du connecteur, lorsqu'il extrait et transmet des objets métier, est de rechercher des erreurs dans le processus message vers objet métier (et vice-versa).

Création d'objets métier

1. Identifiez et configurez l'application à laquelle le courtier d'intégration enverra des objets métier lorsque le courtier sera configuré avec une instance de l'adaptateur JMS.
2. Configurez le connecteur avec un gestionnaire de données capable de transformer les messages émis par des destinations JMS en objets métier pouvant être traités par l'application cible. Pour cela, précisez les propriétés de connecteur DataHandlerConfigMO et DataHandlerMimeType, ou spécifiez la propriété DataHandlerClassName. Pour plus d'informations, voir «Configuration des propriétés du connecteur», à la page 19. Vous pouvez préciser des règles de traitement de gestionnaire de données spéciales dans des méta-objets statiques et dynamiques. Pour plus d'informations, voir «Propriétés du méta-objet», à la page 31.
3. Utilisez Business Object Designer pour créer les objets métier spécifiques à l'application. Pour plus d'informations, voir le guide *Business Object Designer*.
4. Ajoutez les objets métier créés aux Supported Business Objects. Dans Connector Configurator, cliquez sur l'onglet **Supported Business Objects** de l'adaptateur JMS, ajoutez les objets métier créés et définissez **Message Set ID** sur une valeur unique pour chaque objet métier pris en charge. Pour plus d'informations sur l'utilisation de Connector Configurator pour ajouter des objets métier pris en charge, voir «Indication des définitions d'objets métier pris en charge», à la page 90.

Chapitre 4. Résolution des incidents

- «Gestion des erreurs»
- «Traçage», à la page 50
- «Résolution des incidents de démarrage», à la page 51

Le présent chapitre explique comment le connecteur gère les erreurs et le traçage, ainsi que les problèmes que vous êtes susceptible de rencontrer lorsque vous démarrez ou exécutez le connecteur.

Gestion des erreurs

Tous les messages d'erreur générés par le connecteur sont conservés dans un fichier nommé `JMSConnector.txt`. (Le nom du fichier est déterminé par la propriété standard de configuration du connecteur `LogFileName`.) Chaque erreur est associée à un numéro, suivi du message d'erreur :

Numéro du message
Texte du message

Le connecteur gère les erreurs spécifiques décrites dans les sections qui suivent.

Délai dépassé par l'application

Le message d'erreur APP_RESPONSETIMEOUT est retourné lorsque :

- Le connecteur ne peut établir de connexion au fournisseur de service JMS pendant l'extraction du message.
- Le connecteur parvient à convertir un objet métier en un message, mais ne peut le livrer à la file d'attente de sortie en raison d'une perte de connexion.
- Le connecteur émet un message mais dépasse le délai d'attente de réponse pour un objet métier dont la propriété de conversion `TimeoutFatal` est sur `True`.
- Le connecteur reçoit un message de réponse dont le code de retour est APP_RESPONSE_TIMEOUT ou UNABLE_TO_LOGIN.

Objet métier non souscrit

Si le connecteur extrait un message associé à un objet métier non souscrit, ou si un code NO_SUBSCRIPTION_FOUND est retourné par la méthode `gotApplEvent()`, le connecteur livre un message à la file d'attente indiquée par la propriété `UnsubscribedDestination`.

Remarque : Si `UnsubscribedDestination` n'est pas défini, les messages non souscrits sont éliminés.

Connecteur inactif

Lorsque la méthode `gotApplEvent()` retourne un code CONNECTOR_NOT_ACTIVE, la méthode `pollForEvents()` retourne un code APP_RESPONSE_TIMEOUT et l'événement reste dans InProgress Destination, si la destination est précisée.

Conversion du gestionnaire de données

Si le gestionnaire de données ne parvient pas à convertir un message en objet métier, ou en cas d'erreur de traitement spécifique à l'objet métier (par opposition au fournisseur JMS), le message est livré à la file d'attente précisée par `ErrorDestination`. Si `ErrorDestination` n'est pas défini, les messages qui ne peuvent être traités suite à des erreurs sont éliminés.

Si le gestionnaire de données ne parvient pas à convertir un objet métier en message, il retourne un message FAIL.

Traçage

Les messages de trace sont codés en dur dans l'adaptateur. Le traçage est une fonction de débogage facultative que vous pouvez activer pour suivre de près le comportement d'un connecteur. Les messages de trace, par défaut, sont écrits dans STDOUT. Voir les propriétés de configuration du connecteur pour plus d'informations sur la configuration des messages de trace. Pour plus d'informations sur le traçage, pour savoir comment l'activer et le définir, voir *Connector Development Guide for Java*.

Voici le contenu recommandé pour les messages de trace du connecteur.

Niveau 0
: Ce niveau est utilisé pour les messages de trace qui identifient la version du connecteur.

Niveau 1
: Utilisez ce niveau pour les messages de trace qui fournissent des informations clés sur chaque objet métier traité, ou consignent à chaque fois qu'une unité d'exécution d'interrogation détecte un nouveau message dans une file d'attente d'entrée.

Niveau 2
: Utilisez ce niveau pour les messages de trace consignés à chaque fois qu'un objet métier est posté dans un courtier, à partir de `gotApplEvent()` ou `executeCollaboration()`.

Niveau 3
: Utilisez ce niveau pour les messages de trace qui fournissent des informations sur les conversions message-objet métier et objet métier-message, ou sur la livraison du message à la file d'attente de sortie.

Niveau 4
: Utilisez ce niveau pour les messages de trace qui déterminent à quel moment le connecteur entre dans une fonction ou la quitte.

Niveau 5
: Utilisez ce niveau pour les messages de trace qui indiquent l'initialisation du connecteur, représentent des instructions exécutées dans l'application, indiquent quand un message est extrait ou placé dans une file d'attente, ou enregistrent les affichages des objets métier.

  Utilisez ce niveau pour afficher la trace printStackTrace() sur des exceptions levées par l'adaptateur.

Résolution des incidents de démarrage

Incident	Solution potentielle / explication
Le connecteur s'arrête de façon imprévue pendant l'initialisation, avec le message suivant `Exception in thread "main" java.lang.NoClassDefFoundError: javax/jms/JMSException...`	Le connecteur ne trouve pas le fichier `jms.jar`.
Le connecteur s'arrête de façon imprévue pendant l'initialisation, avec le message suivant `Exception in thread "main" java.lang.NoClassDefFoundError: javax/naming/Referenceable...`	Le connecteur ne trouve pas le fichier `jndi.jar`.

Annexe A. Propriétés de configuration standard pour les connecteurs

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 18, à la page 55.)

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 18, à la page 55.

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 18 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 18, 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 18. 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 18. 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.Transformation 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++.
CommonEventInfrastructure	true ou false	false	Redémarrage du composant	
CommonEventInfrastructureURL	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 18. 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 18. 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 18. 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 18. 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 18. 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 64.

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. 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», à la page 79
- «Démarrage de Connector Configurator», à la page 80
- «Création d'un modèle de propriétés spécifiques au connecteur», à la page 81
- «Création d'un fichier de configuration», à la page 84
- «Définition des propriétés d'un fichier de configuration», à la page 87
- «Utilisation de Connector Configurator dans un environnement globalisé», à la page 97

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 80).

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 82 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 87).

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 82.
- 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 90.
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 89.

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 54.

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 plate-forme 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. Didacticiel

- «Présentation du didacticiel»
- «Configuration de votre environnement», à la page 100
- «Exécution des scénarios», à la page 102
- «Exécution du scénario utilisant un méta-objet statique», à la page 102
- «Exécution du scénario basé sur un méta-objet dynamique», à la page 103

La présente annexe indique comment utiliser l'adaptateur pour envoyer et recevoir des objets métier avec une application qui communiquer par des files d'attente de messages JMS. Les scénarios du didacticiel sont conçus pour montrer les points essentiels de la fonctionnalité de l'adaptateur.

Voir la préface de ce document pour obtenir plus d'informations sur les conventions de notation.

Présentation du didacticiel

Le didacticiel utilise deux scénarios, l'un utilise un méta-objet statique et l'autre un méta-objet dynamique. Tous deux font appel à ApplicationX, qui peut échanger des informations de contact d'entreprise lors de leur création, mise à jour et suppression. L'objet métier `Sample_JMS_Contact`, créé par vos soins, correspond aux zones définies dans les messages émis par ApplicationX. ApplicationX envoie et reçoit des messages dans un format compatible avec le gestionnaire de données Delimited, disponible dans les kits de développement IBM WebSphere Business Integration.

Le didacticiel utilise également le référentiel du Port Connector, inclus avec l'installation de WebSphere Adapters. Le Port Connector consiste en une définition de connecteur sans code sous-jacent, et en cette qualité, convient bien à des scénarios de simulation.

Une fois démarré, l'adaptateur JMS extrait les messages de contact postés par ApplicationX dans sa file d'attente d'entrée. A l'aide du gestionnaire de données Delimited, l'adaptateur convertit ces messages en objets métier `Sample_JMS_Contact` et les envoie au courtier d'intégration. Le connecteur Test (également inclus dans l'installation WBI), vous permet de simuler le Port Connector, d'extraire l'objet métier posté par l'adaptateur JMS et d'examiner les attributs. Une fois que vous avez modifié les données, vous livrez de nouveau le message au courtier d'intégration où il est envoyé à l'adaptateur JMS, converti en message et livré à la file d'attente de sortie de l'adaptateur (file d'attente d'entrée de ApplicationX). Dans le didacticiel, l'adaptateur est configuré pour WebSphere MQ Integrator Broker, mais vous n'avez pas besoin d'installer ce courtier pour exécuter le didacticiel.

Avant de poursuivre l'utilisation de ce didacticiel, vérifiez que :

- Vous avez installé et savez comment utiliser le produit IBM WebSphere.
- Vous avez installé un fournisseur de service JMS.
- Vous avez installé l'adaptateur WBI pour JMS.
- Pour avez installé et savez utiliser WebSphere MQ 5.3.

Configuration de votre environnement

Cette section explique comment préparer votre environnement à l'utilisation de ce didacticiel. Dans les instructions qui suivent, *sample_folder* désigne le dossier qui contient les exemples.

1. **Créez les files d'attente** Le didacticiel exige que vous définissiez six files d'attente avec votre fournisseur de service JMS. Auparavant, consultez la documentation du fournisseur JMS. Définissez les files d'attente suivantes (ou rendez-les disponibles par le biais de la recherche JNDI) :
   - `CWLD_Input`
   - `CWLD_InProgress`
   - `CWLD_Error`
   - `CWLD_Archive`
   - `CWLD_Unsubscribed`
   - `CWLD_Output`
2. **Créez et démarrez un gestionnaire de files d'attente WebSphere MQ** avec un initiateur de canal et un programme d'écoute en cours d'exécution.
3. **Définissez les files d'attente** exigées par l'adaptateur WebSphere MQ et le Port Connector pour la configuration du courtier WMQI :
   - `DEFINE QL('JMSConnector/ADMININQUEUE')`
   - `DEFINE QL('JMSConnector/ADMINOUTQUEUE')`
   - `DEFINE QL('JMSConnector/DELIVERYQUEUE')`
   - `DEFINE QL('JMSConnector/FAULTQUEUE')`
   - `DEFINE QL('JMSConnector/REQUESTQUEUE')`
   - `DEFINE QL('JMSConnector/RESPONSEQUEUE')`
   - `DEFINE QL('JMSConnector/SYNCHRONOUSREQUESTQUEUE')`
   - `DEFINE QL('JMSConnector/SYNCHRONOUSRESPONSEQUEUE')`
   - `DEFINE QL('PortConnector/ADMININQUEUE')`
   - `DEFINE QL('PortConnector/ADMINOUTQUEUE')`
   - `DEFINE QL('PortConnector/DELIVERYQUEUE')`
   - `DEFINE QL('PortConnector/FAULTQUEUE')`
   - `DEFINE QL('PortConnector/REQUESTQUEUE')`
   - `DEFINE QL('PortConnector/RESPONSEQUEUE')`
   - `DEFINE QL('PortConnector/SYNCHRONOUSREQUESTQUEUE')`
   - `DEFINE QL('PortConnector/SYNCHRONOUSRESPONSEQUEUE')`
4. **Configurez l'adaptateur** A l'aide de Connector Configurator, ouvrez *sample_folder*\`JMSConnector.cfg`. Pour plus d'informations sur l'utilisation de Connector Configurator, voir l'Annexe B, «Connector Configurator», à la page 79. Pour plus d'informations sur les propriétés spécifiques au connecteur, voir «Configuration des propriétés spécifiques au connecteur», à la page 20. Définissez les propriétés standard suivantes :
   - `Broker Type` Définissez cette propriété sur `WMQI`.
   - `Repository Directory` Définissez cette propriété sur le répertoire *sample_folder*.

   Définissez les propriétés spécifiques au connecteur suivantes :
   - `DuplicateEventElimination` Définissez cette propriété sur `true`.
   - `MonitorQueue` Définissez cette propriété sur `JMSConnector/MONITORQUEUE`

- ConfigurationMetaObject Définissez cette propriété sur Sample_JMS_MO_Config.
- DataHandlerConfigMO Définissez cette propriété sur Sample_JMS_MO_DataHandler.
- DataHandlerMimeType Définissez cette propriété sur text/delimited.
- ErrorQueue Définissez cette propriété sur CWLD_Error.
- InputQueue Définissez cette propriété sur CWLD_Input.
- UnsubscribedQueue Définissez cette propriété sur CWLD_Unsubscribed.

5. **Configurez Port Connector** A l'aide de Connector Configurator, définissez les propriétés standard suivantes :
   - Broker Type Définissez cette propriété sur WMQI.
   - Repository Directory Définissez cette propriété sur le répertoire *sample_folder*.
   - RequestQueue Définissez cette propriété sur JMSConnector/DELIVERYQUEUE (la valeur de la propriété DeliveryQueue pour l'adaptateur JMS).
   - DeliveryQueue Définissez cette propriété sur JMSConnector/REQUESTQUEUE (la valeur de la propriété RequestQueue pour l'adaptateur JMS).
6. **Définissez la prise en charge des objets métier** Pour pouvoir utiliser les objets métier, les adaptateurs doivent d'abord les prendre en charge. Dans Connector Configurator, cliquez sur l'onglet **Supported Business Objects** pour l'adaptateur JMS, ajoutez les objets métier indiqués dans le tableau 19, et définissez **Message Set ID** sur une valeur unique pour chaque objet métier pris en charge.

Tableau 19. Exemples d'objets métier pris en charge pour l'adaptateur JMS

Nom de l'objet métier	Message Set ID
Sample_JMS_MO_Config	1
Sample_JMS_MO_DataHandler	2
Sample_JMS_Contact	3

A l'aide de Connector Configurator, ouvrez la définition du Port Connector PortConnector.cfg, dans *sample_folder*, et ajoutez l'objet métier pris en charge et le Message Set ID affichés dans le tableau 20.

Tableau 20. Exemples d'objets métier pris en charge pour le connecteur Port

Nom de l'objet métier	Message Set ID
Sample_JMS_Contact	1

7. **Créez ou mettez à jour les scripts de démarrage du connecteur**
   Windows :
   a. Ouvrez les propriétés du raccourci de l'adaptateur pour JMS.
   b. Comme dernier argument de la cible, ajoutez -c suivi du *<chemin complet et du nom de fichier de JMSConnector.cfg>* Par exemple :
      -c*Product_Dir*\connectors\JMS\samples\JMSConnector.cfg

   UNIX :
   a. Ouvrez le fichier : *Product_Dir*/bin/connector_manager_JMS.
   b. Définissez la valeur de la propriété AGENTCONFIG_FILE sur -c, suivie du *<chemin complet et du nom de fichier de JMSConnector.cfg>* Par exemple :
      AGENTCONFIG_FILE=-c*Product_Dir*/connectors/JMS/samples/JMSConnector.cfg

Exécution des scénarios

Avant d'exécuter les scénarios :

1. **Démarrez l'adaptateur pour JMS** s'il n'est pas déjà en cours d'exécution.
2. **Démarrez le connecteur Visual Test** s'il n'est pas déjà en cours d'exécution.

Exécution du scénario utilisant un méta-objet statique

Cette partie du didacticiel décrit un scénario basé sur un méta-objet statique. Pour plus d'informations sur les méta-objets statiques, voir «Configuration d'un méta-objet statique», à la page 36.

1. **Simulez le Port Connector** A l'aide du connecteur Visual Test, définissez un profil pour PortConnector :
   a. Sélectionnez **File->Create/Select Profile** dans le menu du connecteur Visual Test, puis **File-> New Profile** dans le menu Connector Profile.
   b. Sélectionnez le fichier du configuration du Port Connector `PortConnector.cfg` dans le répertoire `Samples`, puis configurez `Connector Name` et `Broker Type`, et cliquez sur **OK**.
   c. Sélectionnez le profil créé et cliquez sur **OK**.
   d. Dans le menu du connecteur Visual Test, sélectionnez **File->Connect** pour commencer la simulation.
2. **Traitement de la requête de test**
   a. A l'aide de Test Connector, créez une nouvelle instance de l'objet métier `Sample_JMS_Contact` en sélectionnant l'objet métier dans la boîte déroulante **BoType**, puis **Create** pour l'instance BOInstance.
   b. Si vous le souhaitez, modifiez les valeurs par défaut, définissez l'instruction sur **Create** et envoyez le message en cliquant sur **Send BO**.
3. **Contrôlez la livraison du message** Ouvrez la file d'attente `CWLD_Output` pour voir si un nouveau message de contact au format `CON_CR` est arrivé, en provenance de l'adaptateur JMS.
4. **Testez le traitement de l'événement** Envoyez un message à la file d'attente d'entrée de l'adaptateur JMS. Remarque : cette étape exige un utilitaire capable d'envoyer des messages à une file d'attente. Dans le cas contraire, pour mettre en oeuvre une approche plus simple, vous pouvez définir la propriété `InputQueue` de l'adaptateur JMS sur `CWLD_Output` pour que l'adaptateur interroge ses propres messages. Dès qu'un message figure dans la file d'attente d'entrée, l'adaptateur JMS l'interroge et tente de le convertir en objet métier `Sample_JMS_Contact`. Pour que l'adaptateur interroge le message, il faut que son format soit égal à la valeur associée à l'objet métier `Sample_JMS_Contact` dans le méta-objet `Sample_JMS_MO_Config`. Dans ce scénario, ce format est `CON_CR`. Si l'adaptateur identifie le format du message entrant comme étant `CON_CR` , il utilise le gestionnaire de données pour convertir le message en objet métier `Sample_JMS_Contact` avec l'instruction create. L'objet métier nouvellement créé est ensuite livré au Test Connector.
5. **Confirmez la livraison du message** Si vous avez réussi à exécuter toutes les étapes ci-dessus, vous devez à présent disposer d'un scénario permettant à l'adaptateur JMS d'extraire des messages, de les convertir en objets métier `Sample_JMS_Contact` et de convertir ceux-ci en messages de contact.

Exécution du scénario basé sur un méta-objet dynamique

Ce scénario montre comment utiliser un méta-objet dynamique pour ré-acheminer un objet métier vers différentes files d'attente définies dans votre fournisseur de services JMS. Pour plus d'informations sur les méta-objet dynamiques, voir «Configuration d'un méta-objet enfant dynamique», à la page 38. Les étapes ci-dessous expliquent comment créer un attribut de méta-objet enfant pour `Sample_JMS_Contact`. Vous modifierez les valeurs des files d'attente de sortie de ce méta-objet enfant pour rediriger l'objet métier `Sample_JMS_Contact` sur diverses files d'attente.

Le référentiel de méta-objet enfant, `Sample_JMS_DynMO.xsd`, figure dans *sample_folder*.

1. **Identifiez l'attribut de méta-objet dynamique** D'abord vous devez ajouter des informations spécifiques à l'application pour identifier l'attribut contenant l'objet métier dynamique : dans `Sample_JMS_Contact`, ajoutez `cw_mo_conn=DynMO` aux informations spécifiques à l'application. Ceci identifie l'attribut.
2. **Ajoutez l'attribut** A l'aide de Business Object Designer :
   a. Ouvrez `Sample_JMS_DynMO.xsd` et `Sample_JMS_Contact.xsd` dans le répertoire *sample_folder*.
   b. Dans la fenêtre `Sample_JMS_Contact Object`, ajoutez un attribut nommé `DynMO` ou tapez `Sample_JMS_DynMO`.
   c. Cliquez deux fois sur `Sample_JMS_Contact Object`.
   d. Sélectionnez le dossier des attributs et ajoutez un attribut nommé `DynMO` de type `Sample_JMS_DynMO`.
3. **Définissez une nouvelle file d'attente cible** Définissez une file d'attente temporaire `REROUTE.IN` avec le fournisseur de service JMS. Il s'agit de l'endroit où le méta-objet réacheminera l'objet métier `Sample_JMS_Contact`.
4. **Démarrez l'adaptateur pour JMS** s'il n'est pas déjà en cours d'exécution.
5. **Démarrez le connecteur Visual Test** s'il n'est pas déjà en cours d'exécution.
6. **Simulez le Port Connecor** A l'aide du connecteur Visual Test, définissez un profil pour PortConnector :
   a. Sélectionnez **File->Create/Select Profile** dans le menu du connecteur Visual Test, puis **File-> New Profile** dans le menu Connector Profile.
   b. Sélectionnez le fichier de configuration du connecteur Port `PortConnector.cfg` dans le répertoire `Samples`, puis configurez Connector Name et Broker Type et cliquez sur **OK**.
   c. Sélectionnez le profil créé et cliquez sur **OK**.
   d. Dans le menu du connecteur Visual Test, sélectionnez **File->Connect** pour commencer la simulation.
7. **Créez des instances d'objet métier parent et de méta-objet enfant** A l'aide de Visual Test Connector :
   a. Créez une nouvelle instance de l'objet métier `Sample_JMS_Contact`, en remplaçant si nécessaire les valeurs par défaut.
   b. Cliquez à l'aide du bouton droit sur l'attribut `DynMO` et créez-en une instance, `Sample_JMS_DynMO`.
8. **Définissez la nouvelle file d'attente cible**
   a. Développez l'attribut `DynMO` en cliquant sur le signe + qui figure en regard.
   b. Dans l'attribut `outputQueue` , entrez le nom de la file d'attente cible : `REROUTE.IN`

9. **Envoyez l'objet métier** Cliquez sur **Send BO**.
10. **Confirmez la livraison du message** Ouvrez la file d'attente `REROUTE.IN` pour voir si un nouveau message de contact est arrivé depuis l'adaptateur JMS. Si un nouveau message est arrivé de l'adaptateur JMS dans la file d'attente nommée `REROUTE.IN`, alors le réacheminement a fonctionné.

Annexe D. Configuration des messageries basées sur des rubriques et sur des files d'attente

- «Configuration pour la messagerie basée sur des files d'attente»
- «Configuration de la messagerie basée sur des rubriques», à la page 106

La présente annexe montre comment configurer l'adaptateur pour JMS avec WebSphere MQ comme fournisseur JMS. Pour plus d'informations, voir le guide *WebSphere MQ Using Java*.

Remarque : Si vous utilisez WebSphere MQ comme fournisseur JMS, nous vous recommandons vivement d'utiliser WebSphere Business Integration Adapter for WebSphere MQ pour l'intégration. Les étapes ci-dessous sont présentées à titre indicatif, uniquement pour expliquer comment configurer l'adaptateur JMS à l'aide d'un fournisseur JMS commun.

Voir la préface de ce document pour obtenir plus d'informations sur les conventions de notation.

Configuration pour la messagerie basée sur des files d'attente

1. Installez les bibliothèques client WebSphere MQ et WebSphere MQ (y compris la prise en charge JMS).
2. Assurez-vous que toutes les bibliothèques client MQ, y compris `fscontext.jar` et `providerutil.jar`, figurent dans le chemin de classe du système. Vous avez également la possibilité de modifier le fichier `jmsAdmin.bat` et d'ajouter `-Djava.ext.dirs="<votre répertoire principal MQ>/Java/lib` au script de ligne de commande java, pour vérifier que tous les fichiers de la bibliothèque client sont à la disposition de l'outil. Notez que les éventuelles exceptions ClassDefNotFoundExceptions reportées par l'outil sont dues à des bibliothèques manquantes—vérifiez de nouveau vos chemins de classe.
3. Ouvrez *<votre répertoire principal MQ>*`Java/bin/jmsAdmin.config` et définissez les propriétés suivantes :
   - `INITIAL_CONTEXT_FACTORY=com.sun.jndi.fscontext.RefFSContextFactory`
   - `PROVIDER_URL=file:/c:/temp`
   - `SECURITY_AUTHENTICATION=none`
4. Créez un fichier nommé `MyJNDI.txt` contenant les informations suivantes
   ```
   DEFINE QCF(MyQCF) HOST(<your host name>)  +PORT(<nom de votre port de
   programme d'écoute MQ, par exemple, 1414>)  +
   CHANNEL(<nom de votre canal de connexion de serveur, par exemple,
   CHANNEL1>) +
   QMGR(<nom de votre gestionnaire de file d'attente MQ>) +
   TRAN(client)
   END
   ```
5. Liez des objets à des noms JNDI en exécutant *<votre répertoire MQ principal>*`/java/bin/jmsAdmin.bat < MyJNDI.txt`

6. Configurez les propriétés suivantes spécifiques au connecteur JMS, comme indiqué ci-dessous :

```
CTX_InitialContextFactory = com.sun.jndi.fscontext.RefFSContextFactory
CTX_ProviderURL = file://c:/temp
ConnectionFactoryName = MyQCF
```

Configuration de la messagerie basée sur des rubriques

1. Installez les bibliothèques client WebSphere MQ et WebSphere MQ (y compris la prise en charge JMS).
2. Assurez-vous que toutes les bibliothèques client MQ, y compris `fscontext.jar` et `providerutil.jar`, figurent dans le chemin de classe du système. Vous avez également la possibilité de modifier le fichier `jmsAdmin.bat` et d'ajouter `-Djava.ext.dirs="<votre répertoire principal MQ>/Java/lib` au script de ligne de commande java, pour vérifier que tous les fichiers de la bibliothèque client sont à la disposition de l'outil. Notez que les éventuelles exceptions ClassDefNotFoundExceptions rapportées par l'outil sont dues à des bibliothèques manquantes—vérifiez de nouveau vos chemins de classe.
3. Téléchargez et installez le WebSphere MQ MA0C SupportPac approprié, proposé par IBM pour activer la prise en charge de la messagerie basée sur des rubriques (publication/souscription) dans MQ. Par exemple, une recherche basée sur `ma0c_ntmq52` localisera la messagerie basée sur des rubriques pour MQ 5.2 sous Windows.
4. Allez dans le répertoire *<votre répertoire principal MQ>*`/Java/bin` et exécutez `runmqsc < MQJMS_PSQ.mqsc`.
5. Exécutez `IVTSetup.bat`. Le processus doit afficher `Done!` sans indiquer d'erreur.
6. Ouvrez *<votre répertoire principal MQ>*`Java/bin/jmsAdmin.config` et définissez les propriétés suivantes :
   - `INITIAL_CONTEXT_FACTORY=com.sun.jndi.fscontext.RefFSContextFactory`
   - `PROVIDER_URL=file://c:/temp`
   - `SECURITY_AUTHENTICATION=none`
7. Créez un fichier nommé `MyJNDI.txt` contenant les informations suivantes :

```
DEFINE QCF(MyQCF) HOST(<your host name>)  + PORT(<nom de votre port de
programme d'écoute MQ, par exemple, 1414>)  +
CHANNEL(<nom de votre canal de connexion de serveur, par exemple,
CHANNEL1>) +
QMGR(<nom de votre gestionnaire de file d'attente MQ>) +
TRAN(client)
END
```

8. Liez des objets à des noms JNDI en exécutant *<votre répertoire MQ principal>*`/java/bin/jmsAdmin.bat < MyJNDI.txt`
9. Configurez les propriétés suivantes spécifiques au connecteur JMS, comme indiqué ci-dessous :

```
CTX_InitialContextFactory = com.sun.jndi.fscontext.RefFSContextFactory
CTX_ProviderURL = file://c:/temp
ConnectionFactoryName = MyQCF
```

Annexe E. 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>
```

Annexe F. 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 événements 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*.

Index

A

adaptateur
  architecture 4
  arrêt 46
  démarrage 44
  styles de messagerie 2
API JMS 4
appels Application Response Measurement, prise en charge 113
APPRESPONSETIMEOUT 10
architecture du connecteur 1
archivage 10
  ArchiveDestination 22
  ErrorDestination 22
  InProgressDestination 22
  UnsubscribedDestination 22
ArchivalConnectionFactoryName 22
ArchiveDestination 22, 23
arrêt du connecteur 46
ASI 4
attributs d'en-tête du méta-objet dynamique 39

C

catalogue d'événements, pour Common Event Infrastructure 108
codes de retour asynchrones 13
Common Event Infrastructure
  catalogue d'événements 108
  métadonnées 108
compatibilité du courtier 2
configuration
  exemple d'environnement 100
  JNDI 30
  le nom de l'application 19
  méta-objet statique 36
  propriétés spécifiques au connecteur 20
  propriétés standard du connecteur 20
  scripts de démarrage 43
  style de message 30
configuration d'un méta-objet enfant dynamique 38
configuration de JNDI avec les bibliothèques du client WebSphere MQ Java 31
configuration de votre environnement 100
configuration des méta-objets 31
ConfigurationMetaObject 23
connecteur
  configuration 19
  création de plusieurs instances 43
  distinct de l'adaptateur 1
ConnectionFactoryName 23
contexte JNDI 25
création de plusieurs instances de connecteur 43
CTX_InitialContextFactory 24
CTX_ProviderURL 25
cycle d'interrogation 5

D

DataHandlerClassName 26
DataHandlerConfigMO 26
DataHandlerMimeType 26
DefaultVerb 27
démarrage 44
dépendances de l'adaptateur 3
destination JMS 4
détection d'événement 5
données dépendant des paramètres régionaux
  prise en charge des caractères sur deux octets 3
données localisées 3

E

élimination des événements en double 7
en-têtes JMS
  et attributs de méta-objet dynamique 41
EnableMessageProducerCache 27
environnement de l'adaptateur 1
ErrorDestination 22, 27
événement
  statut et reprise sur incident 6
événements gérés par le conteneur 7
extraction des événements 8

F

Fail on startup 7
flot de messages
  asynchrone 11
  Généralités 4
  synchrone 11
fournisseur JMS 4

G

gestion des erreurs 16
gestionnaire de données
  DataHandlerClassName 26
  DataHandlerConfigMO 26
  DataHandlerMimeType 26
  mappage sur les destinations d'entrée 37
  règles de configuration 12

I

IBM Tivoli Monitoring for Transaction Performance 113
Ignore 7
InDoubtEvents 27
InProgressDestination 22, 27
InputDestination 27
installation de l'adaptateur et des fichiers associés 17

J

Java Virtual Machine 4
jeu de codes de caractères Unicode 4
JMS 1
  standard 1.0.2 2
JNDI
  configuration 30

L

Log error 7
LookupDestinationsUsingJNDI 28

M

magasin JNDI 28
mappage de l'en-tête de message 10
mappage des gestionnaires de données sur les destinations d'entrée 37
Message Oriented Middleware 4
MessageFormatProperty 28
messagerie basée sur des rubriques 2
messagerie basée sur file d'attente 2
messagerie de publication/souscription 2
messagerie point à point 2
messagerie Pub/Sub 2
Messages 4
méta-objet dynamique 9
  quand l'utiliser 31
méta-objet statique 9
  et mappage d'objet métier 9
  quand l'utiliser 31
méta-objets 8
  alimentation des méta-objets dynamiques pendant l'interrogation 41
  attributs d'en-tête du méta-objet dynamique 39
  attributs dynamiques et en-têtes JMS 41
  configuration 31
  configuration, d'un méta-objet dynamique 38
  configuration d'un méta-objet statique 36
  ConfigurationMetaObject 23
  différence entre dynamique et statique 9
  dynamique (quand l'utiliser) 31
  propriétés dynamiques, statiques 31
  propriétés en lecture et en écriture 40
  statique (quand l'utiliser) 31

métadonnées 8
méthode doVerbFor() 11
méthode pollForEvents() 5

O

objet métier
  mappage 9
objets métier
  création 47
  structure 47

P

plateformes de l'adaptateur 2
PollQuantity 28
prise en charge de l'instruction 12
propriétés de méta-objet
  synchrone 13
propriétés de méta-objet synchrone 13
propriétés spécifiques au connecteur 20
propriétés standard du connecteur 20
PTP 4, 30
Pub/Sub 4, 30

R

Recovery with guaranteed event
  delivery 7
ReplyToDestination 28
reprise après erreur 10
Reprocess 7
response_selector 14

S

scripts de démarrage
  configuration 43
SessionPoolSizeForRequests 28
standards de l'adaptateur 2
structure de fichier du connecteur sous
  UNIX 18
structure de fichiers du connecteur sous
  Windows 17
structure de fichiers installée 17
style de message
  configuration 30
styles de messagerie
  Pub/Sub ou basés sur des
    rubriques 2
styles de messagerie PTP (point à point)
  basés sur le point à point et les files
    d'attente 2
surveillance, des transactions 113
surveillance des transactions 113

T

Tivoli Monitoring for Transaction
  Performance 113
traitement asynchrone 11
traitement de la requête de message
  gestion des erreurs 16
  traitement des réponses 16
traitement de message de requête
  asynchrone 12
traitement de requête asynchrone
  Généralités 12
  renseignement de l'en-tête de message
    JMS 12
traitement des événements
  Généralités 5
traitement des messages
  événements 5
traitement des requêtes
  Généralités 11
traitement synchrone 11

U

UnsubscribedDestination 22, 29
UnsubscribeOnTerminate 29
UseDefaults 29

W

WebSphere MQ
  extraction 5
  traitement des requêtes 11

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 Adapters, Version 6.0

WebSphere Business Integration Adapter Framework V2.6.0