Composants IBM WebSphere Business Integration
Adapter

Adapter for Manugistics - Guide de l'utilisateur

V 1.1.x

Composants IBM WebSphere Business Integration
Adapter

Adapter for Manugistics - Guide de l'utilisateur

V 1.1.x

Remarque

Avant d'utiliser le présent document et le produit associé, prenez connaissance des informations générales figurant dans la section «Informations légales» à la page 133.

Remarque

Les captures d'écran de ce manuel ne sont pas disponibles en français à la date d'impression.

Juillet 2004

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

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

Vous pouvez également consulter les serveurs Internet suivants :

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

Compagnie IBM France
Direction Qualité
Tour Descartes
92066 Paris-La Défense Cedex 50

Table des matières

Avis aux lecteurs canadiens

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

Illustrations

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

Terminologie

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

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

Claviers

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

OS/2 et Windows - Paramètres canadiens

Au Canada, on utilise :

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

Nomenclature

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

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

Brevets

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

Assistance téléphonique

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

A propos de ce document

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

Le présent document décrit l'installation, la configuration et le développement des objets métier pour le composant Adapter for Manugistics.

Public visé

Ce document s'adresse aux consultants, développeurs et aux administrateurs système qui utilisent le connecteur sur les sites client.

Documents connexes

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

Vous pouvez télécharger la documentation associée sur les sites suivants :

- Pour obtenir des informations générales sur 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
- Pour plus d'informations sur les courtiers de messages (WebSphere MQ Integrator Broker, WebSphere MQ Integrator et WebSphere Business Integration Message Broker) :

  http://www.ibm.com/software/integration/mqfamily/library/manualsa/
- Pour plus d'informations sur WebSphere Application Server :

  http://www.ibm.com/software/webservers/appserv/library.html

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

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. Pour les consulter, accédez au site du support de WebSphere Business Integration, http://www.ibm.com/software/integration/websphere/support/.

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.
ProductDir	Correspond au répertoire où le produit est installé. Pour l'environnement IBM WebSphere InterChange Server, le répertoire produit par défaut est «IBM\WebSphereICS.» Pour l'environnement des composants IBM WebSphere Business Integration Adapter, le répertoire produit par défaut est «WebSphereAdapters».
`{ }`	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, un trait vertical sépare 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.
`< >`	Les signes inférieur à et supérieur à entourent les différents éléments d'un nom afin de pouvoir les différencier les uns des autres, par exemple, `<nom_serveur><`*`nom_connecteur`*`>tmp.log`.
`/, \`	Dans ce document, les barres obliques inverses (\) sont utilisées comme convention pour les chemins de répertoire. Pour les systèmes UNIX(R), remplacez les barres obliques inverses par des barres obliques (/). Tous les noms de chemin des produits sont associés au répertoire dans lequel le connecteur pour JDBC est installé sur votre système.
UNIX:/Windows:(TM)	Les paragraphes qui commencent par l'un de ces éléments indiquent des remarques sur les différences qui existent entre les systèmes d'exploitation.
`%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 1.1.x

Mise à jour en juin 2004. L'édition de ce document pour la version 1.1.x de l'adaptateur contient les nouveautés ou modifications suivantes :

- L'adaptateur prend en charge Manugistics NetWORKS Collaborate version 7.2. Suite à cette nouvelle fonctionnalité, l'adaptateur a été renommé Adapter for Manugistics.
- Les nouvelles propriétés de configuration spécifiques au connecteur sont QueryTimeOut, ReturnDummyBOForSP, SelectiveCommitForPoll et UseDefaultsWhenPolling (laquelle remplace UseDefaultsForRetrieve). Une propriété existante, DriverSupportForLong, fait l'objet d'une description.
- L'adaptateur prend en charge un nouveau paramètre pour les informations spécifiques à l'application des attributs de type DATE.
- En ce qui concerne l'instruction DeltaUpdate, le connecteur reconnaît les noms d'attributs de procédure stockée suivants : BeforeDeltaUpdateSP, AfterDeltaUpdateSP, DeltaUpdateSP.
- L'arborescence du schéma contient les synonymes/surnoms des noeuds supplémentaires pour l'identification des objets de base de données à associer à la définition d'objet métier générée.
- L'arborescence du schéma contient le nom de schéma ALL SCHEMAS permettant d'extraire les objets des bases de données n'ayant aucun schéma associé aux objets.
- La version 1.1.x du composant Adapter for Manugistics n'étant pas prise en charge sur Solaris 7.0, les références à cette plateforme ont été supprimées de ce guide.

Nouveautés de l'édition 1.0

Mise à jour en février 2004. L'édition de ce document contient les nouveautés ou modifications suivantes : le paramètre nom-valeur [PH=true|false] a été ajouté à la section «Informations spécifiques à l'application pour les attributs simples» du chapitre 3.

Décembre 2003

La version 1.0 constitue la première édition du guide *Adapter for Manugistics Demand and Fulfillment Management User Guide*.

Chapitre 1. Présentation du connecteur

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

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

Le présent chapitre décrit le composant du connecteur IBM WebSphere Business Integration Adapter for Manugistics. Notez que ce document contient des informations à la fois sur l'architecture du connecteur et sur le composant propre à l'application. Il fait référence à ces deux éléments comme étant le connecteur. Pour plus d'informations sur la relation qui existe entre le courtier d'intégration et le connecteur, voir le document *IBM WebSphere InterChange Server System Administration Guide* ou le document *IBM WebSphere Business Integration Adapters Implementation Guide for WebSphere MQ Integrator Broker*.

Ce chapitre contient les sections suivantes :

Composants du connecteur

Le connecteur pour Manugistics permet au courtier d'intégration d'échanger des objets métier avec une application définie sur une base de données prise en charge par un pilote conforme aux spécifications JDBC 2.0 ou supérieur. Cette section explique en détail l'architecture du connecteur et l'utilisation de différents pilotes JDBC.

Pour spécifier le pilote que le connecteur doit utiliser pour se connecter à la base de données, voir «Activation de la prise en charge de plusieurs pilotes» à la page 16.

Le connecteur se connecte à la base de données à l'aide du mécanisme JDBC Connect. Un paramètre de configuration spécifique au connecteur (`DatabaseURL`) vous permet d'indiquer le nom du serveur de base de données auquel le connecteur doit se connecter. Pour plus d'informations sur les paramètres de configuration, voir «Configuration du connecteur» à la page 18.

Lorsque le connecteur démarre, il crée un pool de connexion avec la base de données. Il utilise les connexions de ce pool pour le traitement de toutes les transactions avec la base de données. A l'arrêt du connecteur, toutes les connexions dans le pool sont fermées.

Architecture du connecteur

La figure 1 illustre les composants du connecteur et les relations qu'ils entretiennent avec le système d'intégration métier.

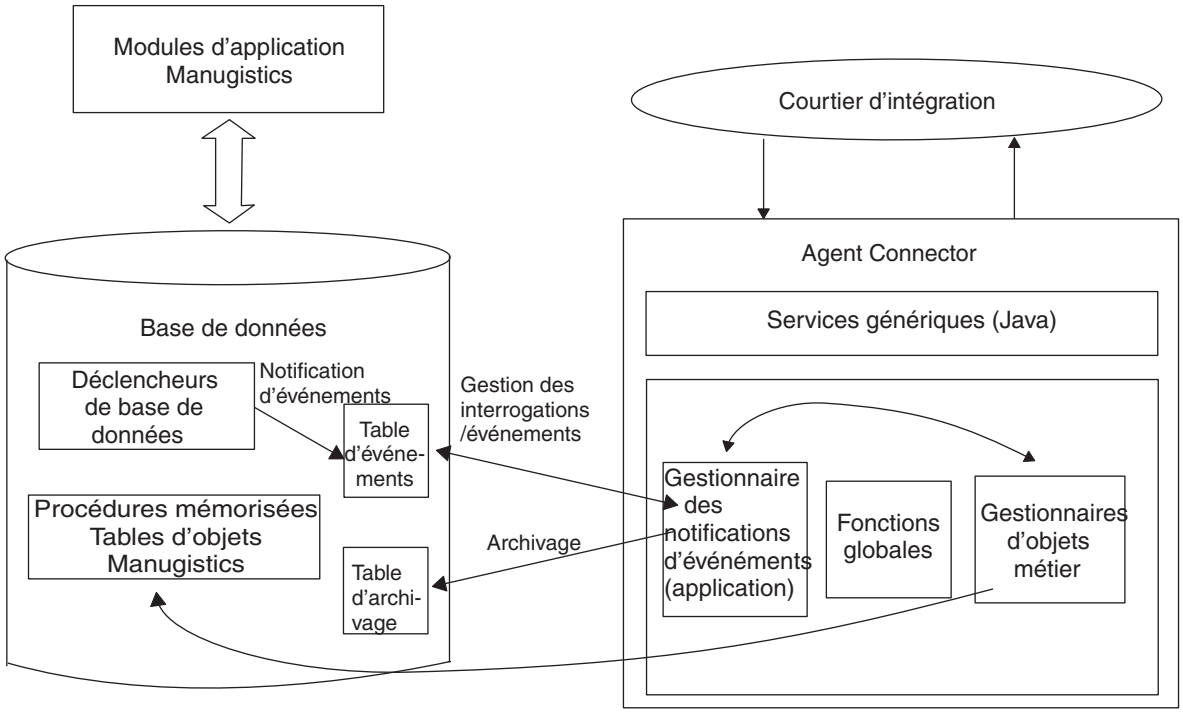


Figure 1. Architecture de requête de l'objet métier

Fonctionnement du connecteur

Cette section explique comment les métadonnées améliorent la souplesse du connecteur et décrit en détail le traitement des objets métier et la notification des événements.

Le connecteur et les métadonnées

Le connecteur est contrôlé par les métadonnées. Le terme **métadonnées**, dans l'environnement d'IBM WebSphere Business Integration Adapter, désigne les données spécifiques à une application qui sont stockées dans des objets métier et qui assistent le connecteur dans ses échanges avec l'application. Un connecteur contrôlé par des métadonnées traite chaque objet métier qu'il prend en charge d'après les métadonnées codées dans la définition de l'objet métier plutôt que d'après les instructions codées en dur dans le connecteur.

Les métadonnées de l'objet métier incluent la structure d'un objet métier, les paramètres de ses propriétés d'attribut et le contenu de ses informations spécifiques à l'application. Dans la mesure où le connecteur est contrôlé par les métadonnées, il peut traiter de nouveaux objets métier ou des objets modifiés sans devoir modifier les codes du connecteur.

Le connecteur exécute les instructions SQL ou les procédures stockées en vue d'extraire ou de modifier les données de la base de données ou de l'application. Pour créer des instructions SQL ou des procédures stockées, le connecteur utilise des métadonnées propres à l'application. Les instructions SQL et les procédures stockées effectuent la récupération requise depuis la base de données ou

l'application ou apportent les modifications nécessaires pour l'objet métier et l'instruction que le connecteur traite. Pour plus d'informations sur les informations spécifiques à l'application, voir Chapitre 3, «Présentation des objets métier pour le connecteur», à la page 35.

Traitement de l'objet métier

Cette section explique comment le connecteur traite les requêtes des objets métier et les événements d'application. Pour plus d'informations, voir «Traitement des instructions d'objet métier» à la page 40.

Traitement des requêtes des objets métier

Lorsque le connecteur reçoit une requête d'exécution d'une opération d'application, il traite les objets métier hiérarchiques de manière récursive : il exécute la même procédure pour chaque objet métier enfant jusqu'à ce qu'il ait traité l'ensemble des objets métier. L'ordre dans lequel le connecteur traite les objets métier enfant et l'objet métier de niveau supérieur dépend si les objets métier enfant sont inclus avec ou sans droits de propriété et s'ils sont de type cardinalité simple ou multiple.

Remarque : Le terme d'objet métier **hiérarchique** désigne un objet métier complet, incluant tous les objets métier enfant qu'il contient à n'importe quel niveau. Le terme d'objet métier **individuel** désigne un objet métier unique, indépendant des objets métier qu'il peut contenir ou qui le contient. Le terme d'objet métier **de niveau supérieur** désigne l'objet métier individuel en haut de la hiérarchie qui ne possède pas lui-même d'objet métier parent.

Extraction de l'objet métier : Lorsqu'un courtier d'intégration demande au connecteur d'extraire un objet métier hiérarchique de la base de données, le connecteur tente de renvoyer un objet métier qui correspond exactement à la représentation de cet objet métier dans la base de données actuelle. Autrement dit, tous les attributs simples de chaque objet métier renvoyé au courtier d'intégration correspondent à la valeur de la zone correspondante dans la base de données. En outre, le nombre d'objets métier dans chaque tableau contenu par l'objet métier renvoyé correspond au nombre d'enfants dans la base de données pour ce tableau.

Pour effectuer cette extraction, le connecteur utilise les valeurs de clé primaire dans l'objet métier de niveau supérieur pour descendre de manière récursive dans les données correspondantes de la base de données.

Extraction de l'objet métier par le contenu : Lorsqu'un courtier d'intégration demande au connecteur d'extraire un objet métier hiérarchique en fonction des valeurs définies dans les attributs non clés dans l'objet métier de niveau supérieur, le connecteur utilise la valeur des attributs non nuls comme critères pour extraire les données.

Création de l'objet métier : Lorsqu'un courtier d'intégration demande au connecteur de créer un objet métier hiérarchique dans la base de données, le connecteur effectue les opérations suivantes :

1. Il crée de manière récursive chaque objet métier enfant de type cardinalité simple contenu avec des droits de propriété dans la base de données.
2. Il traite chaque objet métier enfant de type cardinalité simple dépourvu de droits de propriété.
3. Il crée l'objet métier de niveau supérieur dans la base de données.
4. Il crée chaque objet métier enfant de type cardinalité simple qui stocke la relation parent-enfant dans l'objet enfant.

5. Il crée chaque objet métier enfant de type cardinalité multiple.

Modification de l'objet métier : Lorsqu'un courtier d'intégration demande au connecteur de mettre à jour un objet métier hiérarchique dans la base de données, le connecteur effectue les opérations suivantes :

1. Il utilise les valeurs de clé primaire de l'objet métier source pour extraire l'entité correspondante de la base de données.
2. Il met à jour de manière récursive tous les enfants de type cardinalité simple de l'objet métier de niveau supérieur.
3. Pour les objets métier enfant de type cardinalité simple qui stockent la relation dans l'objet parent, il affecte à chaque valeur de clé étrangère dans l'objet parent la valeur de clé primaire dans l'objet métier enfant de type cardinalité simple.
4. Il met à jour tous les attributs simples de l'objet métier extrait, à l'exception de ceux dont l'attribut correspondant dans l'objet métier source contient la valeur CxIgnore.
5. Il affecte à toutes les valeurs de clé étrangère dans chaque objet enfant qui stocke la relation parent-enfant dans l'objet enfant (de type cardinalité multiple et simple) la valeur de clé primaire de son objet métier parent correspondant.
6. Il traite tous les tableaux de l'objet métier extrait.

Suppression de l'objet métier : Lorsqu'un courtier d'intégration demande au connecteur de supprimer un objet métier hiérarchique de la base de données, le connecteur effectue les opérations suivantes :

1. Il supprime les objets enfant de type cardinalité simple.
2. Il supprime les objets enfant de type cardinalité multiple.
3. Il supprime l'objet métier de niveau supérieur.

Traitement des événements d'application

Le connecteur traite les événements de création, de mise à jour et de suppression générés par l'application comme décrit ci-dessous.

Notification de création : Lorsque le connecteur rencontre un événement de création dans la table d'événements, il crée un objet métier du type indiqué par l'événement, définit les valeurs de clés de l'objet métier (à partir des clés indiquées dans la table d'événements), et extrait l'objet métier de la base de données. Une fois qu'il a extrait l'objet métier, le connecteur l'envoie au courtier d'intégration par le biais de l'instruction Create.

Notification de mise à jour : Lorsque le connecteur rencontre un événement de mise à jour dans la table d'événements, il crée un objet métier du type indiqué par l'événement, définit les valeurs de clés de l'objet métier (à l'aide des clés indiquées dans la table d'événements), et extrait l'objet métier de la base de données. Une fois qu'il a extrait l'objet métier, le connecteur l'envoie au courtier d'intégration par le biais de l'instruction Update.

Notification de suppression : Lorsque le connecteur rencontre un événement de suppression dans la table d'événements, il crée un objet métier du type indiqué par l'événement, définit les valeurs de clés de l'objet métier (à l'aide des clés indiquées dans la table d'événements), et l'envoie au courtier d'intégration par le biais de l'instruction Delete. La valeur CxIgnore est affectée à toutes les valeurs autres que les valeurs de clés. Si des zones non clés sont importantes sur le site, modifiez la valeur des zones, si nécessaire.

Le connecteur traite les opérations de suppression logique et physique qui sont générées par son application. Dans le cas de suppressions physiques, le mécanisme SmartFiltering supprime tous les événements non traités de l'objet métier (comme les événements de création ou de mise à jour) avant d'insérer l'événement de suppression dans la table d'événements. Dans le cas des suppressions logiques, le connecteur insère un événement de suppression dans la table d'événements sans supprimer les autres événements de l'objet métier.

Extraction d'objets métier pour le traitement des événements : Une extraction peut être réalisée de deux manières sur un objet métier pour le traitement des événements. La première méthode est une extraction basée sur les attributs clés dans un objet métier. La seconde méthode est une extraction basée sur les attributs clés et non clés. Dans ce cas, l'objet métier doit prendre en charge l'instruction `RetrieveByContent` et utiliser la paire `nom_valeur` pour les clés de l'objet.

Remarque : Si la clé de l'objet n'utilise pas la paire `nom_valeur`, les clés de la zone clé de l'objet doivent suivre le même ordre que celui des clés dans l'objet métier.

Notification d'événements

Le mécanisme de détection des événements du connecteur utilise une table d'événements, une table d'archivage, des procédures stockées et des déclencheurs de base de données. Etant donné la possibilité de points de défaillance associés au traitement des événements, le processus de gestion des événements ne supprime pas un événement de la table tant qu'il n'a pas été inséré dans la table d'archivage.

Les déclencheurs de la base de données remplissent une table d'événements lorsqu'un événement intéressant se produit dans la base de données. Le connecteur interroge cette table à intervalles réguliers, que vous pouvez définir, extrait les événements et les traite tout d'abord dans l'ordre de priorité, puis de manière séquentielle. Lorsque le connecteur a traité un événement, son état est mis à jour.

Remarque : Vous devez ajouter les déclencheurs à la base de données dans le cadre de la procédure d'installation.

Le paramètre de sa propriété ArchiveProcessed détermine si le connecteur archive un événement dans la table d'archivage après avoir mis à jour son état. Pour plus d'informations sur la propriété ArchiveProcessed, voir «Configuration du connecteur» à la page 18.

Le tableau 1 illustre le mode opératoire de l'archivage en fonction de la définition de la propriété ArchiveProcessed.

Tableau 1. Mode opératoire de l'archivage

Définition de la propriété ArchiveProcessed	Raison de la suppression de la table d'événements	Mode opératoire du connecteur
`true` ou aucune valeur	Traité correctement	Archivé avec l'état `Sent to InterChange`
	Echec du traitement	Archivé avec l'état `Error`
	Aucune inscription pour l'objet métier	Archivé avec l'état `Unsubscribed`
`false`	Traité correctement	Non archivé et supprimé de la table d'événements

Tableau 1. Mode opératoire de l'archivage (suite)

Définition de la propriété ArchiveProcessed	Raison de la suppression de la table d'événements	Mode opératoire du connecteur
	Echec du traitement	Reste dans la table d'événements avec l'état `Error`
	Aucune inscription pour l'objet métier	Reste dans la table d'événements avec l'état `Unsubscribed`

SmartFiltering est un mécanisme dans les déclencheurs de la base de données qui réduit le nombre de traitements que doivent entreprendre le courtier d'intégration et le connecteur. Par exemple, si une application a mis à jour l'objet métier Contract 15 fois depuis la dernière fois que le connecteur a demandé si des événements s'étaient produits, SmartFiltering stocke ces changements dans un seul événement Update.

Gestion des connexions interrompues à une base de données

L'interruption d'une connexion à une base de données peut avoir plusieurs explications. Si la connexion est interrompue, le connecteur s'arrête. La spécification JDBC ne fournit pas de mécanisme pour la détection des connexions interrompues. Etant donné que le connecteur prend en charge des bases de données différentes, il n'existe pas de définition unique des codes d'erreur pour une connexion interrompue.

La propriété `PingQuery` est fournie pour assurer cette fonction. Si une défaillance se produit lors d'une requête d'appel de service, le connecteur exécute la commande `PingQuery` pour confirmer que la défaillance n'était pas due à une connexion interrompue à la base de données. Si la commande PingQuery échoue et que la valeur "false" est affectée à la propriété AutoCommit, le connecteur tentera de rétablir la connexion à la base de données. S'il parvient à rétablir la connexion à la base de données, il poursuivra le traitement. Dans le cas contraire, le connecteur renvoie un message APPRESPONSETIMEOUT, qui provoque l'arrêt du connecteur.

La commande `PingQuery` est exécutée en cas d'échec de l'accès à une base de données pour un type de transaction quelconque. Par exemple :

- lors de l'accès aux tables d'événements et d'archivage ;
- lors de l'extraction de l'objet métier associé à l'événement ;
- lors de la création ou de la mise à jour d'un enregistrement lié à un objet métier.

Traitement des données dépendantes de l'environnement local

Le connecteur a été internationalisé de sorte qu'il puisse prendre en charge les jeux de caractères à deux octets et transmettre le texte du message dans la langue indiquée. Lorsque le connecteur transfère des données depuis un emplacement qui utilise un jeu de codes de caractères spécifique vers un emplacement qui utilise un jeu de codes de caractères différent, il procède à la conversion des caractères afin de conserver le sens des informations.

L'environnement d'exécution Java(TM) dans la machine virtuelle Java (JVM) représente les données dans le jeu de codes de caractères Unicode. Le format Unicode contient des codes pour les caractères présents dans la plupart des jeux de codes de caractères connus (à la fois mono-octet et multi-octets). La plupart des composants du système WebSphere Business Integration sont rédigés en Java. Par

conséquent, lorsque des données sont transférées entre la plupart des composants du système WebSphere Business Integration, 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 de configuration standard de l'environnement local pour votre environnement. Pour plus d'informations sur ces propriétés, voir Annexe A, «Propriétés de configuration standard pour les connecteurs», à la page 97.

Remarque : Pour activer différents jeux de codes de caractères, reportez-vous à la documentation de Manugistics.

Chapitre 2. Installation et configuration du connecteur

Le présent chapitre explique comment installer et configurer IBM WebSphere Business Integration Adapter for Manugistics et comment configurer les applications pour qu'elles fonctionnent avec le connecteur. Il contient les sections suivantes :

Environnement de l'adaptateur

Avant d'installer, de configurer et d'utiliser l'adaptateur, vous devez connaître les spécifications inhérentes à son environnement. Elles font l'objet de la section suivante.

Compatibilité du courtier

L'architecture qu'utilise l'adaptateur doit être compatible avec la version du courtier d'intégration avec lequel l'adaptateur communique. L'architecture de l'adaptateur est mise en place lors de l'installation du courtier d'intégration ou de l'adaptateur, selon le courtier d'intégration utilisé. La version 1.1.x du composant Adapter for Manugistics est prise en charge sur l'architecture de l'adaptateur et les courtiers d'intégration suivants :

- **Architecture de l'adaptateur** :
  WebSphere Business Integration Adapter Framework versions 2.3.1 et 2.4.
- **Courtiers d'intégration** :
  - WebSphere InterChange Server, versions 4.1.1, 4.2, 4.2.1 et 4.2.2
  - WebSphere MQ Integrator, version 2.1.0
  - WebSphere MQ Integrator Broker, version 2.1.0
  - WebSphere Business Integration Message Broker, version 5.0
  - WebSphere Application Server Enterprise, version 5.0.2, avec WebSphere Studio Application Developer Integration Edition, version 5.0.1

Voir les *Notes d'édition* pour connaître les exceptions.

Remarque : Pour plus d'informations sur l'installation du courtier d'intégration et ses composants requis, voir les documents suivants.

Pour WebSphere InterChange Server (ICS), voir *IBM WebSphere InterChange Server System Installation Guide for UNIX* ou *for Windows*.
Pour les courtiers de messages WebSphere, voir *Implementing Adapters with WebSphere Message Brokers*.
Pour WebSphere Application Server, voir *Implementing Adapters with WebSphere Application Server*.

Plateformes de l'adaptateur

L'adaptateur est pris en charge sur les plateformes suivantes :

Systèmes d'exploitation :

- AIX(R) 5.1, AIX 5.2
- Solaris 8.0
- HP UX 11i
- Windows 2000

Bases de données :

- Oracle 9i

Logiciels tiers :

- Manugistics Demand and Fulfillment Management version 7.1.
- Manugistics NetWORKS Collaborate version 7.2

Dépendances de l'adaptateur

Si vous utilisez une base de données Oracle, vous devez installer les bibliothèques OracleOCI.

Internationalisation

Cet adaptateur est compatible avec le jeu de caractères à deux octets (DBCS) et est traduit.

Conditions préalables

Avant d'utiliser le connecteur, vous devez procéder comme suit :

- Installez le composant Adapter Development Kit uniquement si vous envisagez d'étendre ou de modifier les fonctions de l'adaptateur.

  Si le connecteur est exécuté sur une autre machine que celle sur laquelle réside le courtier d'intégration, installez la version d'Adapter Development Kit compatible avec celle du courtier d'intégration.
- Installez le pilote JDBC qui sera utilisé.
- Vérifiez que les logiciels requis spécifiques au fournisseur, notamment les composants requis pour le pilote JDBC, ont été installés.

  Par exemple, si vous utilisez le pilote JDBC Type 2 pour une base de données Oracle, vous devez installer les bibliothèques OracleOCI.
- Vérifiez l'existence d'un compte utilisateur dans l'application.

  Le connecteur traite les données dans les bases de données Manugistics prises en charge par les pilotes conformes aux spécifications JDBC. Pour que le connecteur puisse traiter les données dans la base de données, avec laquelle il communique directement, il doit avoir accès à un compte utilisateur et à un mot de passe valide pour l'application. Le compte utilisateur doit détenir les autorisations nécessaires pour extraire, insérer, mettre à jour et supprimer des

données depuis la base de données de l'application. Si vous ne disposez pas d'un compte utilisateur, vous devez en créer un.

- Vérifiez le jeu de codes de caractères de la base de données connectée.

  L'environnement d'exécution Java dans la machine virtuelle Java (JVM) représente les données dans le jeu de codes de caractères Unicode. Le format Unicode contient des codes pour les caractères présents dans la plupart des jeux de codes de caractères connus (à la fois mono-octet et multi-octets). Etant donné que le connecteur est rédigé en Java, il reconnaît le format Unicode.

Installation de l'adaptateur et des fichiers associés

Pour plus d'informations sur l'installation des produits de WebSphere Business Integration Adapter, voir *Installation Guide for 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 sous-sections suivantes présentent la structure de fichiers de l'adaptateur installée sur un système UNIX ou Windows.

Remarque : Sauf indication contraire, les autres sections de ce chapitre concernent à la fois les installations UNIX et Windows du connecteur.

Installation sur un système UNIX

Pour plus d'informations sur l'installation du composant Adapter for Manugistics sur un système UNIX, voir le document *Installation Guide for WebSphere Business Integration Adapters.*

Le tableau 2 décrit la structure des fichiers UNIX utilisée par le connecteur.

Tableau 2. Structure de fichiers UNIX installée pour le connecteur

Sous-répertoire de *ProductDir*	**Description**
`connectors/Manugistics`	Contient les fichiers BIA_Manugistics.jar et start_Manugistics.sh du connecteur. Le fichier start_Manugistics.sh est un script de démarrage du système destiné au connecteur. Il est appelé depuis le script du gestionnaire de connecteurs générique. Lorsque vous cliquez sur Install dans Connector Configurator (WebSphere MQ Integrator Broker comme courtier d'intégration) ou l'écran Connector Configuration de System Manager (InterChange Server comme courtier d'intégration), le composant Installer for IBM WebSphere Business Integration Adapter crée un encapsuleur personnalisé pour ce script de gestionnaire de connecteurs. Si le connecteur utilise InterChange Server, utilisez cet encapsuleur personnalisé pour démarrer et arrêter le connecteur. Si le connecteur utilise WebSphere MQ Integrator Broker, utilisez cet encapsuleur personnalisé pour démarrer et arrêter le connecteur. Utilisez la commande `mqsiremotestopadapter` pour arrêter le connecteur.

Tableau 2. Structure de fichiers UNIX installée pour le connecteur (suite)

Sous-répertoire de *ProductDir*	Description
connectors/Manugistics/dependencies	Contient les scripts SQL qui permettent de créer les tables d'identificateurs uniques, les tables d'événements et d'archivage.
connectors/messages	Contient le fichier BIA_ManugisticsAdapter.txt.
repository/Manugistics	Contient le fichier BIA_CN_Manugistics.txt.
/lib	Contient le fichier WBIA.jar.
/bin	Contient le fichier CWConnEnv.sh.

Pour plus d'informations sur l'installation du composant du connecteur, voir les documents suivants, selon le courtier d'intégration utilisé :

- *System Installation Guide for UNIX* (lorsqu'InterChange Server est utilisé comme courtier d'intégration)
- *IBM WebSphere Business Integration Adapters Implementation Guide for WebSphere MQ Integrator Broker (lorsque WebSphere MQ Integrator Broker est utilisé comme courtier d'intégration)*

Structure de fichiers installée sur un système Windows

Pour plus d'informations sur l'installation du composant Adapter for Manugistics sur un système Windows, voir le document *Installation Guide for WebSphere Business Integration Adapters.* Le tableau 3 décrit la structure des fichiers Windows utilisée par le connecteur.

Tableau 3. Structure de fichiers Windows installée pour le connecteur

Sous-répertoire de *ProductDir*	Description
connectors\Manugistics	Contient les fichiers BIA_Manugistics.jar et start_Manugistics.bat du connecteur.
connectors\Manugistics\dependencies	Contient les scripts SQL qui permettent de créer les tables d'identificateurs uniques, les tables d'événements et d'archivage.
connectors\messages	Contient le fichier BIA_ManugisticsAdapter.txt.
repository\Manugistics	Contient le fichier BIA_CN_Manugistics.txt.
\lib	Contient le fichier WBIA.jar.
\bin	Contient le fichier CWConnEnv.bat.

Le programme d'installation ajoute une icône correspondant au fichier du connecteur au menu IBM WebSphere Business Integration Adapters. Pour permettre de démarrer rapidement le connecteur, créez un raccourci vers ce fichier sur le bureau.

Pour plus d'informations sur l'installation du composant du connecteur, voir les documents suivants, selon le courtier d'intégration utilisé :

- *System Installation Guide for Windows* (lorsqu'InterChange Server est utilisé comme courtier d'intégration)
- *IBM WebSphere Business Integration Adapters Implementation Guide for WebSphere MQ Integrator Broker* (lorsque WebSphere MQ Integrator Broker est utilisé comme courtier d'intégration)

Activation de Manugistics pour le connecteur

Cette section aborde les points suivants :

- «Configuration des tables de génération d'interfaces Manugistics»
- «Configuration du traitement des événements et de l'archivage»
- «Tables d'événements et d'archivage» à la page 14
- «Scripts SQL pour l'installation des tables d'événements et d'archivage» à la page 15
- «Schéma des tables d'événements et d'archivage» à la page 16

Configuration des tables de génération d'interfaces Manugistics

Le composant Adapter for Manugistics s'intègre à l'aide de l'application Interface Generation Program (IGP) et de ses tables d'interface à la connexion JDBC et aux objets métier de WebSphere Business Integration Adapters qui permettent de créer, mettre à jour, supprimer et extraire des données de la base de données. L'IGP est un processus dans lequel les tables intermédiaires sont utilisées pour propager des données à des tables de base de données actives, via l'exécution de différentes procédures propres à l'application.

Pour que les requêtes d'appels de service puissent créer et mettre à jour les valeurs des données dans la base de données, l'adaptateur utilisera les tables d'interface IGP comme emplacement intermédiaire pour les valeurs. Ce processus nécessite de placer des données dans les tables d'interface qui doivent être traitées par les procédures stockées, puis déplacées vers les tables actives. Ces procédures stockées contrôlent l'intégrité des données.

Vous devrez exécuter et configurer l'application Manugistics Interface Generation Program pour créer des opérations Insert, Update et Upsert pour les données de la table d'intégration souhaitées. Pour plus d'informations, voir *Manugistics WebWorks Guide*.

Vous devez configurer le mécanisme de notification des événements dans la base de données pour que le connecteur puisse effectuer la livraison des événements. Pour ce faire, procédez comme suit :

- Créez les tables d'événements et d'archivage dans la base de données.
- Installez les déclencheurs de la base de données sur les tables de l'application afin de prendre en charge les objets métier requis. Vous êtes chargé de développer vos propres déclencheurs de base de données.
- Le cas échéant, installez un table de compteurs. Effectuez cette procédure uniquement si vous souhaitez que le connecteur crée un ID unique lors de la création d'un objet métier. Pour plus d'informations sur la création d'ID uniques, voir le paramètre `UID=CW.uidcolumnname[=UseIfMissing]`.

Les sections qui suivent expliquent comment créer et configurer les tables d'événements et d'archivage.

Configuration du traitement des événements et de l'archivage

Pour configurer le traitement des événements et de l'archivage, vous devez utiliser les propriétés de configuration pour spécifier les informations suivantes :

- Le nom de la table d'événements (EventTableName). Vous ne devez pas indiquer de valeur pour cette propriété si vous utilisez le connecteur uniquement pour traiter les requêtes des objets métier.
- L'intervalle de fréquence («PollFrequency» à la page 111).
- Le nombre d'événements pour chaque intervalle d'interrogation (PollQuantity).
- Le nom de la table d'archivage (ArchiveTableName).
- Si le connecteur archive des événements non inscrits et non traités (ArchiveProcessed). Pour obtenir des informations d'inscription spécifiques au courtier d'intégration, voir le guide d'implémentation du courtier.
- L'ID unique du connecteur, qui est particulièrement important lorsque plusieurs connecteurs interrogent la même table (ConnectorID).

Vous pouvez également indiquer une valeur pour la propriété EventOrderBy afin d'indiquer l'ordre de traitement des événements. Pour plus d'informations sur les propriétés de configuration, voir l'Annexe A, «Propriétés de configuration standard pour les connecteurs», à la page 97 et le tableau 6 à la page 19.

Remarque : La création des tables d'événements et d'archivage est facultative. Toutefois, si vous indiquez une valeur pour la propriété EventTableName mais que vous n'utilisez pas le connecteur pour consulter les événements et que vous ne créez pas de table d'événements, le connecteur s'arrête. Pour éviter ce délai d'expiration, conservez la valeur de la propriété EventTableName comme étant null (en tant que chaîne).

Par défaut, le nom de la table de la file d'attente des événements est xworlds_events et le nom de la table de la file d'attente d'archivage est xworlds_archive_events.

Pour utiliser le connecteur uniquement pour le traitement des requêtes, utilisez l'option -fno au démarrage et affectez la valeur null (en tant que chaîne) à la propriété EventTableName.

Si le pilote utilisé ne prend pas en charge la classe Java DatabaseMetaData et que vous voulez empêcher le connecteur de vérifier l'existence des tables d'événements et d'archivage, désactivez la propriété CheckForEventTableInInit en lui affectant la valeur false. Par défaut, la valeur affectée est true. Il est recommandé de ne pas affecter la valeur false.

Remarque : Si votre site n'archive pas les événements dans la table d'archivage, affectez à la propriété ArchiveProcessed la valeur false.

Tables d'événements et d'archivage

Le connecteur utilise la table d'événements pour mettre en file d'attente les événements en vue de les extraire. Si vous avez affecté à la propriété ArchiveProcessed la valeur "true" ou aucune valeur, le connecteur utilise la table d'archivage pour stocker les événements après avoir mis à jour leur état dans la table d'événements.

Pour chaque événement, le connecteur extrait de la table d'événements le nom, l'instruction et la clé de l'objet métier. Le connecteur utilise ces informations pour récupérer intégralement l'entité de l'application. Si l'entité a été modifiée après la première journalisation de l'événement, le connecteur extrait l'événement initial et toutes les modifications qui suivent. En d'autres termes, si une entité est créée et

mise à jour avant que le connecteur ne l'extraie de la table d'événements, le connecteur extrait les deux modifications apportées aux données en une seule fois.

Les trois sorties suivantes sont possibles pour chaque événement traité par un connecteur :

- L'événement a été traité correctement.
- L'événement n'a pas été traité correctement.
- L'événement n'a pas reçu d'inscription (pour obtenir des informations d'inscription spécifiques au courtier d'intégration, voir le guide d'implémentation du courtier).

Si les événements n'ont pas été supprimés de la table d'événements une fois que le connecteur les a extraits, ils occupent inutilement de la place. Cependant, si vous les supprimez, tous les événements qui ne sont pas traités sont perdus et vous ne pouvez effectuer aucun audit du traitement des événements. Par conséquent, il est recommandé de créer également une table d'archivage et de conserver la valeur "true" pour la propriété `ArchiveProcessed`. Chaque fois qu'un événement est supprimé de la table d'événements, le connecteur l'insère tout d'abord dans la table d'archivage.

Remarque : Si des problèmes d'accès à la base de données de l'application entraînent l'échec du connecteur pendant la suppression d'un événement de la table d'événements ou pendant l'insertion de l'événement dans la table d'archivage, le connecteur renvoie un message d'erreur APPRESPONSETIMEOUT.

Scripts SQL pour l'installation des tables d'événements et d'archivage

Ces scripts sont fournis uniquement comme modèle pour vous aider à créer les tables requises pour le connecteur. Pour les autres bases de données, créez vos scripts en les utilisant comme instructions.

Les scripts utilisés pour installer les tables d'identificateurs uniques, d'événements et d'archivage pour une base de données Oracle sont les suivantes :

- `event_table_oracle.sql`
- `event_package_oracle.sql`
- `archive_table_oracle.sql`
- `uid_table_oracle.sql`

Ces fichiers résident dans les répertoires suivants :

UNIX :

```
connectors/Manugistics/dependencies/
```

Windows :

```
connectors\Manugistics\dependencies\
```

Remarque : L'ordre et le type de données dans les colonnes de la table sont très importants. Voir «Schéma des tables d'événements et d'archivage» à la page 16 pour connaître l'ordre et le type de données appropriés.

Il est recommandé à l'administrateur de la base de données ou à la personne chargée d'implémenter le connecteur de modifier ces scripts afin qu'ils soient

conformes aux spécifications d'installation et d'optimisation des requêtes. Par exemple, ces scripts ne créent pas d'index dans les tables. Il revient à la personne chargée d'implémenter le connecteur de créer des index afin d'améliorer les performances de l'optimiseur de requête.

Schéma des tables d'événements et d'archivage

Le tableau 4 décrit le contenu des colonnes des tables d'événements et d'archivage.

Tableau 4. Schéma des tables d'événements et d'archivage

Nom	Description	Type	Contrainte
event_id	Identificateur interne de l'événement	NUMBER	Clé primaire
connector_id	ID unique du connecteur à qui est destiné l'événement. Cette valeur est importante lorsque plusieurs connecteurs interrogent la même table.	VARCHAR	
object_key	Clé primaire de l'objet métier. La clé peut être représentée comme une paire nom_valeur ou comme un ensemble de clés délimitées par un signe deux-points ou un autre délimiteur que vous devez définir (par exemple, 1000065:10056:2333). Pour plus d'informations, voir la propriété «EventKeyDel» à la page 24.	VARCHAR	Non `null`
object_name	Nom de l'objet métier	VARCHAR	Non `null`
object_verb	Instruction associée à l'événement	VARCHAR	Non `null`
event_priority	Priorité de l'événement (0 est la valeur la plus élevée, *n* est la plus faible) que le connecteur utilise pour extraire les événements suivant un ordre de priorité. Le connecteur n'utilise pas cette valeur pour diminuer ou augmenter les priorités.	NUMBER	Non `null`
event_time	Date et heure à laquelle s'est produit l'événement	DATETIME	Date/heure actuelle (pour la table d'archivage, l'heure de l'événement en cours)
archive_time	Date et heure à laquelle l'événement a été archivé (concerne uniquement la table d'archivage)	DATETIME	Date/heure d'archivage
event_status	`-2` (Erreur d'envoi de l'événement au courtier d'intégration) `-1` (Erreur de traitement de l'événement) `0` (Prêt à interroger) `1` (Envoyé au courtier d'intégration) `2` (Pas d'inscription pour l'objet métier) `3` (En cours). Cet état est utilisé uniquement dans la table d'événements et non dans la table d'archivage.	NUMBER	Non `null`
event_comment	Description de l'événement ou de la chaîne d'erreur	VARCHAR	

Activation de la prise en charge de plusieurs pilotes

Vous pouvez spécifier le pilote à utiliser en procédant comme suit :

1. Installez le pilote sur votre machine.
2. Placez toutes les bibliothèques dynamiques que le connecteur requiert au moment de l'exécution dans le répertoire `connectors/Manugistics` sous le répertoire produit.

3. Modifiez le fichier de démarrage du connecteur afin d'inclure tous les noms de chemin des classes correspondantes (notamment les informations sur la licence, le cas échéant) dans la variable JDBCDRIVERPATH.

   Sous UNIX, le fichier de démarrage est le suivant :

   ProductDir/connectors/Manugistics/start_Manugistics.sh

   Sous Windows, le fichier de démarrage est le suivant :

   ProductDir\connectors\Manugistics\start_Manugistics.bat

4. Indiquez une valeur pour la propriété de configuration JDBCDriverClass.

   Remarque : Pour toutes les fonctions qu'il prend en charge, le connecteur peut utiliser n'importe quel pilote conforme aux spécifications JDBC 2.0 ou suivant. Si le pilote ne prend pas en charge une caractéristique en particulier, le connecteur ne peut pas fonctionner correctement. Par exemple, si le pilote ne prend pas en charge tous les appels de méthode utilisés par le composant ODA for Manugistics, le fichier journal d'ODA indique le processus que le pilote ne prend pas en charge. Dans ce cas, vous devez utiliser un autre pilote.

Activation de la classe du gestionnaire d'objets métier personnalisé

Le connecteur prend en charge la classe du gestionnaire d'objets métier personnalisé, CustomBOH. Il implémente l'interface JDBCBOhandlerInterface. La syntaxe de cette interface est la suivante :

```
public interface JDBCBOhandlerInterface{
   public int doVerbForCustom(CWConnectorBusObj busObj) throws
     VerbProcessingFailedException, ConnectionFailureException;
}
```

Lorsque vous implémentez la méthode doVerbForCustom, assurez-vous qu'elle envoie les deux exceptions mais qu'elle ne les intercepte pas. En outre, définissez l'état et le message de chaque exception avant de les envoyer.

- VerbProcessingFailedException : envoyée lorsque l'opération indiquée par l'instruction échoue.
- ConnectionFailureException : envoyée lorsque le connecteur ne peut pas se connecter à l'application.

Pour permettre au connecteur de prendre en charge ce gestionnaire d'objets métier :

- Indiquez le nom de classe CustomBOH dans les informations spécifiques à l'application de l'instruction.

  Le connecteur extrait le nom de la classe du gestionnaire d'objets métier personnalisé des informations spécifiques à l'application de l'instruction. Utilisez la syntaxe suivante :

  CustomBOH=*customBOhandlerClassName*

  Par exemple, supposons que les informations spécifiques à l'application de l'instruction soient indiquées comme suit :

  CustomBOH=JDBCBOhandlerForOverrideSQL

  Dans le cas présent, JDBCBOhandlerForOverrideSQL correspond au nom de la classe du gestionnaire d'objets métier personnalisé.

- Assurez-vous que la classe CustomBOH appartient au module com.crossworlds.connectors.JDBC.

  Si le connecteur recherche la classe "CustomBOH" dans les informations spécifiques à l'application de l'instruction et la trouve dans le module

com.crossworlds.connectors.JDBC, il exécute ce gestionnaire d'objets métier personnalisé. S'il ne trouve pas la classe CustomBOH, il émet un message d'erreur indiquant que la classe est introuvable.

Configuration du connecteur

Vous devez au préalable définir les propriétés de configuration standard et les propriétés spécifiques au connecteur pour pouvoir l'exécuter. Pour définir les propriétés de configuration d'un connecteur, utilisez l'outil Connector Configurator :

- Si InterChange Server est le courtier d'intégration, accédez à cet outil à partir de System Manager.
- Si WebSphere MQ Integrator Broker est le courtier d'intégration, accédez à cet outil depuis le dossier programme IBM WebSphere Business Integration Adapter.

Pour plus d'informations sur Connector Configurator, voir Annexe B, «Connector Configurator», à la page 115.

Propriétés de configuration standard

Les propriétés de configuration standard fournissent des informations destinées aux connecteurs. Pour plus d'informations sur ces propriétés, voir Annexe A, «Propriétés de configuration standard pour les connecteurs», à la page 97.

Important : Etant donné que le connecteur pour JDBC prend en charge les courtiers d'intégration InterChange Server et WebSphere MQ Integrator Broker, les propriétés de configuration des deux courtiers correspondent au connecteur.

Par ailleurs, voir le tableau 5 pour connaître les informations de configuration spécifiques à IBM WebSphere Business Integration Adapter for JDBC. Les informations contenues dans ce tableau complètent les informations figurant dans l'annexe.

Tableau 5. Informations sur les propriétés spécifiques à ce connecteur

Propriété	Remarques
CharacterEncoding	Ce connecteur n'utilise pas la propriété CharacterEncoding
Locale	Etant donné que ce connecteur est internationalisé, vous pouvez modifier la valeur de la propriété Locale. **Remarque :** Si vous utilisez WebSphere MQ Integrator Broker comme courtier, vous devez utiliser le même environnement local pour l'adaptateur, le courtier et les applications.

Notez que vous devez indiquer une valeur pour la propriété de configuration ApplicationName afin d'exécuter le connecteur.

Propriétés spécifiques au connecteur

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

Le tableau 6 contient les propriétés de configuration spécifiques au connecteur. Pour obtenir une explication des propriétés, voir les sections suivantes.

Tableau 6. Propriétés de configuration spécifiques au connecteur

Nom	Valeurs possibles	Valeur par défaut	Obligatoire
ApplicationPassword	*Mot de passe pour le compte utilisateur du connecteur*		Oui*
ApplicationUserName	*Nom du compte utilisateur du connecteur*		Oui*
ArchiveProcessed	true ou false	true	Non
ArchiveTableName	*Nom de la table de la file d'attente d'archivage*	xworlds_archive_events	Oui si la valeur affectée à Archive Processed est true
AutoCommit	true ou false	false	Non
CheckforEventTableInInit	true ou false	true	Non
ChildUpdatePhyDelete	true ou false	false	Non
CloseDBConnection	true ou false	false	Non
ConnectorID	*ID unique du connecteur*	null	Non
DatabaseURL	*Nom du serveur de base de données*		Oui
DateFormat	*Chaîne de modèle de date*	MM/jj/aaaaa HH:mm:ss	Non
DriverConnectionProperties	*Propriétés de connexion du pilote JDBC supplémentaires*		Non
DriverSupportForLong	true ou false	true	Non
EventKeyDel	*Caractères de délimitation pour la colonne de la clé de l'objet de la table d'événements*	point-virgule (;)	Non
EventOrderBy	aucune, *ColumnName*, *ColumnName*, ...]		Non
EventQueryType	Fixed ou Dynamic	Fixed	Non
EventTableName	*Nom de la table de la file d'attente des événements*	xworlds_events	Oui, si l'interrogation est requise ; null (comme chaîne) si l'interrogation n'est pas requise
JDBCDriverClass	nom de classe du pilote		Oui
MaximumDatabaseConnections	*Nombre de connexions simultanées à une base de données*	5	Oui
PingQuery	SELECT 1 FROM <*tablename*>		Non
PollQuantity	Les valeurs sont comprises entre 1 et 500	1	Non
PreserveUIDSeq	true ou false	true	Non
QueryTimeOut	Valeur d'entier en secondes		Non
RDBMS.initsession	*Instruction SQL qui initialise toutes les sessions de base de données*		Non
RDBMSVendor	Oracle, *Others*		Oui
ReplaceAllStr	true ou false	false	Non
ReplaceStrList	Ensemble composé d'un seul caractère, d'un caractère de délimitation et de la chaîne de substitution du caractère. Comprend également plusieurs ensembles séparés par des caractères de fin.	Q,DSQRemarque : Dans l'outil de configuration du connecteur, ces caractères représentent un guillemet simple, suivi d'une virgule et de deux guillemets simples.	Non

Tableau 6. Propriétés de configuration spécifiques au connecteur (suite)

Nom	Valeurs possibles	Valeur par défaut	Obligatoire
RetryCountAndInterval	*Nombre, intervalle en secondes*	3,20	Non
ReturnDummyBOForSP	true ou false	false	Non
SchemaName	Schéma dans lequel les événements résident		Non
SelectiveCommitForPoll	true ou false	false	Non
SPBeforePollCall	Nom de la procédure stockée à exécuter pour chaque appel d'interrogation		Non
StrDelimiter	Délimiteurs de caractères et de fin utilisés dans la propriété ReplaceStrList	,:	Non
TimingStats	0, 1, 2	0	Non
UniqueIDTableName	*Nom de la table utilisée pour créer les ID*	xworlds_uid	Non
UseDefaults	true ou false	false	Oui
UseDefaultsForCreatingChildBOs	true ou false	false	Non
UseDefaultsWhenPolling	true ou false	true	Non

*Les propriétés ApplicationPassword et ApplicationUserName ne sont pas obligatoires si vous utilisez une authentification sécurisée.

ApplicationPassword

Mot de passe défini pour le compte utilisateur du connecteur.

Il n'existe pas de valeur par défaut.

ApplicationUserName

Nom du compte utilisateur du connecteur.

Il n'existe pas de valeur par défaut.

ArchiveProcessed

Indique si le connecteur archive des événements pour lesquels il n'existe pas d'inscription en cours.

Affectez à cette propriété la valeur true pour que les événements soient insérés dans la table d'archivage avant d'être supprimés de la table d'événements.

Affectez à cette propriété la valeur false pour que le connecteur ne procède pas à l'archivage. Dans ce cas, il ne vérifie pas la valeur de la propriété ArchiveTableName. Si la propriété ArchiveProcessed a la valeur false, le connecteur procède comme suit :

- Si l'événement est traité correctement, le connecteur le supprime de la table d'événements et ne l'archive pas.
- Si le connecteur ne s'inscrit pas à l'objet métier de l'événement, il conserve l'événement dans la table d'événements et lui affecte l'état Unsubscribed. Pour obtenir des informations d'inscription spécifiques au courtier d'intégration, voir le guide d'implémentation du courtier.
- Si un incident se produit pendant le traitement de l'objet métier, le connecteur conserve l'événement dans la table d'événements avec l'état Error.

Si cette propriété a la valeur false et que le nombre d'interrogations est faible, le connecteur semble interroger la table d'événements mais est simplement en train de prélever de manière répétée les mêmes événements.

Si cette propriété ne contient pas de valeur, le connecteur présume que la valeur est true. Si la propriété ArchiveTableName ne contient pas de valeur, le connecteur présume que le nom de la table d'archivage est xworlds_archive_events.

La valeur par défaut est true.

ArchiveTableName

Nom de la table de la file d'attente d'archivage.

Si la propriété ArchiveProcessed a la valeur false, il n'est pas nécessaire de définir une valeur pour cette propriété.

Le nom par défaut est xworlds_archive_events.

AutoCommit

Cette propriété permet de configurer le paramètre AutoCommit. Lorsque ce paramètre a la valeur true, toutes les transactions sont automatiquement validées. Certaines bases de données (comme Sybase) requièrent que le paramètre AutoCommit ait la valeur true. Si la valeur affectée est false, les procédures stockées sur Sybase échouent.

Si la connexion à la base de données est interrompue, le connecteur tentera de rétablir la connexion afin de redémarrer le traitement complet tant que le paramètre AutoCommit a la valeur false. Si la nouvelle connexion est incorrecte ou que le paramètre AutoCommit a la valeur true, le connecteur renvoie le message d'erreur APPRESPONSETIMEOUT, qui entraîne la fermeture du connecteur.

La valeur par défaut est false.

CheckforEventTableInInit

Si vous affectez la valeur false à cette propriété, le connecteur ne peut pas vérifier l'existence des tables d'événements et d'archivage pendant son initialisation. Il est recommandé de toujours affecter la valeur true sauf si le pilote JDBC que vous utilisez ne prend pas en charge la classe JDBC DatabaseMetaData.

Lorsque la propriété a la valeur false, même si le connecteur ne vérifie pas l'existence des tables EventTable et ArchiveTable, les tables d'événements et d'archivage doivent toujours exister car le connecteur les utilise pendant le processus d'initialisation. Pour empêcher le connecteur d'utiliser les tables d'événements et d'archivage pendant l'initialisation, affectez à la propriété EventTableName la valeur null.

La valeur par défaut est true.

ChildUpdatePhyDelete

Pendant une opération de mise à jour, cette propriété indique comment le connecteur traite les données représentées par un objet métier enfant qui est absent de l'objet métier entrant mais qui existe dans la base de données.

Affectez à cette propriété la valeur true pour que le connecteur supprime physiquement les données de la base de données.

Affectez à cette propriété la valeur `false` pour que le connecteur supprime de manière logique les données de la base de données en affectant à la colonne d'état la valeur appropriée. Les informations spécifiques à l'application extraient le nom de la colonne d'état et la valeur associée du paramètre StatusColumnValue (SCN) indiqué dans ses informations spécifiques à l'application au niveau de l'objet métier. Pour plus d'informations, voir «Informations spécifiques à l'application au niveau de l'objet métier» à la page 60.

La valeur par défaut est `false`.

CloseDBConnection

Cette propriété permet de configurer la fermeture de la connexion à la base de données. Lorsque la valeur affectée est `true`, pour chaque requête d'appel de service et d'interrogation, la connexion à la base de données est arrêtée. La valeur `true` affecte les performances, et de ce fait n'est pas recommandée.

La valeur par défaut est `false`.

ConnectorID

ID unique du connecteur. Cet ID permet d'extraire les événements d'une instance spécifique du connecteur.

La valeur par défaut est `null`.

DatabaseURL

Nom du serveur de la base de données à laquelle le connecteur doit se connecter.

Si vous utilisez le pilote SQLServer personnalisé pour WebSphere Business Integration, l'adresse URL recommandée est :

```
jdbc:ibm-crossworlds:sqlserver://MachineName:PortNumber;
DatabaseName=DBname
```

Important

Si `AutoCommit` a la valeur `false`, vous devez définir un autre paramètre, `SelectMethod` : `jdbc:ibm-crossworlds:sqlserver://MachineName:PortNumber;DatabaseName=DBname;SelectMethod=cursor`

Par défaut, `SelectMethod` a la valeur `direct`. Pour plus d'informations, voir «AutoCommit» à la page 21.

Vous devez indiquer une valeur pour cette propriété pour que le connecteur puisse fonctionner correctement.

DateFormat

Indique le format de date que le connecteur s'attend à recevoir et à renvoyer. Cette propriété prend en charge tous les formats basés sur la syntaxe indiquée dans le tableau 7 à la page 23.

Le tableau 7 à la page 23 définit la syntaxe de la propriété DateFormat à l'aide d'une chaîne de modèle de date. Dans ce modèle, toutes les lettres ASCII sont réservées comme étant des lettres modèle.

Tableau 7. Syntaxe de DateFormat

Symbole	Signification	Présentation	Exemple
G	ère	(Text)	ap J.-C.
y	année	(Number)	1996
M	mois de l'année	(Text & Number)	Juillet & 07
d	jour du mois	(Number)	10
h	heure am/pm (1-12)	(Number)	12
H	heure du jour (0-23)	(Number)	0
m	minute	(Number)	30
s	seconde	(Number)	55
S	milliseconde	(Number)	978
E	jour de la semaine	(Text)	mardi
D	jour de l'année	(Number)	189
F	jour de la semaine dans le mois	(Number)	2 (2ème mercredi de juillet)
w	semaine de l'année	(Number)	27
W	semaine dans le mois	(Number)	2
a	indicateur am/pm	(Text)	PM
k	heure du jour (1-24)	(Number)	24
K	heure am/pm (0-11)	(Number)	0
z	fuseau horaire	(Text)	Heure standard du Pacifique
'	caractère d'échappement	(Delimiter)	
''	guillemet simple	(Literal)	'

Tableau 8. Exemples avec l'environnement local américain

Modèle de format	Résultat
"yyyy.MM.dd G 'à' hh:mm:ss z"	1996.07.10 ap J.-C. à 15:08:56 PDT
"EEE, MMM d, ''yy"	Mer, Juillet 10, '96
"h:mm a"	12:08 PM
"hh 'o''clock' a, zzzz"	12 heures PM, Heure du Pacifique
"K:mm a, z"	0:00 PM, PST
"yyyy.MMMMM.dd GGG hh:mm aaa"	1996.juillet.10 ap J.-C. 12:08 PM

DriverConnectionProperties

Outre le nom d'utilisateur et le mot de passe, un pilote JDBC peut nécessiter des propriétés ou des informations supplémentaires. La propriété du connecteur `DriverConnectionProperties` prendra les propriétés supplémentaires que requiert un pilote JDBC, sous la forme de paires nom-valeur. Les propriétés doivent être indiquées comme suit :

```
property1=value1[;property2=value2...]
```

Les propriétés doivent être indiquées sous la forme de paires nom-valeur, séparées par des points-virgules. La propriété est séparée de sa valeur par un signe égal (sans espace).

Par exemple, imaginons que le pilote JDBC requiert des informations sur la licence et le numéro de port. Le nom de propriété attendu pour les informations sur la licence est `MyLicense` et la valeur correspondante est `ab23jk5`. Le nom de propriété attendu pour le numéro de port est `PortNumber` et la valeur correspondante est `1200`. La propriété `DriverConnectionProperties` doit avoir la valeur `MyLicense=ab23jk5;PortNumber=1200`.

DriverSupportForLong
Cette propriété indique comment les paramètres de type entier sont traités pour PreparedStatements. Lorsque cette propriété a la valeur ″true″, la valeur `setLong` est utilisée pour définir le paramètre de type entier. Lorsqu'elle a la valeur ″false″, la propriété `setInt` est utilisée pour définir le paramètre.

La valeur par défaut est true.

EventKeyDel
Cette propriété indique le délimiteur utilisé lorsque la colonne `object_key` de la table d'événements contient plusieurs valeurs d'attribut.

Il existe deux manières d'extraire l'objet métier qui a été créé, mis à jour ou supprimé dans l'application de déclenchement.

- La première méthode consiste à indiquer dans la colonne `object_key` les valeurs des attributs clés contenus dans un objet métier. Affectez à la propriété de configuration `EventKeyDel` un caractère unique qui ne fait pas partie de la zone de clé. **Par exemple**, si le délimiteur est "**;**", la colonne `object_key` se présentera comme suit : *xxx ;123*
- La seconde méthode consiste à indiquer dans la colonne `object_key` les valeurs des attributs contenus dans un objet métier. Ces valeurs doivent être représentées comme une paire `nom_valeur`. Le premier délimiteur est réservé à la paire `nom_valeur` et le second est réservé aux clés. Par exemple, si le délimiteur est "**=;**", la colonne `object_key` se présentera comme suit : *CustomerName =xxx;CustomerId=123*;

  Si le délimiteur est "**=:**", la colonne `object_key` se présentera comme suit : *CustomerName =xxx:CustomerId=123*:

Remarque : L'ordre dans lequel les valeurs clés sont définies doit suivre le même ordre que les attributs de clé dans un objet métier.

Important : Si vous utilisez des données d'attributs Date, évitez d'utiliser un signe deux-points (:), car celui-ci peut-être inclus dans les données d'attributs.

La valeur par défaut est un point-virgule (;), qui est basé sur les clés et non pas sur des paires `nom_valeur`.

EventOrderBy
Cette propriété indique si l'ordre des événements est activé ou désactivé, ou indique un ordre de traitement des événements différent de l'ordre par défaut.

Par défaut, à chaque interrogation, le connecteur extrait uniquement le nombre d'événements indiqué dans sa propriété `PollQuantity`, et organise le traitement des événements par les valeurs contenues dans les colonnes event_time et event_priority de la table d'événements.

Si vous ne souhaitez pas que le connecteur trie les événements, affectez à cette propriété la valeur `none`.

Si vous ne souhaitez pas que le connecteur ordonne les différentes colonnes dans la table d'événements, indiquez le nom de ces colonnes. Séparez les noms de colonnes par une virgule (**,**). Si vous indiquez une valeur pour cette propriété, vous remplacez le comportement par défaut.

Il n'existe pas de valeur par défaut pour cette propriété.

EventQueryType

La propriété `EventQueryType` est utilisée pour indiquer si le connecteur doit créer dynamiquement une requête en vue d'extraire les événements de la table d'événements ou utiliser sa requête intégrée. Pour la requête créée de manière dynamique, le connecteur mappe sa structure d'événements aux colonnes dans la table d'événements. L'ordre des données dans les colonnes de la table est très important. Voir «Schéma des tables d'événements et d'archivage» à la page 16 pour connaître l'ordre approprié.

Si la valeur affectée à la propriété `EventQueryType` est `Fixed` (comme chaîne), la requête par défaut est exécutée. Si la valeur affectée est `Dynamic` (comme chaîne), une nouvelle requête est créée en utilisant les noms de colonne de la table indiquée dans la propriété «EventTableName».

Les noms de colonne de la table d'événements peuvent changer, cependant l'ordre et le type de données des colonnes doivent être conservés tel que spécifiés dans la section relative à la création d'une table d'événements. La propriété «EventOrderBy» à la page 24 sera ajoutée à la requête par défaut ou à la requête créée de manière dynamique.

Si la propriété `EventQueryType` n'est pas ajoutée ou qu'elle ne contient pas de valeur, elle a par défaut la valeur `Fixed`.

La valeur par défaut est `Fixed` (comme chaîne).

EventTableName

Nom de la table de la file d'attente des événements, utilisée par le système d'interrogation du connecteur.

Le nom par défaut est `xworlds_archive_events`.

Affectez à cette propriété la valeur `null` (comme chaîne) lorsque l'interrogation est désactivée pour le connecteur. De cette manière, le connecteur ne recherche pas l'existence des tables d'événements et d'archivage.

Pour une table d'événements définie par l'utilisateur, vérifiez que `event_id` correspond à l'un des types JDBC suivants : `INTEGER` , `BIGINT`, `NUMERIC`, `VARCHAR`.

JDBCDriverClass

Indique le nom de classe d'un pilote. Pour utiliser un pilote JDBC spécifique, indiquez le nom de classe du pilote dans cette propriété de configuration. Par exemple, pour indiquer le pilote Thin Oracle, affectez à cette propriété le nom de classe : `oracle.jdbc.driver.OracleDriver`.

Pour plus d'informations, voir «Activation de la prise en charge de plusieurs pilotes» à la page 16 et «UseDefaultsForCreatingChildBOs» à la page 31.

Aucune valeur par défaut n'est indiquée.

MaximumDatabaseConnections

Indique le nombre maximal de connexions simultanées autorisées à la base de données. Au moment de l'exécution, le nombre de connexions ouvertes à une base de données équivaut à la somme de cette valeur plus 1.

Si la propriété «PreserveUIDSeq» a la valeur ″false″, au moment de l'exécution, le nombre de connexions ouvertes à une base de données équivaut à la somme de cette valeur plus 2.

La valeur par défaut est 5.

PingQuery

Indique l'instruction SQL ou la procédure stockée que le connecteur exécute pour vérifier la connectivité à la base de données.

Voici un exemple d'instruction SQL utilisée comme requête ping :

```
SELECT 1 FROM <tablename>
```

Voici un exemple d'appel de procédure stockée (sampleSP) utilisée comme requête ping avec une base de données Oracle :

```
call sampleSP( )
```

Notez que les appels de procédure stockée ne peuvent pas comporter de paramètres de sortie. Si un paramètre d'entrée est requis par la base de données, la valeur d'entrée doit être indiquée dans la requête ping. Par exemple :

```
Call checkproc(2)
```

Il n'existe pas de valeur par défaut. Pour plus d'informations, voir «Gestion des connexions interrompues à une base de données» à la page 6 et «Interruption de la connexion à l'application» à la page 94.

PollQuantity

Nombre de lignes présentes dans une table de la base de données que le connecteur extrait par intervalle d'interrogation. Les valeurs autorisées sont comprises entre 1 et 500.

La valeur par défaut est 1.

PreserveUIDSeq

Indique si l'ID unique entrant sera conservé ou non dans la table d'identificateurs uniques.

Si cette propriété a la valeur `true`, l'ID unique n'est pas validé tant que l'objet métier n'est pas correctement traité dans l'application cible. Tous les autres processus qui tentent d'accéder à la table d'identificateurs uniques doivent attendre la validation de la transaction.

Si la valeur affectée à la propriété est `false`, l'ID unique est validé lorsque l'objet métier le demande. Les traitements de l'objet métier et de l'ID unique disposent chacun d'un bloc de transactions (interne au connecteur). Cela est possible uniquement si la transaction associée à la table d'identificateurs uniques possède sa propre connexion.

Remarque : Si cette propriété n'est pas ajoutée à la configuration du connecteur, le comportement par défaut reviendrait à ajouter une propriété et à lui affecter la valeur `true`. Par ailleurs, si la propriété «AutoCommit» à la page 21 a la valeur `true`, le connecteur procède comme si `PreserveUIDSeq` avait la valeur `false`.

Si la propriété «PreserveUIDSeq» à la page 26 a la valeur ″false″, au moment de l'exécution, le nombre de connexions ouvertes à une base de données équivaut à la somme de cette valeur plus 2.

La valeur par défaut est `true`.

QueryTimeOut

La valeur de cette propriété correspond à un nombre entier exprimé en secondes qui définit la propriété QueryTimeOut pour toutes les extractions par le nombre indiqué. Si aucune valeur n'est indiquée, cela signifie qu'aucun délai d'expiration n'est défini pour la requête. Si la requête prend plus de temps que le nombre de secondes indiqué, la base de données génère une exception SQL qui est enregistrée. Le message associé est consigné dans le fichier journal.

Aucune valeur par défaut n'est fournie.

RDBMS.initsession

Instruction SQL qui initialise toutes les sessions de base de données. Le connecteur émet une requête et l'exécute au démarrage. Il ne doit pas exister de valeur de retour pour cette requête. Le nom de la propriété est obligatoire, mais pas la valeur.

Il n'existe pas de valeur par défaut.

RDBMSVendor

Indique quel SGBDR est utilisé par le connecteur pour un traitement spécifique. Affectez à cette propriété la valeur `Oracle` pour la base de données Oracle.

Si vous utilisez une base de données autre que celle par défaut, vérifiez que le pilote chargé est approprié. Si cette propriété a la valeur `Others`, le connecteur détermine quelle base de données utiliser en localisant le pilote.

Il est indispensable d'indiquer une valeur pour que le connecteur fonctionne correctement.

Aucune valeur par défaut n'est fournie.

ReplaceAllStr

Indique si le connecteur remplace toutes les instances de chaque caractère identifié dans la propriété `ReplaceStrList` par la chaîne de substitution indiquée dans cette propriété. Le connecteur analyse la propriété ReplaceAllStr uniquement si le paramètre `ESC=[true|false]` de chaque propriété AppSpecificInfo d'attribut ne contient pas de valeur. En d'autres termes, si le paramètre `ESC` a été indiqué, sa valeur a priorité sur la valeur définie pour la propriété `ReplaceAllStr`. Pour que le connecteur utilise la valeur de la propriété `ReplaceAllStr`, vérifiez que le paramètre `ESC` n'a pas été indiqué.

La valeur par défaut de la propriété `ReplaceAllStr` est `false`.

Remarque : Le paramètre ESC et les propriétés ReplaceAllStr et ReplaceStrList assurent la prise en charge de la fonction du caractère d'échappement dans la base de données (par exemple, guillemets simples d'échappement). Dans la mesure où cette même fonction est également disponible dans les instructions préparées fournies par le pilote JDBC, ces propriétés seront déconseillées dans les éditions

ultérieures du connecteur. Le connecteur prend actuellement en charge l'utilisation des instructions préparées du pilote JDBC.

ReplaceStrList

Indique un ou plusieurs ensembles de substitution, chacun d'eux constitué d'un caractère à remplacer, d'un caractère de délimitation et d'une chaîne de substitution. Le connecteur effectue cette substitution sur une valeur d'attribut uniquement si une valeur est indiquée pour le paramètre `ESC=[true|false]` de la propriété AppSpecificInfo de l'attribut ou pour la propriété `ReplaceAllStr` du connecteur.

Remarque : Le paramètre ESC et les propriétés ReplaceAllStr et ReplaceStrList assurent la prise en charge de la fonction du caractère d'échappement dans la base de données (par exemple, guillemets simples d'échappement). Dans la mesure où cette même fonction est également disponible dans les instructions préparées fournies par le pilote JDBC, ces propriétés seront déconseillées dans les éditions ultérieures du connecteur. Le connecteur prend actuellement en charge l'utilisation des instructions préparées du pilote JDBC.

La syntaxe de cet attribut est la suivante :

single_char1,*substitution_str1*[:*single_char2*,*substitution_str2*[:...]]

où :

single_char	Caractère à remplacer.
substitution_str	Chaîne de substitution que le connecteur utilise pour remplacer le caractère.
,	Caractère de délimitation qui sépare le caractère à remplacer de la chaîne qui le remplace. Par défaut, le caractère de délimitation est une virgule (,). Vous pouvez configurer ce délimiteur en définissant le premier délimiteur dans la propriété `StrDelimiter`.
:	Délimiteur de fin qui sépare les ensembles de substitution (chacun d'eux est constitué du caractère à remplacer, d'un caractère de délimitation et d'une chaîne de substitution). Par défaut, le délimiteur de fin est un signe deux-points (:). Vous pouvez configurer ce délimiteur en définissant le second délimiteur dans la propriété `StrDelimiter`.

Par exemple, imaginons que vous vouliez remplacer un signe pourcentage (%) par deux signes pourcentage (%%), et un caret (^) par une barre oblique inverse et un caret (\^). Par défaut, la propriété StrDelimiter indique une virgule (,) comme délimiteur de caractères et un signe deux-points (:) comme délimiteur de fin. Si vous conservez ces délimiteurs par défaut, utilisez la chaîne suivante comme valeur de `ReplaceStrList` :

```
%,%%:^,\^
```

Remarque : Une restriction de l'outil de configuration du connecteur empêche d'entrer des guillemets simples. Par conséquent, vous devez représenter un guillemet simple par le caractère `Q` et deux guillemets simples par les caractères `DSQ`. Dans l'exemple précédent, si vous

voulez également remplacer un guillemet simple (') par deux guillemets simples (''), utilisez la notation suivante : `Q,DSQ:%,%%:^,\^`

RetryCountAndInterval

Indique le nombre de tentatives ainsi que l'intervalle en secondes que le connecteur doit utiliser lorsqu'il ne peut pas verrouiller des données pendant une mise à jour.

Pour effectuer une mise à jour, le connecteur verrouille les lignes relatives à la mise à jour et tente d'extraire les données courantes. Si le connecteur ne parvient pas à verrouiller les lignes, il tente à nouveau de verrouiller les données d'après le nombre et l'intervalle indiqués dans cette propriété de configuration. Le délai du connecteur expire s'il ne peut pas verrouiller les données conformément aux valeurs indiquées.

Indiquez la valeur sous cette forme : nombre, intervalle en secondes. Par exemple, une valeur de `3,20` indique trois tentatives dans un intervalle de 20 secondes.

La valeur par défaut est `3,20`.

ReturnDummyBOForSP

Cette propriété est utilisée pour renvoyer des paramètres de sortie même lorsque le jeu de résultats est vide.

Dans le cas de la propriété RetrieveSP, un jeu de résultats est renvoyé. Si le jeu de résultats est vide, aucun objet métier n'est créé, et les paramètres de sortie renvoyés par l'appel de procédure ne peuvent pas être récupérés. Toutefois, si la propriété ReturnDummyBOForSP a la valeur `true`, un objet métier fictif comprenant des valeurs issues des paramètres d'entrée/sortie indiqués dans les attributs correspondants sera renvoyé.

La valeur par défaut est ″false″.

SchemaName

Cette propriété limite la recherche pour les tables d'événements et d'archivage dans ce schéma spécifique. Si cette propriété n'est pas ajoutée ou qu'elle est vide, le connecteur recherchera tous les schémas auxquels l'utilisateur a accès. La propriété SchemaName est également utilisée lors de la création de requêtes d'accès aux tables d'événements et d'archivage.

La base de données Oracle assure la prise en charge des noms de schéma. Pour plus d'informations, voir la documentation du pilote JDBC.

Aucune valeur par défaut n'est indiquée.

SelectiveCommitForPoll

Indique lorsque des validations de base de données auront lieu. Si la propriété a la valeur `true`, une seule validation aura lieu, après le traitement d'un événement. Si la valeur affectée est `false`, le comportement standard est utilisé : les validations de la base de données se produiront à chaque étape du traitement des événements.

La valeur par défaut est ″false″.

SPBeforePollCall

Cette propriété nomme la procédure stockée qui est exécutée pour chaque appel d'interrogation. Si la propriété `SPBeforePollCall` a une valeur (le nom d'une

procédure stockée), à chaque début d'appel d'interrogation, le connecteur appelle la procédure stockée, en lui transmettant les valeurs des propriétés du connecteur `ConnectorID` et `PollQuantity`. La procédure mettra à jour le nombre de lignes dans la propriété `PollQuantity`, en affectant à la colonne d'ID connecteur la valeur `ConnectorID` où status=0 et connector-id correspond à `null`. Cela permet d'équilibrer la charge dans le connecteur.

Remarque : Dans le cas où un appel d'interrogation échoue prématurément (la base de données est inactive ou la connexion est interrompue), l'ID connecteur reste défini. Du fait de cet échec, des enregistrements peuvent être ignorés pendant l'interrogation. Il est par conséquent recommandé de réaffecter régulièrement à l'ID connecteur la valeur `null` pour tous les enregistrements contenus dans la table d'événements avec l'état 0.

StrDelimiter

Cette propriété indique les délimiteurs de caractères ou de fin à utiliser dans la propriété `ReplaceStrList`.

- Le délimiteur de caractères sépare le caractère à remplacer de la chaîne qui le remplace. Le délimiteur de caractères occupe la première position (à gauche) des valeurs de la propriété et a par défaut la valeur virgule (**,**).
- Le délimiteur de fin sépare les ensembles de substitution (chacun d'eux est constitué du caractère à remplacer, d'un caractère de délimitation et d'une chaîne de substitution). Le délimiteur de fin occupe la seconde position (à droite) des valeurs de la propriété et a par défaut la valeur deux-points (**:**).

Vous pouvez indiquer votre propre valeur pour l'un des délimiteurs. Dans le cas contraire, n'indiquez pas d'espace ou d'autres caractères entre eux.

La valeur par défaut est une virgule suivie immédiatement d'un signe deux-points (**,:**).

TimingStats

Cette propriété vous permet de prévoir chaque instruction du connecteur afin d'identifier les problèmes éventuels. Les paramètres disponibles sont les suivants :

`0` Aucune statistique sur les délais

`1` Heure affichée à l'entrée et à la sortie de l'opération d'instruction pour un objet métier hiérarchique entier.

`2` Heure affichée à l'entrée et à la sortie de l'opération d'instruction pour chaque objet métier dans un objet métier hiérarchique.

Les messages de délai sont des messages de fichier journal plutôt que des messages trace. Ils peuvent être activés ou désactivés, indépendamment des niveaux de trace.

La valeur par défaut est `0`.

UniqueIDTableName

Indique la table qui contient la dernière valeur utilisée pour créer un ID unique. Par défaut, la table contient une colonne (`id`). Vous pouvez personnaliser la table afin d'ajouter une colonne pour chaque attribut qui requiert la création d'un UID.

La valeur par défaut est `xworlds_uid`.

UseDefaults

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

Si la propriété UseDefaults a la valeur `false`, le connecteur vérifie si une valeur correcte est indiquée pour chaque attribut d'objet métier nécessaire. L'opération de création échoue si la valeur indiquée n'est pas correcte.

La valeur par défaut est `false`.

UseDefaultsForCreatingChildBOs

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

Si la propriété UseDefaultsForCreatingChildBOs a la valeur `false`, le connecteur vérifie si une valeur correcte est indiquée pour chaque attribut d'objet métier nécessaire. L'opération de création échoue si la valeur indiquée n'est pas correcte.

La valeur par défaut est ″false″.

UseDefaultsWhenPolling

Si la propriété UseDefaultsWhenPolling a la valeur `true`, les valeurs par défaut seront définies dans l'objet métier avant qu'il ne soit extrait de la base de données et transféré sur le serveur.

Si la propriété UseDefaultsWhenPolling a la valeur `false`, les valeurs par défaut ne seront pas définies dans l'objet métier avant qu'il ne soit extrait de la base de données et transféré sur le serveur.

La valeur par défaut est `true`.

Remarque : Cette propriété de configuration spécifique au connecteur remplace la propriété UseDefaultsForRetrieve.

Création d'instances multiples d'un 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 31.
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**. Le script de démarrage doit résider dans le répertoire d'exécution du connecteur :

ProductDir\connectors*connName*

où *connName* identifie le connecteur. Le nom du script de démarrage dépend de la plateforme du système d'exploitation, comme le montre le tableau 9.

Tableau 9. Script de démarrage pour un connecteur

Système d'exploitation	Script de démarrage
Systèmes UNIX	connector_manager_*connName*
Windows	start_*connName*.bat

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* [-c*configFile*]
  - Sur les systèmes UNIX :

    connector_manager_*connName* -start

  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 d'Adapter Monitor (produit WebSphere Business Integration Adapters uniquement), qui est 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 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.
  - Sur les systèmes UNIX, les connecteurs s'exécutent à l'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), qui est lancé lorsque vous démarrez System Manager :

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

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

Chapitre 3. Présentation des objets métier pour le connecteur

Ce chapitre décrit la manière dont le connecteur pour JDBC traite les objets métier et décrit les suppositions établies par le connecteur lors de l'extraction et de la modification de données. Il contient les sections suivantes :

- «Conventions d'affectation de noms pour les objets métier et les attributs»
- «Structure de l'objet métier»
- «Traitement des instructions d'objet métier» à la page 40
- «Propriétés des attributs d'objet métier» à la page 57
- «Informations spécifiques à l'application d'objets métier» à la page 59

Vous pouvez utiliser ces informations pour modifier les objets métier existants ou en implémenter des nouveaux. Pour plus d'informations sur l'utilitaire qui automatise la création de fichiers de définitions d'objets métier WebSphere Business Integration Adapter à partir de tables de base de données, voir Chapitre 4, «Création de définitions d'objets métier à l'aide d'IBM ODA for Manugistics», à la page 71.

Le connecteur établit des suppositions concernant la structure des objets métier pris en charge, les relations entre les objets métier parent et enfant, le format des informations spécifiques à l'application et la représentation de l'objet métier dans la base de données. Par conséquent, lorsque vous créez ou modifiez un objet métier qui sera traité par le connecteur, ces modifications être conformes aux règles que le connecteur est supposé suivre. Dans le cas contraire, le connecteur ne peut pas traiter correctement les nouveaux objets métier ou les objets métier modifiés.

Conventions d'affectation de noms pour les objets métier et les attributs

Le nom d'un objet métier utilisé par le connecteur peut uniquement comporter des caractères alphanumériques et le caractère de soulignement. Les noms d'attributs d'objets métier peuvent également comporter uniquement des caractères alphanumériques et le caractère de soulignement.

Structure de l'objet métier

Dans la plupart des cas, le connecteur suppose que chaque objet métier est représenté par une table de base de données ou une vue et que chaque **attribut simple** (c'est-à-dire un attribut qui représente une valeur unique de type `String`, `Integer` ou `Date`) dans l'objet est représenté par une colonne dans cette table ou vue. De cette façon, les attributs à l'intérieur d'un même objet métier ne peuvent pas être stockés dans différentes tables de base de données. Toutefois, les situations suivantes sont envisageables :

- La table de base de données peut avoir davantage de colonnes que l'objet métier correspondant ne dispose d'attributs simples (en d'autres termes, certaines colonnes de la base de données ne sont pas représentées dans l'objet métier). Par conséquent, incluez à votre conception uniquement les colonnes nécessaires au traitement de l'objet métier.
- L'objet métier peut avoir davantage d'attributs simples que la table de base de données correspondante ne dispose de colonnes (en d'autres termes, certains attributs de l'objet métier ne sont pas représentés dans la base de données). Les

attributs n'ayant aucune représentation dans la base de données ne disposent d'aucune information spécifique à l'application, ont une valeur par défaut ou indiquent des procédures stockées.

- L'objet métier peut représenter une vue qui concerne plusieurs tables de base de données. Le connecteur peut utiliser cet objet métier lors du traitement des événements de création, d'extraction, de mise à jour et de suppression déclenchés dans l'application. Toutefois, lors du traitement des requêtes d'objets métier, le connecteur peut utiliser cet objet métier uniquement pour les requêtes d'extraction.
- L'objet métier peut représenter un objet encapsuleur utilisé en tant que conteneur pour les objets métier non associés. L'objet encapsuleur n'est pas représenté par une table de base de données ou une vue. Les objets encapsuleur ne peuvent pas être utilisés en tant qu'enfants d'autres objets.

Remarque : Si un objet métier est basé sur une procédure stockée, chaque attribut simple (autre que les attributs de procédures stockées spécifiques) peuvent avoir des informations spécifiques à l'application ou non. Pour plus d'informations, voir «Procédures stockées» à la page 49.

Les objets métier WebSphere Business Integration Adapter peuvent être plats ou hiérarchiques. Tous les attributs d'un objet métier **plat** sont simples et représentent une valeur unique. Le terme d'objet métier **hiérarchique** désigne un objet métier complet, incluant tous les objets métier enfant qu'il contient à n'importe quel niveau. Le terme d'objet métier **individuel** désigne un objet métier unique, indépendant des objets métier qu'il peut contenir ou qui le contient. Le terme d'objet métier **de niveau supérieur** désigne l'objet métier individuel en haut de la hiérarchie qui ne possède pas lui-même d'objet métier parent.

Un objet métier hiérarchique dispose d'attributs qui représentent un objet métier enfant, un tableau d'objets métier enfant ou une combinaison des deux. A son tour, chaque objet métier enfant peut contenir un objet métier enfant ou un tableau d'objets métier enfant et ainsi de suite. Le terme de **relation à cardinalité simple** est utilisé lorsqu'un attribut contenu dans un objet métier parent représente un objet métier enfant unique. Dans ce cas, l'attribut est de même type que l'objet métier enfant.

Le terme de **relation à cardinalité multiple** est utilisé lorsqu'un attribut contenu dans l'objet métier parent représente un tableau d'objets métier enfant. Dans ce cas, l'attribut est un tableau de même type que les objets métier enfant.

Le connecteur prend en charge les relations suivantes entre les objets métier :

- «Relations à cardinalité simple» à la page 37
- «Relations à cardinalité simple et données sans droits de propriété» à la page 37
- «Relations à cardinalité multiple» à la page 38
- «Relations à cardinalité simple qui stockent la relation dans l'enfant» à la page 39
- «Objets encapsuleur» à la page 40

Dans chaque type de cardinalité, la relation entre les objets métier parent et enfant est décrite par les informations d'application de l'attribut clé de l'objet métier hébergeant la relation. Pour plus d'informations sur ces informations spécifiques à l'application, voir «FK=[fk_object_name.]fk_attribute_name» à la page 62.

Relations à cardinalité simple

En règle générale, un objet métier contenant un objet métier enfant de type cardinalité simple possède au moins deux attributs représentant la relation. L'un des attributs est de même type que l'enfant. L'autre attribut est un attribut simple qui contient la clé primaire de l'enfant considérée comme la clé étrangère dans le parent. Le parent dispose d'autant d'attributs de clé étrangère que l'enfant dispose d'attributs de clé primaire.

Dans la mesure où les clés étrangères qui établissent la relation sont stockées dans le parent, chaque parent ne peut contenir qu'un seul enfant de type cardinalité simple d'un type donné.

La figure 2 illustre une relation spécifique à cardinalité simple. Dans cet exemple, `fk1` représente l'attribut simple qui contient la clé primaire de l'enfant et `child[1]` désigne l'attribut qui représente l'objet métier enfant.

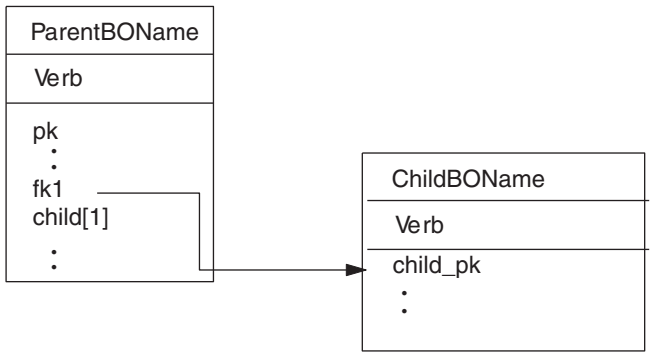


Figure 2. Relation spécifique à cardinalité simple

Relations à cardinalité simple et données sans droits de propriété

En règle générale, chaque objet métier parent est **propriétaire** des données contenues dans l'objet métier enfant qu'il contient. Par exemple, si chaque objet métier Customer contient un objet métier Address unique, une nouvelle ligne est alors insérée dans les tables des clients et des adresses lors de la création d'un client. La nouvelle adresse est propre au nouveau client. De la même manière, lors de la suppression d'un client de la table des clients, l'adresse du client est également supprimée de la table des adresses.

Toutefois, il existe certains cas dans lesquels plusieurs objets métier hiérarchiques possèdent les mêmes données dont aucun d'eux n'est propriétaire. Par exemple, supposons qu'un objet métier Address possède un attribut `StateProvince[1]` représentant la `table de recherche StateProvince avec le type de cardinalité simple`. Dans la mesure où la table de `recherche` est rarement mise à jour et qu'elle est gérée indépendamment des données d'adresse, la création ou la modification des données d'adresse n'affecte pas les données contenues dans la table de recherche. Le connecteur recherche un nom de région ou de département existant ou échoue. Il n'ajoute ni ne modifie aucune valeur dans la table de recherche.

Lorsque plusieurs objets métier contiennent le même objet métier enfant de type cardinalité simple, l'attribut de clé étrangère contenu dans chaque objet métier parent doit spécifier la relation `NO_OWNERSHIP`. Lorsqu'un courtier d'intégration envoie un objet métier hiérarchique au connecteur avec une requête de création, de

suppression ou de mise à jour, le connecteur ignore les enfants de type cardinalité simple contenus sans droits de propriété. Le connecteur réalise uniquement l'extraction de ces objets métier. Si le connecteur ne parvient pas à extraire cet objet métier de type cardinalité simple, il renvoie un message d'erreur et met fin au traitement.

Pour plus d'informations sur la manière de spécifier la relation sans droits de propriété, voir «Attributs représentant un objet métier enfant de type cardinalité simple» à la page 68. Pour plus d'informations sur la spécification de relations de clé étrangère, voir «Spécification de la clé étrangère d'un attribut» à la page 64.

Données dénormalisées et données sans droits de propriété

Outre la simplification de l'utilisation des tables de recherche statiques, le confinement sans droits de propriété présente un autre intérêt : la synchronisation des données normalisées et dénormalisées.

Synchronisation à partir de données normalisées vers des données dénormalisées : Grâce à la relation `NO_OWNERSHIP`, vous pouvez créer ou modifier les données lorsque vous effectuez une synchronisation depuis une application normalisée vers une application dénormalisée. Par exemple, supposons que votre application source normalisée stocke des données dans deux tables, A et B. Supposons également que votre application cible dénormalisée stocke toutes les données dans une seule table de telle manière que chaque entité de la table A stocke de manière redondante les données de la table B.

Dans cet exemple, pour synchroniser une modification apportée aux données de la table B depuis votre application source vers votre application cible, vous devez déclencher un événement de table A chaque fois que les données de la table B sont modifiées. De plus, dans la mesure où les données de la table B sont stockées de manière redondante dans la table A, vous devez envoyer un objet métier pour chaque ligne dans la table A contenant les données modifiées de la table B.

Synchronisation à partir de données dénormalisées vers des données normalisées : Lors de la synchronisation de données depuis une application source dénormalisée vers une application cible normalisée, le connecteur ne crée, ne supprime ni ne met à jour aucune donnée contenue sans droits de propriété dans l'application normalisée.

Lors de la synchronisation de données vers une application normalisée, le connecteur ignore tous les enfants de type cardinalité simple contenus sans droits de propriété. Pour pouvoir créer, supprimer ou modifier ces données enfant, vous devez traiter les données manuellement.

Relations à cardinalité multiple

En règle générale, un objet métier contenant un tableau d'objets métier enfant possède un seul attribut représentant la relation. L'attribut est un tableau de même type que les objets métier enfant. Pour permettre à un parent de contenir plusieurs enfants, les clés étrangères qui établissent la relation sont stockées dans chaque enfant.

Par conséquent, chaque enfant possède au moins un attribut simple contenant la clé primaire du parent considérée comme une clé étrangère. L'enfant dispose d'autant d'attributs de clé étrangère que le parent dispose d'attributs de clé primaire.

Dans la mesure où les clés étrangères qui établissent la relation sont stockées dans l'enfant, chaque parent peut n'avoir aucun enfant ou en avoir plusieurs.

La figure 3 illustre une relation à cardinalité multiple. Dans cet exemple, `parentId` représente l'attribut simple qui contient la clé primaire du parent et `child[n]` désigne l'attribut qui représente le tableau des objets métier enfant.

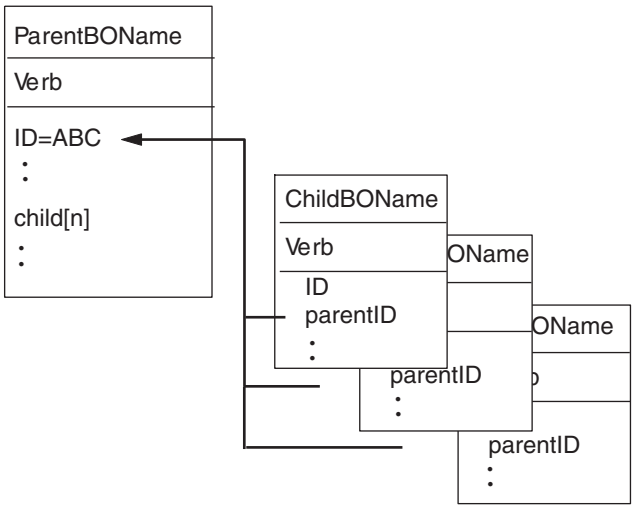


Figure 3. Relation d'objet métier à cardinalité multiple

Relations à cardinalité simple qui stockent la relation dans l'enfant

Certaines applications stockent une entité enfant unique de sorte que la relation est stockée dans l'enfant plutôt que dans le parent. En d'autres termes, l'enfant contient une clé étrangère dont la valeur est identique à celle stockée dans la clé primaire du parent.

La figure 4 illustre ce type de relation spécifique à cardinalité simple.

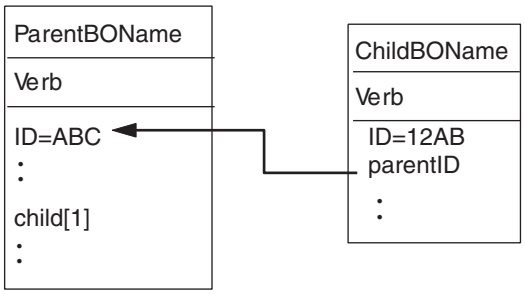


Figure 4. Objet métier de type cardinalité avec une relation stockée dans l'enfant

Les applications utilisent ce type de relation à cardinalité simple lorsque les données enfant n'existent pas indépendamment de leurs parents et qu'elles sont accessibles uniquement par l'intermédiaires des parents. Ces données enfant n'appartiennent jamais à plus d'un parent et requièrent l'existence du parent et de sa valeur de clé primaire avant que l'enfant et sa valeur de clé étrangère ne puissent être créés.

Pour concilier de telles applications, le connecteur prend également en charge les objets métier hiérarchiques qui contiennent un enfant de type cardinalité simple mais qui stockent la relation dans l'enfant plutôt que dans le parent.

Pour spécifier qu'un objet métier parent contient un enfant de type cardinalité simple de cette manière, n'incluez pas le paramètre `CONTAINMENT` lorsque vous spécifiez les informations d'application de l'attribut contenant l'enfant. Pour plus d'informations, voir «Attributs représentant un objet métier enfant de type cardinalité simple» à la page 68.

Objets encapsuleur

Un **objet encapsuleur** est un objet métier de niveau supérieur qui ne correspond à aucune table de base de données ni à aucune vue. L'objet encapsuleur est dénoté par la propriété `WRAPPER` de l'objet métier de niveau supérieur dont la valeur est `true`. L'objet encapsuleur est un parent fictif utilisé en tant que conteneur pour les enfants non associés. Lors du traitement d'un objet encapsuleur, le connecteur ignore l'objet métier de niveau supérieur et traite uniquement les enfants. L'objet encapsuleur peut contenir les entités de cardinalité N et/ou N-1.

Une entité de cardinalité N doit au minimum avoir un attribut unique signalé en tant que clé primaire et au minimum un attribut signalé en tant que clé étrangère. Cette clé étrangère sera ensuite ajoutée en tant que clé primaire à l'objet encapsuleur. La clé étrangère de l'entité fera référence à la clé primaire de l'objet encapsuleur tout juste ajoutée.

Dans le cas d'une entité de cardinalité N-1, la clé primaire doit être signalée à la fois en tant que clé primaire et clé étrangère, référençant la clé primaire dans l'encapsuleur, identique à la clé primaire de l'entité N-1.

Traitement des instructions d'objet métier

Cette section décrit les aspects suivants du traitement des instructions d'un objet métier :

- La section «Détermination des instructions» explique comment le connecteur détermine l'instruction à utiliser pour chaque objet métier source individuel.
- La section «Images postérieures et deltas» à la page 41 définit les termes et explique comment le connecteur gère les images postérieures.
- La section «Traitement des instructions» à la page 42 explique les étapes réalisées par le connecteur lors de la création, l'extraction, la mise à jour ou la suppression d'un objet métier.
- La section «Instructions SQL» à la page 49 explique comment le connecteur utilise les instructions SQL simples pour les opérations de sélection, de mise à jour, d'extraction ou de suppression.
- La section «Procédures stockées» à la page 49 explique comment le connecteur utilise les procédures stockées.
- La section «Validation et annulation de transaction» à la page 57 explique brièvement comment le connecteur utilise les blocs de transaction.

Détermination des instructions

Un objet métier de niveau supérieur et chacun de ses objets métier enfant individuels peuvent contenir leurs propres instructions. Par conséquent, un courtier d'intégration peut transmettre au connecteur un objet métier doté d'instructions différentes pour les objets métier parent et enfant. Lorsque cela se

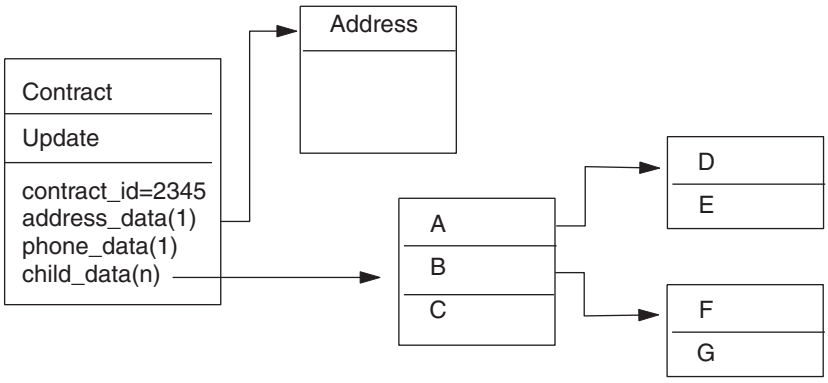
produit, le connecteur utilise l'instruction de l'objet métier parent de niveau supérieur pour déterminer le mode de traitement de l'ensemble de l'objet métier. Pour plus d'informations, voir «Traitement des instructions» à la page 42.

Images postérieures et deltas

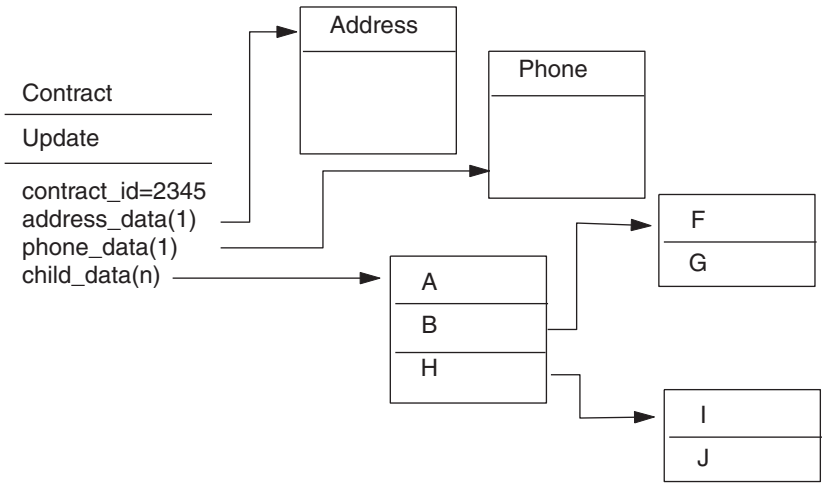
Une **image postérieure** représente l'état d'un objet métier après que toutes les modifications ont été apportées. Un **delta** est un objet métier utilisé dans une opération de mise à jour qui contient uniquement les valeurs de clés et les données à modifier. Dans la mesure où le connecteur prend uniquement en charge les images postérieures, dès réception d'un objet métier pour sa mise à jour, le connecteur suppose que l'objet métier représente l'état souhaité des données après mise à jour.

Par conséquent, lorsqu'un courtier d'intégration envoie au connecteur un objet métier avec l'instruction Update, le connecteur modifie la représentation actuelle de l'objet métier dans la base de données de sorte qu'elle corresponde exactement à l'objet métier source. Pour ce faire, le connecteur modifie les valeurs d'attribut simple et ajoute ou supprime des objets métier enfant.

Par exemple, supposons que l'état actuel de l'objet métier Contract 2345 dans la base de données soit le suivant :



Supposons également que le courtier d'intégration transmette au connecteur l'objet métier suivant :



Pour procéder à la mise à jour, le connecteur applique les modifications suivantes à la base de données :

- mise à jour des attributs simples dans les objets métier Contract et Address de niveau supérieur ;
- création de l'objet métier Phone ;
- mise à jour des attributs simples dans les objets métier enfant A, B, F et G ;
- suppression des objets métier enfant C, D et E ;
- création des objets métier enfant H, I et J.

Dans la mesure où le connecteur suppose que chaque objet métier provenant du courtier d'intégration représente une image postérieure, il est important de veiller à ce que chaque objet métier envoyé à ce connecteur pour la mise à jour contienne des objets métier enfant existants et valides. Même si aucun des attributs simples d'un objet métier n'a été modifié, l'objet métier enfant doit être inclus dans l'objet métier source.

Vous avez toutefois la possibilité d'empêcher certains connecteurs de supprimer les objets métier enfant manquants au cours d'une opération de mise à jour. Vous pouvez utiliser les informations d'application relatives à l'attribut représentant l'enfant ou le tableau d'enfants pour demander au connecteur de conserver les objets métier enfant qui ne sont pas inclus dans l'objet métier source. Pour ce faire, attribuez au paramètre `KEEP_RELATIONSHIP` la valeur `true`. Pour plus d'informations, voir «Spécification de la clé étrangère d'un attribut» à la page 64.

Traitement des instructions

Cette section présente les étapes réalisées par le connecteur lors de la création, l'extraction, la mise à jour ou de la suppression d'un objet métier envoyé par un courtier d'intégration. Le connecteur traite les objets métier hiérarchiques de manière récursive. En d'autres termes, il procède aux mêmes étapes pour chaque objet métier enfant jusqu'à ce qu'il ait traité tous les objets métier individuels.

Remarque : Un objet métier de niveau supérieur correspondant à un encapsuleur prend en charge les instructions de création, d'extraction, de mise à jour et de suppression. La seule différence avec le traitement d'un objet encapsuleur réside dans le fait que seuls les objets contenus dans l'objet encapsuleur sont traités et non l'objet lui-même.

Comparaison des objets métier

A différents points du traitement décrit ci-après, le connecteur compare deux objets métier pour voir s'ils sont identiques. Par exemple, au cours d'une opération de mise à jour, le connecteur détermine si un objet métier spécifique existe dans un tableau d'objets métier. Pour effectuer cette vérification, le connecteur compare l'objet métier concerné avec chaque objet métier du tableau. Pour que deux objets métier soient considérés comme identiques, les deux conditions suivantes doivent être remplies :

- Les objets métier comparés doivent être de type identique. Par exemple, un objet métier Customer n'est jamais considéré comme identique à un objet métier Contact même s'ils ont exactement les mêmes attributs.
- Tous les attributs de clé correspondants dans les deux objets métier doivent contenir des valeurs identiques. Si un attribut de clé a la valeur `CxIgnore` dans les deux objets métier, le connecteur les considère comme identiques. Toutefois, si un attribut de clé a la valeur `CxIgnore` dans l'un des objets métier et non dans l'autre, les objets métier ne sont pas identiques.

Opérations de création

Lors de la création d'un objet métier, le connecteur renvoie l'état `VALCHANGE` si l'opération a réussi (que l'opération ait entraîné ou non des modifications à l'objet métier) ou `FAIL` si l'opération a échoué.

Lors de la création d'un objet métier hiérarchique, le connecteur procède aux étapes suivantes :

1. Il insère de manière récursive chaque objet métier enfant de type cardinalité simple contenu avec des droits de propriété dans la base de données. En d'autres termes, le connecteur crée l'enfant et tous les objets métier enfant que l'enfant et ses enfants contiennent.

   Si la définition d'objet métier spécifie qu'un attribut représente un objet métier enfant de type cardinalité simple et que cet attribut est vide, le connecteur ignore l'attribut. Toutefois, si la définition d'objet métier requiert que l'attribut représente un enfant et que ce n'est pas le cas, le connecteur renvoie un message d'erreur et met fin au traitement.
2. Il traite chaque objet métier enfant de type cardinalité simple contenu avec des droits de propriété comme suit :
   a. Il tente de manière récursive d'extraire l'enfant de la base de données à l'aide des valeurs de clé transmises par le courtier d'intégration.
   b. Si l'extraction échoue, indiquant ainsi que l'enfant n'existe pas actuellement dans la base de données, le connecteur renvoie un message d'erreur et met fin au traitement. Si l'extraction réussit, le connecteur met à jour l'objet métier enfant de manière récursive.

   Remarque : Pour permettre à cette approche de fonctionner correctement lorsque l'objet métier enfant existe déjà dans la base de données de l'application, veillez à ce que les renvois aux attributs de clé primaire contenus dans les objets métier enfant soient corrects dans les opérations de création. Si l'objet métier enfant n'existe pas déjà dans la base de données de l'application, affectez aux attributs de clé primaire la valeur `CxBlank`.
3. Il insère l'objet métier de niveau supérieur dans la base de données comme suit :
   a. Il affecte à chaque valeur de clé étrangère les valeurs de clé primaire de l'objet métier enfant correspondant représenté avec le type cardinalité simple. Dans la mesure où les valeurs des objets métier enfant peuvent être définies par des compteurs ou des séquences de base de données ou par la base de données elle-même lors de la création de l'enfant, cette étape garantit la validité des valeurs de clé étrangère dans le parent avant que ce dernier ne soit inséré dans la base de données par le connecteur.
   b. Il génère une nouvelle valeur d'ID unique pour chaque attribut défini automatiquement par la base de données. Le nom du compteur ou de la séquence de base de données est stocké dans les informations spécifiques à l'application de l'attribut. Si un attribut possède un compteur ou une séquence de base de données qui lui est associé, la valeur générée par le connecteur remplace toute valeur transmise par le courtier d'intégration. Pour plus d'informations sur la spécification d'un compteur ou d'une séquence de base de données, voir `UID=AUTO` dans «Informations spécifiques à l'application pour les attributs simples» à la page 62.
   c. Il copie la valeur d'un attribut sur la valeur d'un autre attribut comme indiqué par le paramètre `CA` (CopyAttribute) des informations d'application

de l'attribut. Pour plus d'informations sur l'utilisation du paramètre CA, voir CA=set_attr_name dans «Informations spécifiques à l'application pour les attributs simples» à la page 62.

d. Il insère l'objet métier de niveau supérieur dans la base de données.

Remarque : Un objet métier de niveau supérieur correspondant à un encapsuleur ne sera pas inséré dans la base de données.

4. Il traite chaque objet métier enfant de type cardinalité simple qui stocke la relation parent-enfant dans l'enfant, comme suit :
   a. Il définit les valeurs de clé étrangère dans l'enfant pour faire référence à la valeur contenue dans les attributs de clé primaire correspondants dans le parent. Dans la mesure où les valeurs de clé primaire du parent ont été générées au cours de la création du parent, la validité des valeurs de clé étrangère dans chaque enfant est ainsi garantie avant que l'enfant ne soit inséré dans la base de données.
   b. Il insère l'enfant dans la base de données.
5. Il traite chaque objet métier enfant de type cardinalité multiple comme suit :
   a. Il définit les valeurs de clé étrangère dans chaque enfant pour faire référence à la valeur contenue dans les attributs de clé primaire correspondant dans le parent. Dans la mesure où les valeurs de clé primaire du parent ont été générées au cours de la création du parent, la validité des valeurs de clé étrangère dans chaque enfant est ainsi garantie avant que l'enfant ne soit inséré dans la base de données.
   b. Il insère chaque objet métier enfant de type cardinalité multiple dans la base de données.

Opérations d'extraction

Lors de l'extraction d'un objet métier hiérarchique, le connecteur procède aux étapes suivantes :

1. Il supprime tous les objets métier enfant de l'objet métier de niveau supérieur reçu du courtier d'intégration.
2. Il extrait l'objet métier de niveau supérieur de la base de données.
   - Si l'extraction renvoie une ligne, le connecteur poursuit le traitement.
   - Si l'extraction ne renvoie aucune ligne, indiquant que l'objet métier de niveau supérieur n'existe pas dans la base de données, le connecteur renvoie la valeur BO_DOES_NOT_EXIST.
   - Si l'extraction renvoie plusieurs lignes, le connecteur renvoie la valeur FAIL.

   Remarques :
   a. Un objet métier peut avoir des attributs qui ne correspondent à aucune colonne de la base de données, tels que les attributs d'espace réservé. Lors de l'extraction, le connecteur ne modifie pas ces attributs dans l'objet métier de niveau supérieur : ils conservent les valeurs reçues du courtier d'intégration. Dans les objets métier enfant, le connecteur affecte à ces attributs leurs valeurs par défaut lors de l'extraction.
   b. Un objet métier de niveau supérieur correspondant à un encapsuleur doit contenir toutes les valeurs d'attribut des objets situés au niveau inférieur immédiat de l'objet encapsuleur car ces valeurs seront nécessaires pour extraire les objets, notamment les clés et les attributs d'espace réservé. L'ensemble des clés et attributs d'espace réservé doit être renseigné dans l'objet encapsuleur. Les attributs simples de l'objet encapsuleur qui seront utilisés en tant que clés étrangères dans les objets d'un niveau inférieur à l'encapsuleur devront être signalés en tant que clés dans l'objet encapsuleur.

3. Il extrait de manière récursive tous les objets métier enfant de type cardinalité multiple.

   Remarque : Le connecteur n'impose pas le caractère unique des valeurs lors du remplissage d'un tableau d'objets métier. Il en va de la responsabilité de la base de données d'assurer cette unicité. Si la base de données renvoie des objets métier enfant en double, le connecteur renvoie à son tour des enfants en double.

4. Il extrait de manière récursive chaque enfant de type cardinalité simple, que l'objet métier enfant soit contenu avec ou sans droits de propriété.

   Remarque : Tous les objets métier enfant de type cardinalité simple sont traités en fonction des occurrences dans l'objet métier et avant le traitement de l'objet métier parent. La propriété ou la non-propriété des objets enfant ne détermine pas la séquence de traitement mais en détermine le type.

Opérations d'extraction par contenu

Une instruction RetrieveByContent est applicable uniquement pour l'objet métier de niveau supérieur car le connecteur réalise une extraction en fonction des attributs contenus uniquement dans l'objet métier de niveau supérieur.

Si un objet métier de niveau supérieur utilise l'instruction RetrieveByContent, tous les attributs (y compris les attributs non clés) dont la valeur n'est pas null sont utilisés comme critères d'extraction.

Si plusieurs lignes sont renvoyées, le connecteur utilise la première ligne en tant que ligne de résultat et renvoie la valeur `MULTIPLE_HITS`.

Remarque : Une instruction RetrieveByContent n'est pas applicable pour un objet métier de niveau supérieur correspondant à un encapsuleur.

Opérations de mise à jour

Lors de la mise à jour d'un objet métier, le connecteur renvoie l'état `VALCHANGE` si l'opération a réussi (que l'opération ait entraîné ou non des modifications à l'objet métier) ou `FAIL` si l'opération a échoué. Lors de l'utilisation d'une base de données Oracle, le connecteur verrouille les données lors de leur extraction afin de garantir leur intégrité.

Lors de la mise à jour d'un objet métier hiérarchique, le connecteur procède aux étapes suivantes :

1. Il utilise les valeurs de clé primaire de l'objet métier source pour extraire l'entité correspondante de la base de données. L'objet métier extrait est une représentation exacte de l'état en cours des données dans la base de données.
   - Si l'extraction échoue, indiquant que l'objet métier de niveau supérieur n'existe pas dans la base de données, le connecteur renvoie la valeur `BO_DOES_NOT_EXIST` et la mise à jour échoue.

     Remarque : Un objet métier de niveau supérieur correspondant à un encapsuleur ne doit pas exister dans la base de données. Toutefois, il doit contenir toutes les valeurs d'attribut des objets situés au niveau inférieur immédiat de l'objet encapsuleur car ces valeurs seront nécessaires pour extraire les objets, notamment les clés et les attributs d'espace réservé. L'ensemble des clés et attributs d'espace réservé doit être renseigné dans

l'objet encapsuleur. Les attributs simples de l'objet encapsuleur qui seront utilisés en tant que clés étrangères dans les objets d'un niveau inférieur à l'encapsuleur devront être signalés en tant que clés dans l'objet encapsuleur.

- Si l'extraction réussit, le connecteur compare l'objet métier extrait avec l'objet métier source pour déterminer les objets métier enfant qui requièrent des modifications dans la base de données. Le connecteur ne doit pas, toutefois, comparer les valeurs contenues dans les attributs simples de l'objet métier source avec celles contenues dans l'objet métier extrait. Le connecteur met à jour la valeur de tous les attributs simples non clés.

  Si tous les attributs simples contenus dans l'objet métier de niveau supérieur représentent des clés, le connecteur ne peut générer aucune requête de mise à jour pour l'objet métier de niveau supérieur. Dans ce cas, le connecteur consigne un message d'avertissement et procède à l'étape 2.

2. Il met à jour de manière récursive tous les enfants de type cardinalité simple de l'objet métier de niveau supérieur.

   Si la définition d'objet métier nécessite qu'un attribut représente un objet métier enfant, l'enfant doit exister à la fois dans l'objet métier source et dans l'objet métier extrait. Dans le cas contraire, la mise à jour échoue et le connecteur renvoie un message d'erreur.

   Le connecteur gère les enfants de type cardinalité simple contenus avec des droits de propriété de l'une des manières suivantes :

   - Si l'enfant est présent dans les objets métier source et extrait, le connecteur supprime l'enfant existant dans la base de données et en crée un nouveau au lieu de le mettre à jour.
   - Si l'enfant est présent dans l'objet métier source et non dans l'objet métier extrait, le connecteur le crée de manière récursive dans la base de données.
   - Si l'enfant est présent dans l'objet métier extrait et non dans l'objet métier source, le connecteur le supprime de manière récursive de la base de données. Le type de suppression, physique ou logique, dépend de la valeur de la propriété `ChildUpdatePhyDelete` de l'enfant.

   Pour les enfants de type cardinalité simple contenus sans droits de propriété, le connecteur tente d'extraire chaque enfant de la base de données présent dans l'objet métier source. S'il parvient à le faire, le connecteur alimente l'objet métier enfant sans le mettre à jour dans la mesure où les enfants de type cardinalité simple contenus sans droits de propriété ne peuvent être modifiés par le connecteur.

3. Pour les objets métier enfant de type cardinalité simple qui stockent la relation dans le parent, le connecteur affecte à chaque valeur de clé étrangère dans le parent la valeur de la clé primaire contenue dans l'objet métier enfant de type cardinalité simple correspondant. Cette étape est nécessaire dans la mesure où des enfants de type cardinalité simple peuvent avoir été ajoutés à la base de données au cours des étapes précédentes, entraînant ainsi la création d'ID uniques supplémentaires.
4. Il met à jour tous les attributs simples de l'objet métier extrait à l'exception de ceux dont l'attribut correspondant dans l'objet métier source contient la valeur `CxIgnore`.

   Dans la mesure où l'objet métier mis à jour doit être unique, le connecteur vérifie qu'une seule ligne est traitée en retour. Il renvoie un message d'erreur si plusieurs lignes sont traitées.
5. Il affecte à toutes les valeurs de clé étrangère dans chaque enfant qui stocke la relation parent-enfant dans l'enfant (de type cardinalité multiple et simple) la valeur de clé primaire de l'objet métier parent correspondant.

Lorsqu'InterChange Server est utilisé en tant que courtier d'intégration, ces valeurs sont généralement mises en référence croisée au cours du mappage des données. Toutefois, cette étape est importante pour garantir la validité des valeurs de clé étrangère des nouveaux enfants qui stockent la relation dans l'enfant avant que le connecteur mette à jour ces enfants.

6. Il traite chaque enfant de type cardinalité multiple de l'objet métier extrait de l'une des manières suivantes :
   - Si l'enfant existe dans les tableaux des objets métier source et extrait, le connecteur le met à jour de manière récursive dans la base de données.
   - Si l'enfant existe dans le tableau de l'objet métier source et non dans le tableau de l'objet métier extrait, le connecteur le crée de manière récursive dans la base de données.
   - Si l'enfant existe dans le tableau de l'objet métier extrait et non dans le tableau de l'objet métier source, le connecteur le supprime de manière récursive de la base de données à moins que la valeur `KEEP_RELATIONSHIP` des informations d'application de l'attribut représentant l'enfant dans le parent soit `true`. Dans ce cas, le connecteur ne supprime pas l'enfant de la base de données. Pour plus d'informations, voir «Spécification de la clé étrangère d'un attribut» à la page 64. Le type de suppression, physique ou logique, dépend de la valeur de la propriété `ChildUpdatePhyDelete` de l'enfant.

Remarque : Le courtier d'intégration doit veiller à ce que les objets métier contenus avec une cardinalité multiple dans l'objet métier source soient uniques (en d'autres termes, un tableau ne doit pas contenir plusieurs copies d'un même objet métier). Si le connecteur extrait un objet métier en double dans un tableau source, il traite l'objet métier deux fois, ce qui peut entraîner des résultats incertains.

Opérations de mise à jour delta

Le traitement de l'instruction DeltaUpdate est différent du traitement de l'instruction Update sur les points suivants :

1. Avec une instruction DeltaUpdate, aucune extraction n'est réalisée avant la mise à jour, contrairement au traitement de l'instruction Update.
2. Aucune comparaison n'est effectuée entre l'objet métier entrant et l'objet métier contenu dans la base de données.
3. Tous les enfants sont traités en fonction de l'instruction définie dans chaque objet enfant. Si aucune instruction n'est définie pour l'un des enfants, le connecteur renvoie un message d'erreur.

Lors de la mise à jour delta d'un objet métier, le connecteur renvoie l'état `VALCHANGE` si l'opération a réussi (que l'opération ait entraîné ou non des modifications à l'objet métier) ou `FAIL` si l'opération a échoué.

Lors de la mise à jour delta d'un objet métier hiérarchique, le connecteur procède aux étapes suivantes :

1. Il traite de manière récursive tous les enfants de type cardinalité simple de l'objet parent. Si un enfant est signalé par la valeur IsRequired dans la spécification de l'objet métier, il doit être présent dans l'objet des communications entrantes. Dans le cas contraire, la mise à jour delta échoue et le connecteur renvoie un message d'erreur.
2. Il affecte à toutes les valeurs de clé étrangère dans le parent qui fait référence aux attributs contenus dans les enfants de type cardinalité simple les valeurs des enfants correspondants. Cette étape est nécessaire dans la mesure où des

enfants de type cardinalité simple peuvent avoir été ajoutés à la base de données au cours des étapes précédentes, entraînant ainsi la création de valeurs de séquence supplémentaires.

3. Il met à jour l'objet courant en cours de traitement via une instruction SQL `UPDATE` ou une procédure stockée. Tous les attributs simples de l'objet métier individuel sont mis à jour, à l'exception des attributs ayant la valeur IsIgnore dans l'objet métier des communications entrantes. Le connecteur ne compare pas l'objet des communications entrantes avec l'objet courant en fonction des attributs pour déterminer les attributs devant être ajoutés à l'instruction de mise à jour : ils sont tous mis à jour. Dans la mesure où l'objet mis à jour doit être unique, le connecteur vérifie qu'une seule ligne est traitée en retour. Un message d'erreur est renvoyé si plusieurs lignes sont traitées.
4. Il affecte à toutes les valeurs de clé étrangère dans tous les enfants de cardinalité N de l'objet courant qui fait référence aux attributs parent les valeurs parent correspondantes. En règle générale, ces valeurs ont déjà été renvoyées au cours du mappage des données. Cela peut toutefois ne pas être le cas pour les nouveaux enfants dans les conteneurs de cardinalité N. La validité de toutes les valeurs de clé étrangère dans tous les enfants de cardinalité N est ainsi garantie avant la mise à jour de ces enfants.
5. Il met à jour tous les conteneurs de cardinalité N de l'objet courant.

Lors du traitement des objets enfant, chaque instruction d'enfant est prise en compte et l'opération appropriée est réalisée. Les instructions autorisées sur un enfant dans une instruction DeltaUpdate sont les suivantes : Create, Delete et DeltaUpdate.

- Si une instruction Create est trouvée dans l'enfant, ce dernier est créé dans la base de données s'il s'agit d'un enfant avec droits de propriété. Les enfants sans droits de propriétés sont extraits pour que leur présence soit validée dans la base de données.
- Si une instruction Delete est trouvée dans l'enfant, ce dernier est supprimé.
- Si une instruction DeltaUpdate est trouvée dans l'enfant, ce dernier est mis à jour dans la base de données.

Opérations de suppression

Lors de la suppression d'un objet métier, le connecteur renvoie l'état `SUCCESS` si l'opération a réussi ou `FAIL` si l'opération a échoué. L'objet métier parent est tout d'abord extrait. Ensuite, l'adaptateur supprime de manière récursive tous les enfants de type cardinalité simple ayant une relation avec droits de propriété avec le parent, puis l'objet métier parent lui-même et enfin tous les enfants de cardinalité N. Les enfants de type cardinalité simple sans droits de propriété ne sont jamais supprimés. Si aucun objet métier n'existe, le connecteur renvoie le message `FAIL`.

Le connecteur prend en charge la suppression logique et physique, selon la valeur SCN (Status Column Name) dans les informations spécifiques à l'application de l'objet. Si la valeur SCN est définie, le connecteur procède à une suppression logique. Si la valeur SCN n'est pas définie, le connecteur procède à une suppression physique.

Suppression physique : Lors de la suppression physique d'un objet métier hiérarchique, le connecteur procède aux étapes suivantes :

1. Il supprime de manière récursive tous les objets métier enfant de type cardinalité simple contenus avec droits de propriétés.
2. Il supprime l'objet métier de niveau supérieur.

3. Il supprime de manière récursive tous les objets métier enfant de type cardinalité multiple.

Remarque : Un objet métier de niveau supérieur correspondant à un encapsuleur ne dispose d'aucune table de base de données correspondante : il ne sera, par conséquent, pas supprimé de la base de données. Toutes les valeurs d'attributs simples d'un encapsuleur seront ignorées.

Suppression logique : Lors de la suppression logique d'un objet métier, le connecteur procède aux étapes suivantes :

1. Il émet une instruction `UPDATE` qui affecte à l'attribut d'état de l'objet métier la valeur spécifiée par les informations d'application de l'objet métier. Le connecteur vérifie que la mise à jour concerne une seule ligne de la base de données puis renvoie un message d'erreur dans le cas contraire.
2. Il supprime logiquement tous les enfants de type cardinalité simple contenus avec droits de propriété et tous les enfants de type cardinalité multiple de manière récursive. Le connecteur ne supprime pas les enfants de type cardinalité simple contenus sans droits de propriété.

Instructions SQL

Le connecteur peut utiliser des instructions SQL simples pour les opérations de sélection, de mise à jour, d'extraction ou de suppression. Les noms des colonnes relatives aux instructions SQL proviennent de la propriété `AppSpecificInfo` d'un attribut. Chaque requête couvre une seule table, à moins qu'elle ne désigne une vue.

Procédures stockées

Une procédure stockée est un groupe d'instructions SQL formant une unité logique et réalisant une tâche spécifique. Une procédure stockée rassemble un ensemble d'opérations ou de requêtes que le connecteur doit exécuter sur un objet dans un serveur de base de données.

Le connecteur fait appel aux procédures stockées dans les cas suivants :

- avant le traitement d'un objet métier pour réaliser les processus opérationnels préparatoires ;
- après le traitement d'un objet métier pour réaliser les processus post-opérationnels ;
- pour réaliser un ensemble d'opérations sur un objet métier, au lieu d'utiliser une simple instruction `INSERT`, `RETRIEVE`, `UPDATE` ou `DELETE`.

Lors du traitement d'un objet métier hiérarchique, le connecteur peut utiliser une procédure stockée pour traiter l'objet métier de niveau supérieur ou l'un de ses objets métier enfant. Toutefois, chaque objet métier ou tableau d'objets métier doit avoir sa propre procédure stockée.

Spécification d'une procédure stockée

Cette section décrit les étapes à entreprendre pour obliger le connecteur à utiliser une procédure stockée pour un objet métier. Elle contient les sections suivantes :

Ajout d'attributs à l'objet métier : Vous devez ajouter une catégorie spécifique d'attributs à l'objet métier pour chaque type de procédure stockée traitée par le connecteur. Ces attributs représentent uniquement le type de procédure stockée et les informations spécifiques à l'application qui la définissent. Ces attributs n'utilisent pas les paramètres des informations spécifiques à l'application disponibles pour un attribut simple standard.

Nommez l'attribut en fonction du type de procédure stockée à utiliser. Par exemple, pour obliger le connecteur à utiliser les procédures stockées AfterUpdate et BeforeRetrieve, ajoutez les attributs `AfterUpdateSP` et `BeforeRetrieveSP`.

Le connecteur reconnaît les noms d'attributs d'objets métier suivants :

```
BeforeCreateSP
AfterCreateSP
CreateSP
BeforeUpdateSP
AfterUpdateSP
UpdateSP
BeforeDeleteSP
AfterDeleteSP
DeleteSP
BeforeRetrieveSP
AfterRetrieveSP
RetrieveSP
BeforeRetrieveByContentSP
AfterRetrieveByContentSP
RetrieveByContentSP
BeforeRetrieveUpdateSP
AfterRetrieveUpdateSP
RetrieveUpdateSP
BeforeDeltaUpdateSP
AfterDeltaUpdateSP
DeltaUpdateSP
```

Remarque : Créez un attribut uniquement pour les procédures stockées que vous souhaitez que le connecteur exécute. Utilisez les informations spécifiques à l'application ou le mappage (uniquement si InterChange Server est utilisé en tant que courtier d'intégration) pour spécifier les valeurs de ces attributs avant que l'objet métier ne soit envoyé au connecteur. Le connecteur doit être redémarré pour prendre en compte les modifications apportées à ces valeurs pour les appels ultérieurs sur un objet métier.

Syntaxe d'une procédure stockée : La syntaxe utilisée pour spécifier une procédure stockée est la suivante :

```
SPN=StoredProcedureName;RS=true|false[;IP=Attribute_Name1[:Attribute_Name2[:...]]]
[;OP=Attribute_Name1| RS[:Attribute_Name2| RS[:...]]]
[;IO=Attribute_Name1[:Attribute_Name2[:...]]]
```

où :

StoredProcedureName	Nom de la procédure stockée.
RS	A la valeur `true` si la procédure stockée renvoie un ensemble de résultats, ou `false` dans le cas contraire. La valeur par défaut est `false`. Si la valeur est `true`, la propriété `ColumnName` contenue dans les informations d'application d'un attribut désigne la colonne appropriée dans l'ensemble de résultats. Si le paramètre RS fait partie de la liste

	des paramètres de sortie, ce paramètre spécifique renvoie un ensemble de résultats. Un seul paramètre `OUT` pour l'ensemble de résultats est pris en charge. Si plusieurs ensembles de résultats sont renvoyés sous la forme d'un paramètre `OUT`, seul le premier ensemble de résultats est renvoyé et les autres sont ignorés. Actuellement, cette fonction est prise en charge pour Oracle 8i et versions ultérieures et pour les procédures stockées qui utilisent le pilote JDBC Oracle. Pour la procédure stockée dans la base de données, le paramètre correspondant doit renvoyer un type `REFCURSOR`.
`IP`	Input Parameters (paramètres d'entrée) : liste des attributs d'objets métier dont le connecteur doit utiliser les valeurs en tant que valeurs d'entrée lors de l'exécution de la procédure stockée.
`OP`	Output Parameters (paramètre de sortie) : liste des attributs d'objets métier auxquels le connecteur doit renvoyer les valeurs après l'exécution de la procédure stockée. Voir le paramètre RS pour obtenir une description de l'ensemble de résultats.
`IO`	InputOutput Parameters (paramètres d'entrée-sortie) : liste des attributs d'objets métier dont le connecteur doit utiliser les valeurs en tant que valeurs d'entrée et auxquels le connecteur doit renvoyer les valeurs après l'exécution de la procédure stockée.

L'ordre des valeurs `StoredProcedureName`, `RS` et des paramètres est important, alors que l'ordre des paramètres entre eux ne l'est pas. En d'autres termes, cela ne fait aucune différence pour le connecteur si la procédure stockée regroupe tous les paramètres de chaque type ou qu'elle disperse les types de paramètre. Lorsque plusieurs paramètres de même type sont regroupés ensemble, séparez les valeurs par deux-points (:). Il n'est pas nécessaire de répéter le type de paramètre pour chaque valeur. Séparez les paramètres de types différents par un point virgule. Lorsque vous indiquez des valeurs de paramètre, n'insérez aucun espace de chaque côté du signe égal (=).

Exemples de procédures stockées renvoyant des ensembles de résultats : Les exemples suivants utilisent les procédures stockées nommées `CustomerAddressRetrieve` et `CustomerAddressRetrieveForOracleDB` pour renvoyer un ensemble de résultats qui contient plusieurs adresses et qui est utilisé pour créer un objet métier enfant de cardinalité N.

Remarque : Les ensembles de résultats sont traités pour l'attribut RetrieveSP uniquement et sont utilisés pour créer un objet métier enfant de cardinalité N.

Pour les bases de données Oracle, l'ensemble de résultats est renvoyé en tant que paramètre de sortie et est traité en conséquence par l'adaptateur. Pour les autres bases de données, l'ensemble de résultats est une valeur de retour provenant de la procédure stockée.

- CustomerAddressRetrieve (pour les bases de données autres que Oracle)

```
Attribute : RetrieveSP
ASI : SPN=CustomerAddressRetrieve;RS=true;IP=CustomerName:IP=Customerld;
OP=ErrorStatus;OP=ErrorMsg
```

- CustomerAddressRetrieveForOracleDB (pour les bases de données Oracle)

```
Attribute : RetrieveSP
ASI : SPN=CustomerAddressRetrieveForOracleDB;RS=true;IP=CustomerName:
IP=Customerld;OP=RS;OP=ErrorStatus;OP=ErrorMsg
```

(OP=RS signifie que le premier paramètre de sortie est un ensemble de résultats)

Exemples de procédures stockées ne renvoyant pas d'ensemble de résultats : Les exemples suivants utilisent les procédures stockées nommées `CustomerInsert` et `VendorInsert` qui obtiennent des valeurs auprès de deux attributs d'entrée et renvoient ces valeurs à quatre attributs de sortie. Ces exemples illustrent différentes structures de procédures stockées.

- Les paramètres de même type sont regroupés ensemble (IP, IP, OP, OP, OP, OP, IO) :

```
SPN=CustomerInsert;RS=false;IP=LastName:FirstName;OP=CustomerName:
CustomerID:ErrorStatus:ErrorMessage;IO=VendorID
```

- Les paramètres de même type sont dispersés (IP, OP, OP, OP, IP, IO, OP) :

```
SPN=VendorInsert;RS=false;IP=LastName;OP=CustomerName:
CustomerID:ErrorStatus;IP=FirstName;IO=VendorID;OP=ErrorMessage
```

Le connecteur prend en charge uniquement les types de données simples pris en charge par le pilote JDBC.

Spécification de la procédure stockée : Il existe deux méthodes pour spécifier le nom de la procédure stockée et ses valeurs de paramètre :

- Propriété `AppSpecificInfo` de l'attribut

  Si la longueur du texte qui spécifie la procédure stockée est inférieure ou égale à 4000 octets, vous pouvez spécifier la valeur dans la propriété `AppSpecificInfo` de l'attribut. Vous pouvez utiliser cette propriété pour spécifier la procédure stockée, que le connecteur ait lancé une requête pour rechercher l'objet métier (en d'autres termes, l'objet métier représente un événement d'application) ou qu'il ait reçu l'objet métier en tant que requête du courtier d'intégration.

  L'exemple suivant illustre la spécification de la procédure stockée dans les informations spécifiques à l'application. Dans ce cas, la valeur spécifiée pour la propriété `MaxLength` n'a aucune importance pour la procédure stockée.

```
   [Attribute]
   Name = BeforeCreateSP
   Type = String
   MaxLength = 15
   IsKey = false
   IsRequired = false
   AppSpecificInfo = SPN=ContactInsert;IP=LastName:FirstName;OP=CustomerName:
CustomerID:ErrorStatus:ErrorMessage

   [End]
```

- Valeur d'attribut (pertinente uniquement si InterChange Server est utilisé en tant que courtier d'intégration)

  Si la longueur du texte qui spécifie la procédure stockée est supérieure à 4000 octets, vous devez utiliser le mappage pour spécifier la procédure stockée. Vous pouvez utiliser le mappage pour spécifier la procédure stockée uniquement si l'objet métier représente une requête du courtier d'intégration. En d'autres termes, vous ne pouvez pas utiliser la valeur d'un attribut pour spécifier une procédure stockée lorsque le connecteur recherche des événements.

Si la longueur du texte de la procédure stockée est supérieure à 4000 octets et que vous utilisez le mappage pour la spécifier, n'oubliez pas d'augmenter la valeur de la propriété `MaxLength` pour l'adapter au texte intégral.

Remarque : Si une procédure stockée qui gère une opération de création, de mise à jour ou de suppression est exécutée sur un objet métier hiérarchique contenant un tableau d'objets métier enfant, le connecteur traite chaque objet métier enfant individuellement. Par exemple, si le connecteur exécute une procédure stockée `BeforeCreate`, il ne traite pas le tableau en tant qu'unité mais traite chaque membre de ce tableau. Lorsqu'il traite une procédure stockée `BeforeRetrieve`, le connecteur agit sur un seul objet métier. Lorsqu'il traite une procédure stockée `AfterRetrieve`, le connecteur agit sur tous les objets métier renvoyés par l'extraction.

Traitement des objets métier à l'aide des procédures stockées ou des instructions SQL simples

Les sections suivantes expliquent comment le connecteur traite les procédures stockées :

- «Opérations de création d'objets métier»
- «Opérations de mise à jour d'objets métier» à la page 54
- «Opérations de suppression d'objets métier» à la page 54
- «Opérations d'extraction d'objets métier» à la page 55
- «Opérations d'extraction par contenu d'objets métier» à la page 56
- «Opérations d'extraction pour mise à jour d'objets métier» à la page 56

Opérations de création d'objets métier : Une procédure stockée de création renvoie généralement les valeurs utilisées par le connecteur pour renseigner les attributs simples dans l'objet métier de niveau supérieur. Le connecteur procède comme suit lors du traitement des procédures stockées de création (`BeforeCreate`, `Create`, `AfterCreate`) :

1. Il vérifie que l'objet métier contient un attribut `BeforeCreateSP`. Le cas échéant, il appelle la procédure stockée `BeforeCreate`.
2. Si la procédure stockée renvoie des valeurs via les paramètres de sortie, il utilise ces valeurs pour définir la valeur des attributs simples dans l'objet métier.
3. Il crée les objets métier enfant de type cardinalité simple.
4. Il affecte à chacune des valeurs de clé étrangère de l'objet métier de niveau supérieur la valeur de clé primaire de chaque objet métier enfant de type cardinalité simple.
5. Il vérifie que l'objet métier contient un attribut `CreateSP`. Le cas échéant, il appelle la procédure stockée `Create` pour créer l'objet métier de niveau supérieur. Dans le cas contraire, il génère et exécute une instruction `INSERT` pour créer l'objet métier de niveau supérieur.
6. Si la procédure stockée `Create` renvoie des valeurs via les paramètres de sortie, il utilise ces valeurs pour définir la valeur des attributs simples dans l'objet métier.
7. Il affecte à la valeur de clé étrangère de chaque enfant de type cardinalité multiple la valeur de l'attribut de clé primaire de leur parent.
8. Il crée les objets métier enfant de type cardinalité multiple.
9. Il vérifie que l'objet métier contient un attribut `AfterCreateSP`. Le cas échéant, il appelle la procédure stockée `AfterCreate`.

10. Si la procédure stockée renvoie des valeurs via les paramètres de sortie, il utilise ces valeurs pour définir les valeurs des attributs simples dans l'objet métier.

Le connecteur peut utiliser les valeurs renvoyées à l'étape 10 pour modifier les valeurs d'un objet métier créé au cours des étapes 3 ou 5.

Opérations de mise à jour d'objets métier : Une procédure stockée de mise à jour renvoie généralement les valeurs utilisées par le connecteur pour renseigner les attributs simples dans l'objet métier de niveau supérieur. Le connecteur procède comme suit lors du traitement des procédures stockées de mise à jour (`BeforeUpdate`, `Update`, `AfterUpdate`) :

1. Il vérifie que l'objet métier contient un attribut `BeforeUpdateSP`. Le cas échéant, il appelle la procédure stockée `BeforeUpdate`.
2. Si la procédure stockée `BeforeUpdate` renvoie des valeurs via les paramètres de sortie, il utilise ces valeurs pour définir la valeur des attributs simples dans l'objet métier.
3. Il met à jour les objets métier enfant de type cardinalité simple.
4. Il affecte à chacune des valeurs de clé étrangère de l'objet métier de niveau supérieur la valeur de clé primaire de chaque objet métier enfant contenu de type cardinalité simple.
5. Il vérifie que l'objet métier contient un attribut `UpdateSP`. Le cas échéant, il appelle la procédure stockée `Update` pour mettre à jour l'objet métier de niveau supérieur. Dans le cas contraire, il génère et exécute une instruction `UPDATE` pour mettre à jour l'objet métier de niveau supérieur.
6. Si la procédure stockée `Update` renvoie des valeurs via les paramètres de sortie, il utilise ces valeurs pour définir la valeur des attributs simples dans l'objet métier.
7. Il définit les valeurs de clé étrangère dans les enfants de type cardinalité multiple pour faire référence à la valeur contenue dans les attributs de clé primaire correspondants dans le parent.
8. Il met à jour les objets métier enfant de type cardinalité multiple.
9. Il vérifie que l'objet métier contient un attribut `AfterUpdateSP`. Le cas échéant, il appelle la procédure stockée `AfterUpdate`.
10. Si la procédure stockée renvoie des valeurs via les paramètres de sortie, il utilise ces valeurs pour définir la valeur des attributs simples dans l'objet métier.

Opérations de suppression d'objets métier : Une procédure stockée de suppression ne renvoie pas les valeurs au connecteur. Le connecteur procède comme suit lors du traitement des procédures stockées de suppression (`BeforeDelete`, `Delete`, `AfterDelete`) :

1. Il vérifie que l'objet métier contient un attribut `BeforeDeleteSP`. Le cas échéant, il appelle la procédure stockée BeforeDelete.
2. Il supprime les objets métier enfant de type cardinalité simple.
3. Il supprime les objets métier enfant de type cardinalité multiple.
4. Il vérifie que l'objet métier contient un attribut `DeleteSP`. Le cas échéant, il appelle la procédure stockée Delete pour supprimer l'objet métier de niveau supérieur. Dans le cas contraire, il génère et exécute une instruction `DELETE`.
5. Il vérifie que l'objet métier contient un attribut `AfterDeleteSP`. Le cas échéant, il appelle la procédure stockée `AfterDelete`.

Opérations d'extraction d'objets métier : Pour les opérations d'extraction simples, les procédures stockées peuvent être utilisées pour l'objet métier de niveau supérieur, pour les enfants de type cardinalité simple ainsi que pour les enfants de type cardinalité multiple. L'ordre des procédures est le suivant :

- `BeforeRetrieve`
- `Retrieve`
- `AfterRetrieve`

Le connecteur crée un objet temporaire pour extraire un objet métier enfant de type cardinalité simple ou multiple. Le connecteur applique ensuite la procédure stockée `BeforeRetrieve` à l'objet métier temporaire. La procédure stockée `AfterRetrieve` est alors appliquée à chaque objet enfant extrait pour le conteneur.

Le connecteur exécute la procédure stockée `AfterRetrieve` après avoir exécuté une requête `Retrieve` générée dynamiquement à partir des métadonnées de l'objet métier ou de la procédure stockée sur l'objet métier.

Selon la spécification JDBC, il existe trois types d'appel `StoredProcedure` comme suit :

- `{call <spName>(?,?,?)}`
- `{call <spName>}`
- `{?= call <spName>(?,?,?)}`

où `spName` désigne le nom de la procédure stockée.

Le connecteur prend en charge les deux premiers types. Il traitera le paramètre `ResultSet` renvoyé par l'appel `StoredProcedure`.

Si `RS=true` dans la syntaxe de la procédure stockée, l'ensemble de résultats provenant de la procédure stockée est traité. Si `RS=false`, l'ensemble de résultats n'est pas traité. La valeur par défaut du paramètre `RS` est `false`. Une fois les valeurs de l'ensemble de résultats traitées, les variables de sortie de la procédure stockée sont traitées. Si `RS=true`, les enfants de type cardinalité multiple ne peuvent pas spécifier les variables de sortie dans la procédure stockée associée.

Remarque : Le traitement de l'ensemble de résultats est pris en charge uniquement pour les opérations de l'instruction Retrieve et de l'attribut RetrieveSP.

Traitement de l'ensemble de résultats renvoyé par la procédure stockée d'extraction (RetrieveSP) : Le paramètre `ResultSetMetaData` est obtenu pour l'ensemble de résultats renvoyé par la procédure stockée. Les valeurs de toutes les colonnes contenues dans l'ensemble de résultats sont obtenues et définies sur l'attribut correspondant de l'objet métier. La propriété ColumnName des informations d'application d'un attribut doit contenir le nom de la colonne ResultSet pour faire correspondre l'attribut et la colonne.

Pour les objets de type cardinalité simple, l'ensemble de résultats correspondant ne doit contenir qu'une seule ligne. Si plusieurs lignes sont renvoyées dans l'ensemble de résultats, un message d'erreur est signalé.

Pour les enfants de type cardinalité multiple, plusieurs lignes peuvent être renvoyées via l'ensemble de résultats. Pour chaque ligne renvoyée, un objet est créé et ajouté au conteneur. Le conteneur est ensuite ajouté à l'objet parent à l'index d'attribut obligatoire.

L'enfant de cardinalité N d'un objet métier encapsuleur possède les attributs de la procédure stockée qui représentent les paramètres d'entrée et les colonnes de l'ensemble de résultats. Le paramètre `WRAPPER=true` est défini au niveau des informations spécifiques à l'application de l'objet métier. Les informations d'application de l'objet métier enfant ont la valeur `TN=dummy`.

Opérations d'extraction par contenu d'objets métier : Pour les opérations `RetrieveByContent` simples, les procédures stockées peuvent uniquement être utilisées pour l'objet métier de niveau supérieur et ses enfants de type cardinalité simple. En d'autres termes, elles ne peuvent pas être utilisées pour renvoyer un ensemble de résultats ou plusieurs lignes. L'ordre des procédures est le suivant :

- `BeforeRetrieveByContent`
- `RetrieveByContent`
- `AfterRetrieveByContent`

Le connecteur crée un objet temporaire pour extraire un objet métier enfant de type cardinalité simple ou multiple. Pour les objets métier de type cardinalité multiple, le connecteur applique ensuite la procédure stockée `BeforeRetrieveByContent` à l'objet métier temporaire. La procédure stockée `AfterRetrieveByContent` est alors appliquée à chaque objet enfant extrait pour le conteneur.

Le connecteur exécute la procédure stockée `AfterRetrieveByContent` après avoir exécuté une requête `RetrieveByContent` générée dynamiquement à partir des métadonnées de l'objet métier ou de la procédure stockée sur l'objet métier. Dans ce cas, bien que l'extraction d'un objet métier hiérarchique entraîne également l'extraction de ses objets métier enfant, le connecteur exécute la procédure stockée `AfterRetrieveByContent` sur chaque objet métier présent dans le tableau.

Opérations d'extraction pour mise à jour d'objets métier : Les procédures stockées suivantes sont appelées sur l'objet métier de niveau supérieur et extraient tous les objets métier enfant de la même manière que lorsqu'il s'agit de l'instruction simple `Retrieve`.

L'ordre des procédures est le suivant :

- `BeforeRetrieveUpdate`
- RetrieveUpdate
- `AfterRetrieveUpdate`

Ces procédures stockées procèdent aux mêmes opérations que les procédures `BeforeRetrieve` et `AfterRetrieve`. Elles portent un nom distinctif pour que vous puissiez créer des attributs distincts afin d'obliger le connecteur à exécuter les opérations `BeforeRetrieve` et `BeforeRetrieveUpdate`, ainsi que les opérations `AfterRetrieve` et `AfterRetrieveUpdate`.

Le connecteur crée un objet temporaire pour extraire un objet métier enfant de type cardinalité simple ou multiple. Pour les objets métier de type cardinalité multiple, le connecteur applique ensuite la procédure stockée `BeforeRetrieveUpdate` à l'objet métier temporaire. La procédure stockée `AfterRetrieveUpdate` est alors appliquée à chaque objet enfant extrait pour le conteneur.

Le connecteur exécute la procédure stockée `AfterRetrieveUpdate` après avoir exécuté une requête `RETRIEVE` générée dynamiquement à partir des métadonnées de l'objet métier ou de la procédure stockée sur l'objet métier. Dans ce cas, bien

que l'extraction d'un objet métier hiérarchique entraîne également l'extraction de ses objets métier enfant, le connecteur exécute la procédure stockée `AfterRetrieveUpdate` sur chaque objet métier présent dans le tableau.

Validation et annulation de transaction

Chaque fois que le connecteur reçoit un objet métier à traiter, il commence un bloc de transactions. Toutes les instructions `SQL` que le connecteur exécute lors du traitement de cet objet métier sont regroupées dans le bloc de transactions. Une fois le traitement de l'objet métier terminé, le connecteur valide le bloc de transactions si le traitement est une réussite ou l'annule s'il rencontre une erreur.

Propriétés des attributs d'objet métier

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

Propriété de nom

Chaque attribut d'objet métier doit avoir un nom unique.

Propriété de type

Chaque attribut d'objet métier doit avoir un type, tel que `Integer`, `String` ou le type d'un objet métier enfant. Lorsque le connecteur rencontre un attribut de type `Date`, `Long Text` ou `String`, il insère la valeur entre guillemets puis la traite en tant que données de type caractères.

Propriété de cardinalité

Chaque attribut d'objet métier représentant un enfant ou un tableau d'objets métier enfant a la valeur 1 ou N respectivement. Tous les attributs représentant des objets métier enfant ont également une propriété `ContainedObjectVersion` (qui indique le numéro de version de l'enfant) et une propriété `Relationship` (qui indique la valeur du paramètre `Containment`).

Propriété de longueur maximale

Si l'attribut est de type `String`, ce type de propriété indique la longueur maximale autorisée pour la valeur de l'attribut.

Propriété de clé

Un attribut simple au minimum doit être spécifié dans chaque objet métier en tant que clé. Pour définir un attribut en tant que clé, affectez à cette propriété la valeur `Yes`. Si l'attribut d'objet métier est de type `String`, il est recommandé de définir des données de type `Varchar` plutôt que `char` dans la base de données.

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

Si la propriété de clé a la valeur `true` pour un attribut simple, le connecteur ajoute cet attribut à la clause `WHERE` des instructions SQL `SELECT`, `UPDATE`, `RETRIEVE` et `DELETE` générées lors du traitement de l'objet métier.

Si la propriété de clé a la valeur `true` pour un attribut dans un enfant qui stocke la relation parent-enfant dans l'enfant (de type cardinalité simple et multiple), le connecteur utilise les clés primaires du parent contenus dans la clause `WHERE` de l'instruction `SELECT` et n'utilise pas la propriété `Key`. Pour plus d'informations sur la spécification du nom des attributs d'objet métier dont les valeurs sont utilisées pour définir les attributs de clé étrangère de l'enfant, voir «Informations spécifiques à l'application au niveau de l'attribut» à la page 61.

Propriété de clé étrangère

Le connecteur utilise cette propriété pour déterminer si un attribut est une clé étrangère.

Propriété obligatoire

La propriété `Required` indique si un attribut doit contenir une valeur.

Si cette propriété est spécifiée pour un attribut qui représente un objet métier de type cardinalité simple, le connecteur oblige l'objet métier parent à contenir un objet métier enfant pour cet attribut.

Lorsque le connecteur reçoit un objet métier avec une requête Create, le connecteur provoque l'échec de l'opération de création si les deux conditions suivantes sont réunies :

- Un attribut requis de l'objet métier ne dispose d'aucune valeur correcte ou d'aucune valeur par défaut.
- Les informations spécifiques à l'application n'indiquent pas que le connecteur génère l'ID unique.

Lorsque le connecteur reçoit un objet métier avec une requête Retrieve et qu'un attribut requis de l'objet métier ne dispose d'aucune valeur correcte ou d'aucune valeur par défaut, le connecteur provoque l'échec de l'opération d'extraction.

Le connecteur n'utilise pas cette propriété pour les attributs contenant un tableau d'objets métier enfant.

Remarque : Si l'attribut de clé utilise une séquence ou un compteur ou qu'il est renseigné par la base de données (`UID=AUTO`), il ne doit pas porter la valeur Required.

Propriété AppSpecificInfo

Pour plus d'informations sur cette propriété, voir «Informations spécifiques à l'application au niveau de l'attribut» à la page 61.

Propriété de valeur par défaut

Cette propriété indique la valeur par défaut que le connecteur utilise pour renseigner un attribut simple s'il n'est pas renseigné par une valeur dans la table de la base de données. Le connecteur n'évalue pas cette propriété pour les attributs représentant un objet métier enfant ou un tableau d'objets métier enfant.

Le connecteur évalue cette propriété uniquement si la propriété de configuration `UseDefaults` a la valeur `true`. Pour plus d'informations, voir tableau 6 à la page 19.

Valeur d'attribut spéciale

Les attributs simples contenus dans les objets métier peuvent avoir la valeur spéciale `CxIgnore`. Lorsqu'il reçoit un objet métier provenant du courtier d'intégration, le connecteur ignore tous les attributs dotés de la valeur `CxIgnore`. Il fait comme si ces attributs étaient invisibles.

Lorsque le connecteur extrait des données de la base de données et que l'instruction `SELECT` renvoie une valeur `null` pour un attribut, le connecteur affecte à cet attribut la valeur `CxIgnore` par défaut. Si une valeur a été spécifiée pour le paramètre `UNVL` des informations d'application de l'attribut, le connecteur utilise cette valeur pour représenter la valeur `null`.

Dans la mesure où le connecteur demande à ce que chaque objet métier dispose au minimum d'un attribut de clé primaire, les développeurs doivent veiller à ce que les objets métier WebSphere Business Integration Adapter transmis au connecteur disposent au minimum d'une clé primaire n'ayant pas la valeur `CxIgnore`. La seule exception à cette condition concerne un objet métier dont la clé primaire doit être générée par le connecteur à l'aide d'un compteur ou d'une séquence ou par la base de données.

Lorsque le connecteur insère des données dans la base de données et qu'un attribut d'objet métier ne dispose d'aucune valeur spécifiée, il utilise la valeur spécifiée par la propriété `UseNullValue` de l'attribut. Pour plus d'informations sur la propriété `UseNullValue`, voir `UNVL=value` dans tableau 11 à la page 62.

Informations spécifiques à l'application d'objets métier

Cette section fournit des informations au niveau de l'objet, de l'attribut et du format des informations d'application des instructions pour les objets métier pris en charge par le connecteur.

Les informations spécifiques à l'application dans les définitions d'objets métier fournissent au connecteur les instructions dépendantes de l'application concernant le traitement des objets métier. Le connecteur analyse les informations spécifiques à l'application depuis les attributs ou l'instruction d'un objet métier ou depuis l'objet métier lui-même pour générer des requêtes pour les opérations de création, de mise à jour, d'extraction ou de suppression.

Le connecteur stocke quelques-unes des informations d'application de l'objet métier en mémoire cache et les utilise pour générer des requêtes pour toutes les instructions.

Si vous développez ou modifiez un objet métier spécifique à une application, vous devez veiller à ce que les informations spécifiques à l'application contenues dans la définition d'objet métier correspondent à la syntaxe attendue par le connecteur.

Remarque : Dans un environnement InterChange Server, les performances sont plus significatives lorsque la taille des objets métier spécifiques à l'application est inférieure à 1 Mo. Ils ne doivent cependant jamais dépasser 5 Mo. Les objets métier de grande taille entraînent des incidents de performance en raison des limitations établies par la machine virtuelle Java sur laquelle InterChange Server s'exécute.

Le tableau 10 à la page 60 présente la fonctionnalité disponible dans les informations spécifiques à l'application des objets métier.

Tableau 10. Présentation des informations spécifiques à l'application dans les objets métier pris en charge

Portée des informations spécifiques à l'application	Fonctionnalité
Intégralité de l'objet métier	Indique : • le nom de la table de base de données correspondante ; • la colonne dont le connecteur utilise la valeur dans la clause `WHERE` pour procéder à une suppression logique (ou logicielle) ; • si l'objet métier de niveau supérieur est un encapsuleur.
Attributs simples	Indique : • le nom de colonne de la base de données d'un attribut ; • la relation de clé étrangère entre un attribut dans l'objet métier courant et un objet métier parent ou enfant ; • la génération automatique de valeurs d'ID uniques ; • le nom d'un autre attribut dans le même objet métier dont le connecteur doit utiliser la valeur pour définir la valeur de l'attribut courant ; • si l'attribut courant doit être utilisé lors du tri d'une extraction ; • la valeur à utiliser lorsque la valeur de l'attribut courant est `null` ; • le comportement de substitution de chaîne ; • l'opérateur à utiliser (`LIKE` ou `=`) lors de la comparaison de chaînes ; • la valeur à utiliser en tant que position des caractères génériques lorsque l'opérateur `LIKE` est utilisé.
Attributs contenant un enfant ou un tableau d'objets métier enfant	Indique si un enfant de type cardinalité simple appartient au parent. Indique si le connecteur supprime les données enfant au cours d'une opération de mise à jour si ces données ne sont pas représentées dans l'objet métier source.
Instruction d'objet métier	Utilisé uniquement pour l'instruction Retrieve, ce texte indique les attributs à inclure dans la clause `WHERE` lors d'une extraction. Vous pouvez également spécifier des opérateurs et des valeurs d'attribut.

Les sections suivantes traitent cette fonctionnalité plus en détail.

Informations spécifiques à l'application au niveau de l'objet métier

Les informations spécifiques à l'application au niveau de l'objet métier vous permettent :

- de spécifier le nom de la table de base de données correspondante ;
- de fournir les informations nécessaires à la réalisation d'une suppression logique ou physique ;
- de spécifier que l'objet métier de niveau supérieur est un objet encapsuleur.

Au niveau de l'objet métier, les informations spécifiques à l'application comportent des paramètres séparés par deux-points (:) ou un point-virgule (;) :

`TN=`*`TableName`*`; SCN=`*`StatusColumnName`*`:`*`StatusValue`*`; WRAPPER=true|false`

où TableName identifie la table de la base de données, *StatusColumnName* désigne le nom de la colonne de la base de données utilisée pour réaliser les suppressions logiques et *StatusValue* correspond à la valeur indiquant qu'un objet métier est inactif ou supprimé.

Par exemple, supposons que la valeur suivante soit indiquée pour les informations spécifiques à l'application d'un objet métier Customer :

```
TN=CUSTOMER; SCN=CUSTSTATUS:DELETED
```

Supposons également que le connecteur reçoive une requête pour supprimer le client. Une telle valeur oblige le connecteur à exécuter l'instruction SQL suivante :

```
UPDATE CUSTOMER SET CUSTSTATUS = 'DELETED' WHERE CUSTOMER_ID = 2345
```

Si le paramètre SCN n'est pas inclus ou qu'aucune valeur n'est spécifiée pour ce paramètre, le connecteur supprime physiquement l'objet métier de la base de données. En d'autres termes, si l'objet métier doté de l'instruction Delete inclut le paramètre SCN dans ses informations spécifiques à l'application, le connecteur procède à une suppression logique. Si l'objet métier doté de l'instruction Delete n'inclut pas le paramètre SCN dans ses informations spécifiques à l'application, le connecteur procède à une suppression physique.

Les opérations de mise à jour et de suppression peuvent utiliser la valeur de la propriété SCN :

- Lors de l'exécution d'une mise à jour, le connecteur utilise la valeur de sa propriété ChildUpdatePhyDelete pour déterminer si les données enfant manquantes doivent être supprimées de manière logique ou physique. S'il utilise la suppression logique des données enfant, il utilise la valeur de son paramètre SCN pour obtenir le nom de la colonne d'état et le texte de la valeur d'état. Pour plus d'informations, voir «Opérations de mise à jour» à la page 45.
- Lors de l'exécution d'une suppression, le connecteur utilise la valeur de son paramètre SCN pour déterminer si l'intégralité de l'objet métier doit être supprimée de manière logique ou physique. Si le paramètre SCN contient une valeur, le connecteur procède à une suppression logique. Si le paramètre SCN ne contient aucune valeur, le connecteur procède à une suppression physique. Pour plus d'informations, voir «Opérations de suppression» à la page 48.

Au niveau de l'objet métier, les informations spécifiques à l'application peuvent être utilisées pour spécifier un encapsuleur :

```
WRAPPER=true|false
```

Si le paramètre de l'encapsuleur a la valeur true, l'objet métier de niveau supérieur est un objet encapsuleur. L'objet encapsuleur n'est pas représenté par une table de base de données ou une vue. Un encapsuleur est utilisé en tant que conteneur pour les objets métier non associés. Le connecteur ignore l'objet de niveau supérieur et traite uniquement les enfants. L'objet encapsuleur peut contenir les entités de cardinalité N et/ou N-1.

Informations spécifiques à l'application au niveau de l'attribut

Les informations spécifiques à l'application pour les attributs diffèrent selon que l'attribut est un attribut simple ou un attribut représentant un enfant ou un tableau d'objets métier enfant. Les informations spécifiques à l'application pour un attribut représentant un enfant diffèrent également selon que la relation parent-enfant est stockée dans l'enfant ou dans le parent. Pour plus d'informations sur les

informations spécifiques à l'application pour les attributs représentant un enfant ou un tableau d'objets métier enfant, voir «Spécification de la clé étrangère d'un attribut» à la page 64.

Informations spécifiques à l'application pour les attributs simples

Pour les attributs simples, le format des informations spécifiques à l'application comprend onze paramètres nom-valeur, chacun incluant le nom du paramètre et sa valeur. Chaque ensemble de paramètres est séparé par deux-points (:).

Le format des informations spécifiques à l'application est indiqué ci-dessous. Les crochets ([ ]) entourent un paramètre facultatif. Une barre verticale (|) sépare les membres d'un jeu d'options. Réservez aux deux-points la fonction de délimiteur.

```
CN=col_name:[FK=[fk_object_name.]fk_attribute_name]:
[UID=[AUTO|uid_name| schema_name.uid_name[=UseIfMissing]|CW.uidcolumnname
[=UseIfMissing]]]:
[PH=true|false]:[CA=set_attr_name| ..set_attr_name]:[OB=[ASC|DESC]]:[UNVL=value]:
[ESC=true|false]:[FIXEDCHAR=true|false]:
[BYTEARRAY=true|false]:[USE_LIKE=true|false:
[WILDCARD_POSITION=non-negative number|NONE|BEGIN|END|BOTH]:
[CLOB=true]:
{TS=true|false]]
```

Le seul paramètre requis pour un attribut simple que le connecteur doit traiter est le nom de colonne. Par exemple, pour indiquer uniquement le nom de colonne, utilisez le format suivant :

```
CN=customer_id
```

Le tableau 11 décrit chaque paramètre nom-valeur.

Tableau 11. Paramètres nom-valeur dans les informations d'application d'un attribut

Paramètre	Description
CN=*col_name*	Nom de la colonne de base de données pour cet attribut.
FK=[*fk_object_name*.]*fk_attribute_name*	La valeur de cette propriété diffère selon que la relation parent-enfant est stockée dans l'objet métier parent ou dans l'enfant. Si un attribut n'est pas une clé étrangère, n'incluez pas ce paramètre dans les informations spécifiques à l'application. Pour plus d'informations, voir «Spécification de la clé étrangère d'un attribut» à la page 64.
UID=AUTO UID=*uid_name*| schema_name.uid_name [=UseIfMissing] UID=CW.*uidcolumnname*[=UseIfMissing]	Le connecteur utilise ce paramètre pour générer l'ID unique de l'objet métier. Si un attribut ne requiert pas la création d'un ID unique, n'incluez pas ce paramètre dans les informations spécifiques à l'application. Pour plus d'informations sur la conservation de l'ID unique au cours du traitement de l'objet métier, voir la description de la propriété PreserveUIDSeq. Pour plus d'informations, voir «Génération d'un ID unique d'objet métier» à la page 67. **Remarque :** CW est un mot-clé utilisé pour représenter le type d'UID et ne représente pas le nom de la table.
PH=true|false	Si PH=true, l'attribut simple correspondant est un attribut d'espace réservé. Un attribut simple est également un espace réservé si les informations spécifiques à l'application qui lui sont associées sont vides ou de valeur null.

Tableau 11. Paramètres nom-valeur dans les informations d'application d'un attribut (suite)

Paramètre	Description
CA=*set_attr_name*| ..set_attr_name	Si set_attr_name est défini par le nom d'un autre attribut dans l'objet métier individuel courant, le connecteur utilise la valeur de l'attribut spécifié pour définir la valeur de cet attribut avant d'ajouter l'objet métier à la base de données au cours d'une opération de création. La valeur du paramètre set_attr_name ne peut pas faire référence à un attribut dans un objet métier enfant mais il peut faire référence à un attribut dans l'objet métier parent si le paramètre set_attr_name est précédé de deux points. Si vous n'incluez pas ce paramètre dans les informations spécifiques à l'application, le connecteur utilise la valeur de l'attribut courant sans copier la valeur de l'attribut (CA) depuis un autre attribut.
OB=[ASC|DESC]	Si une valeur est spécifiée pour ce paramètre et que l'attribut se trouve dans un objet métier enfant, le connecteur utilise la valeur de l'attribut dans la clause ORDER BY des requêtes d'extraction. Le connecteur peut extraire les objets métier enfant dans l'ordre croissant ou décroissant. Utilisez le paramètre ASC pour spécifier une extraction dans l'ordre croissant. Utilisez le paramètre DESC pour spécifier une extraction dans l'ordre décroissant. Si vous n'incluez pas ce paramètre dans les informations spécifiques à l'application, le connecteur n'utilise pas cet attribut lors de la spécification de l'ordre d'extraction.
UNVL=*value*	Indique la valeur utilisée par le connecteur pour représenter une valeur null lors de l'extraction d'un objet métier avec des attributs de valeur null. Si vous n'incluez pas ce paramètre dans les informations spécifiques à l'application, le connecteur insère la valeur CxIgnore pour l'attribut.
ESC=[true|false]	Détermine si le connecteur remplace toutes les instances de chaque caractère identifié dans la propriété ReplaceAllStr par les chaînes de substitution spécifiées également dans la propriété ReplaceStrList. Si ce paramètre ne contient aucune valeur, le connecteur utilise la valeur de la propriété ReplaceStrList pour prendre sa décision. **Remarque :** Le paramètre ESC et les propriétés ReplaceAllStr et ReplaceStrList fournissent une prise en charge de la fonction du caractère d'échappement dans la base de données (par exemple, guillemets simples d'échappement). Dans la mesure où cette même fonction est également disponible dans les instructions préparées fournies par le pilote JDBC, ces propriétés seront déconseillées dans les éditions ultérieures du connecteur. Le connecteur prend actuellement en charge l'utilisation des instructions préparées du pilote JDBC.
FIXEDCHAR=true|false	Indique si la longueur de l'attribut est fixe lorsque les colonnes de la table sont de type CHAR, et non VARCHAR. Par exemple, si un attribut spécifique est lié à une colonne de type CHAR, le connecteur s'attend à ce que la valeur de sa longueur soit FIXEDCHAR. Pour les informations d'application de cet attribut, indiquez FIXEDCHAR=true. Veillez à ce que la propriété MaxLength de l'attribut ait la valeur de longueur CHAR, comme indiqué dans la base de données. Par défaut, FIXEDCHAR=false.
BYTEARRAY=true|false	Si BYTEARRAY=true, le connecteur lira et écrira des données binaires dans la base de données et enverra ces données sous forme de chaîne à InterChange Server ou WebSphere MQ Integrator Broker. BYTEARRAY=false est la valeur par défaut. Pour plus d'informations, voir «Utilisation de données binaires» à la page 68.
USE_LIKE=true|false	Indique si le connecteur compare les chaînes au moyen de l'opérateur = ou LIKE. Si USE_LIKE a la valeur true, les requêtes de caractères génériques peuvent être exécutées en définissant le paramètre WILDCARD_POSITION. Si USE_LIKE a la valeur false, l'opérateur = est utilisé.

Tableau 11. Paramètres nom-valeur dans les informations d'application d'un attribut (suite)

Paramètre	Description
`WILDCARD_POSITION=non-negative number|NONE|BEGIN|END|BOTH`	Si `USE_LIKE` a la valeur true, le paramètre `WILDCARD_POSITION` est utilisé pour préciser la position du caractère générique. Cette valeur peut correspondre à n'importe quel nombre non négatif, `NONE`, `BEGIN`, `END` ou `BOTH`. Par exemple, l'utilisation de la valeur `BEGIN` place le caractère générique en première position dans la chaîne (%chaîne). L'utilisation de la valeur `END` place le caractère générique en dernière position dans la chaîne (chaîne%). L'utilisation de la valeur `BOTH` place les caractères génériques en première et en dernière position dans la chaîne (%chaîne%).
`CLOB=true`	Uniquement applicable aux attributs de type String. Indique si les données de la colonne de base de données correspondant à cet attribut sont de type `CLOB`. **Remarque :** Un type de données `CLOB` est défini comme suit : • L'attribut `CLOB` est de type String et sa longueur est utilisée pour définir la longueur de l'objet CLOB. • L'attribut `CLOB` a la valeur ASI=CN=xyz ; `CLOB=true`. • Tout autre type d'attribut en référence à l'objet `CLOB` dans les informations spécifiques à l'application génère une erreur. • La valeur `CLOB=false` génère une erreur. Un type String classique est identique et sans référence à l'objet `CLOB` dans les informations spécifiques à l'application. Les types de données `CLOB` de 4k ou plus peuvent être insérés ou mis à jour. Cependant, ils ne peuvent être utilisés qu'avec Oracle et requièrent le dernier pilote Thin avec prise en charge de l'objet CLOB. L'utilisation d'un autre pilote peut générer des erreurs.
`TS=true|false`	Pour les attributs de type `DATE`, lorsque le paramètre TS=false est spécifié dans les informations d'application de l'attribut, ce dernier est considéré en tant que type `DATE`. Lorsque le paramètre TS=true, l'attribut est considéré en tant que type `TIMESTAMP`. La valeur par défaut du paramètre TS est true.

Remarque : Si aucune des informations d'application des attributs d'un objet métier n'oblige le connecteur à générer ou à exécuter une requête, le connecteur consigne un message d'avertissement et poursuit le traitement. Il n'émet aucune exception et ne renvoie aucun message d'échec.

Spécification de la clé étrangère d'un attribut : La valeur de cette propriété diffère selon que la relation parent-enfant est stockée dans l'objet métier parent ou dans l'enfant :

- Stockée dans le parent : définissez la valeur pour inclure le type de l'objet métier enfant ainsi que le nom de l'attribut dans l'enfant à utiliser en tant que clé étrangère.
- Stockée dans l'enfant : définissez la valeur pour inclure uniquement le nom de l'attribut dans le parent à utiliser en tant que clé étrangère.

Si la valeur du paramètre `fk_object_name` ne correspond pas au type de l'objet métier enfant et que la valeur du paramètre *`fk_attribute_name`* ne correspond pas au nom de l'attribut dans le parent ou l'enfant (selon le cas), le connecteur ne peut pas traiter cet attribut en tant que clé étrangère. La casse du nom de l'objet métier et de l'attribut a une importance.

Par exemple, supposons que l'objet métier Customer contienne l'attribut `Addr[1]`, qui représente l'objet métier enfant Address, et l'attribut `AID`, qui stocke la clé primaire de l'objet métier enfant en tant que clé étrangère. Dans ce cas, les informations d'application de l'attribut de clé étrangère du parent doivent contenir le type de l'objet métier enfant (Address) ainsi que le nom de son attribut de clé primaire (ID). Dans cet exemple, les informations d'application de l'attribut `AID` doivent inclure `FK=Address.ID`.

Appellation d'un attribut de clé étrangère : Plusieurs objets métier parent peuvent contenir le même objet métier enfant, que l'enfant soit stocké avec un type de cardinalité simple ou multiple et que la relation parent-enfant soit stockée sur le parent ou sur l'enfant. Toutefois, tous les objets métier parent qui stockent la relation parent-enfant doivent utiliser des attributs ayant un nom identique pour contenir la clé primaire de l'enfant. De plus, tous les objets métier enfant qui stockent la relation parent-enfant doivent utiliser des attributs ayant un nom identique pour contenir la clé primaire du parent. La figure 5 illustre ces relations.

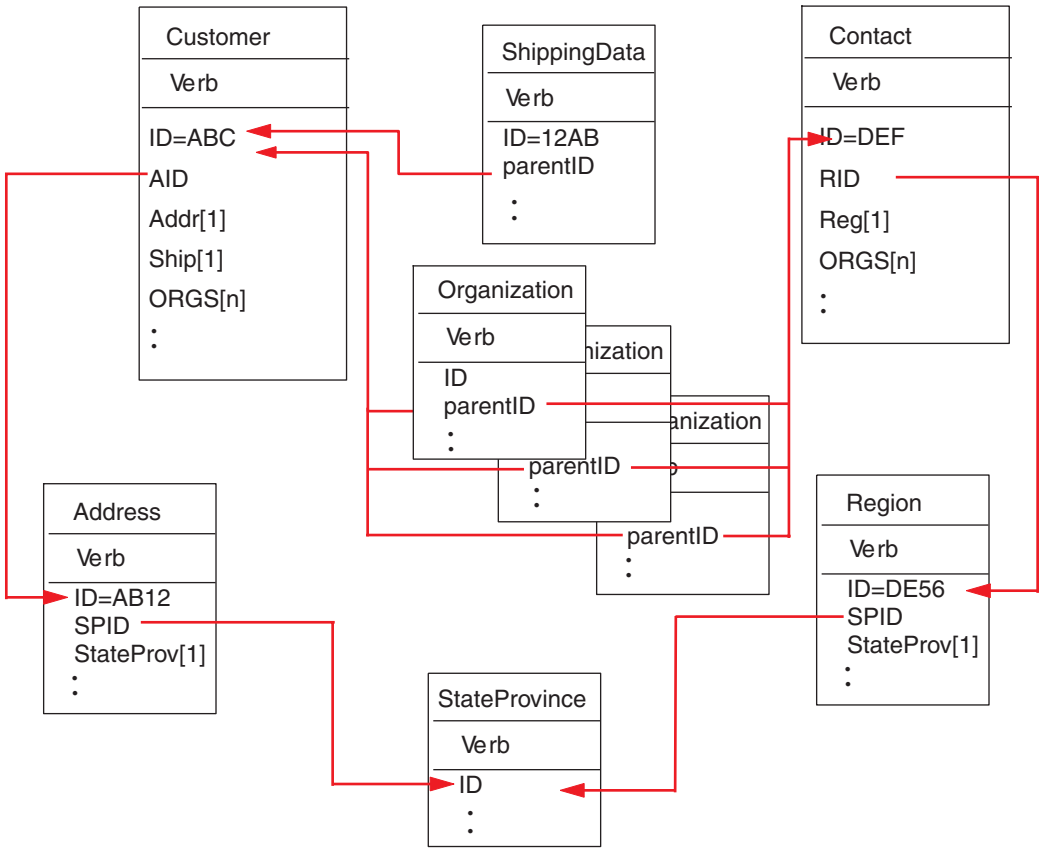


Figure 5. Exemple de relations entre les objets métier

La figure 5 illustre les relations suivantes :

- L'attribut `ORGS[n]` des objets métier Customer ABC et Contact DEF représente un tableau des objets métier Organization. La valeur de clé étrangère de chaque objet métier dans le tableau des objets métiers Organization correspond à la valeur de clé primaire dans l'attribut `ID` des objets métier Customer et Contact. Dans ce cas, chaque objet métier du tableau est contenu par plusieurs parents.

  Les informations d'application de l'attribut `ORGS` peuvent être :

  ```
  KEEP_RELATIONSHIP=true
  ```

Pour plus d'informations sur le paramètre KEEP_RELATIONSHIP, voir «Informations spécifiques à l'application pour les attributs représentant des enfants» à la page 68.

Les informations d'application de l'attribut `parentID` de chaque enfant dans le tableau des objets métiers Organization contiennent le nom de la colonne dans la base de données qui correspond à l'attribut courant et indiquent la clé étrangère de l'attribut courant en détenant le nom de l'attribut de clé primaire du parent. Par exemple :

```
CN=ORG_ID:FK=ID
```

Remarque : Pour permettre à plusieurs objets métier de contenir le même enfant (dans lequel la relation parent-enfant est stockée), tous les objets métier parent doivent utiliser un attribut de même nom pour contenir la clé étrangère de l'enfant. Le paramètre de clé étrangère des informations d'application de cet enfant identifie uniquement le nom de l'attribut et non le type de l'objet métier parent. Le connecteur suppose que le parent direct est le propriétaire de chaque enfant.

- L'attribut `Addr`[1] de l'objet métier Customer représente l'objet métier Address avec des droits de propriété. L'attribut `AID` de l'objet métier Customer identifie la clé primaire de l'objet métier Address en tant que clé étrangère dans le parent. Dans ce cas, l'attribut de clé étrangère du parent doit contenir le type de l'objet métier enfant ainsi que le nom de son attribut de clé primaire. L'enfant de type cardinalité simple, Address, appartient à un seul parent.

  Les informations d'application de l'attribut `Addr` sont :

  ```
  CONTAINMENT=OWNERSHIP
  ```

  Les informations d'application de l'attribut `AID` contiennent le nom de la colonne dans la base de données qui correspond à l'attribut courant et indiquent la clé étrangère de l'attribut courant en détenant le type de l'objet métier enfant ainsi que le nom de son attribut de clé primaire. Par exemple :

  ```
  CN=FK_AD:FK=Address.ID
  ```

  Les informations d'application de l'attribut de clé primaire de l'enfant sont :

  ```
  CN=pk
  ```

- Les attributs `StateProv`[1] des objets métier Address et Region représentent l'objet métier StateProvince sans droits de propriété. Les attributs `SPID` des objets métier Address et Region contiennent le type de l'objet métier enfant (StateProvince) ainsi que le nom de son attribut de clé primaire, utilisé en tant que clé étrangère du parent. Le même enfant de type cardinalité simple, StateProvince, appartient à plusieurs parents.

  Les informations d'application de l'attribut `SPID` sont :

  ```
  CONTAINMENT=NO_OWNERSHIP
  ```

  Pour plus d'informations sur le paramètre CONTAINMENT, voir «Informations spécifiques à l'application pour les attributs représentant des enfants» à la page 68.

  Les informations d'application de l'attribut `SPID` de l'objet métier Address contiennent le nom de la colonne dans la base de données qui correspond à l'attribut courant et indiquent la clé étrangère de l'attribut courant en détenant le type de l'objet métier enfant ainsi que le nom de son attribut de clé primaire. Par exemple :

  ```
  CN=FK_SP:FK=StateProvince.ID
  ```

  Les informations d'application de l'attribut de clé primaire de l'enfant sont :

  ```
  CN=SP_ID
  ```

Remarque : Pour permettre à plusieurs objets métier (qui stockent la relation parent-enfant dans le parent) de contenir le même enfant, tous les objets métier enfant doivent utiliser un attribut de même nom pour contenir la clé étrangère du parent.

- L'attribut `Ship`[1] de l'objet métier Customer représente un objet métier ShippingData qui contient les informations d'expédition du client. L'attribut `ID` de l'objet métier Customer fonctionne en tant que clé étrangère des données d'expédition. Dans ce cas, dans la mesure où l'objet métier ShippingData ne peut pas exister indépendamment de son parent et qu'il est créé uniquement après la création de son parent, la relation parent-enfant est stockée dans l'enfant.

  Les informations d'application de l'attribut `parentID` de l'enfant contiennent le nom de la colonne dans la base de données qui correspond à l'attribut courant et indiquent la clé étrangère de l'attribut courant en détenant le nom de l'attribut de clé primaire de son parent. Par exemple :

  ```
  CN=SD_ID:FK=ID
  ```

Génération d'un ID unique d'objet métier : Le connecteur utilise le paramètre `UID` pour générer l'ID unique de l'objet métier. Le connecteur génère des ID uniques au moyen de séquences (comme Oracle) ou de compteurs (structurés sous forme de tables), puis exécute l'instruction `INSERT`.

Le connecteur utilise une séquence ou un compteur pour générer l'ID puis exécute l'instruction `INSERT` :

- Si le paramètre `UID = uid_name`, la valeur de la variable *uid_name* fournit le nom de la séquence Oracle utilisée par le connecteur pour générer un ID unique pour l'attribut. Une fois la valeur de la séquence extraite, le connecteur renseigne l'attribut de clé et exécute l'instruction `INSERT`.
- Si le paramètre `UID = uid_name=UseIfMissing` et que la valeur de l'attribut n'est pas `CxIgnore`, le connecteur utilise la valeur de l'attribut au lieu de générer un ID unique. Le paramètre `=UseIfMissing` ne peut contenir aucun espace et ne respecte pas les majuscules et minuscules.
- Si le paramètre `UID=CW.uidcolumnname`, le connecteur utilise une table de compteur pour générer un ID unique pour l'attribut. Cette table, dont le nom est configurable, est créée avec une colonne unique appelée `id`. Vous pouvez personnaliser la table pour ajouter une colonne à chaque attribut nécessitant la génération d'un UID. Utilisez le paramètre *uidcolumnname* pour spécifier le nom de la colonne que le connecteur doit utiliser lors de la génération de l'ID unique. Notez que le connecteur prend uniquement en charge les types de données numériques pour les colonnes nécessitant la génération d'un UID.

  Pour plus d'informations sur la configuration d'un nom de table, voir `UniqueIDTableName.` Le script d'installation de cette table est le suivant :

  ```
  \connectors\Manugistics\dependencies\BIA_uid_table_oracle.sql
  ```
- Si le paramètre `UID=CW.uidcolumnname=UseIfMissing` et que la valeur de l'attribut n'est pas `CxIgnore`, le connecteur utilise la valeur de l'attribut au lieu de générer un ID unique. Le paramètre `=UseIfMissing` ne peut contenir aucun espace et ne respecte pas les majuscules et minuscules.

Pour plus d'informations sur la conservation d'une séquence d'ID uniques au cours du traitement, voir la propriété «PreserveUIDSeq» à la page 26.

Informations spécifiques à l'application pour les attributs représentant des enfants

Les attributs qui représentent des objets métier enfant de type cardinalité simple peuvent indiquer si l'enfant appartient au parent ou s'il est partagé entre plusieurs parents.

Les attributs qui représentent un enfant de type cardinalité simple ou un tableau d'objets métier enfant peuvent indiquer le comportement du connecteur lors de la mise à jour du parent ou d'un sous-ensemble d'enfants.

Attributs représentant un objet métier enfant de type cardinalité simple : Le format des informations spécifiques à l'application pour les attributs représentant un objet métier enfant de type cardinalité simple est le suivant :

```
CONTAINMENT= [OWNERSHIP|NO_OWNERSHIP]
```

Affectez au paramètre `CONTAINMENT` la valeur `OWNERSHIP` pour représenter une relation à cardinalité simple dans laquelle le parent est propriétaire de l'objet métier enfant. Affectez au paramètre `CONTAINMENT` la valeur `NO_OWNERSHIP` pour représenter une relation à cardinalité simple dans laquelle le parent partage l'objet métier enfant. N'incluez pas le paramètre `CONTAINMENT` lorsque vous représentez une relation à cardinalité simple qui stocke la relation dans l'enfant plutôt que dans le parent.

Pour plus d'informations, voir «Relations à cardinalité simple et données sans droits de propriété» à la page 37 et «Relations à cardinalité simple qui stockent la relation dans l'enfant» à la page 39.

Attributs représentant un enfant qui stocke la clé du parent : Pour les opérations de mise à jour sur un tableau d'objets métier qui stocke la relation parent-enfant dans l'enfant, il existe une valeur spéciale pour l'attribut représentant l'enfant : vous pouvez affecter à `KEEP_RELATIONSHIP` la valeur `true` pour empêcher le connecteur de supprimer des données enfant existantes qui ne sont pas représentées dans l'objet métier source.

Par exemple, supposons qu'un contrat existant soit associé à un site existant, comme New York. Supposons également que le connecteur reçoive une requête pour mettre à jour un objet métier Contract contenant un objet métier enfant unique auquel San Francisco est associé en tant que site. Si `KEEP_RELATIONSHIP` a la valeur `true` pour l'attribut qui représente les données du site, le connecteur met à jour le contrat pour lui ajouter une association avec San Francisco sans supprimer son association avec New York.

Toutefois, si `KEEP_RELATIONSHIP` a la valeur `false`, le connecteur supprime toutes les données enfant existantes non contenues dans l'objet métier source. Le contrat est alors uniquement associé à San Francisco.

Le format de ces informations spécifiques à l'application est :

```
KEEP_RELATIONSHIP=[true|false]
```

La casse est ignorée dans la vérification de ces informations spécifiques à l'application.

Utilisation de données binaires : Si `BYTEARRAY=true`, le connecteur lira et écrira des données binaires dans la base de données. Dans la mesure où les données binaires ne sont pas prises en charge dans la version actuelle du système WebSphere Business Integration, ces données sont converties en données de type

String puis envoyées au courtier d'intégration. Le format de cette chaîne est un nombre hexadécimal composé de 2 caractères par octet. Par exemple, si les données binaires contenues dans la base de données font 3 octets avec les valeurs décimales (1, 65, 255), la chaîne sera "0141ff".

Format des informations spécifiques à l'application pour les instructions

Le connecteur utilise des informations spécifiques à l'application pour les instructions Retrieve et RetrieveByContent. Ce texte vous permet de spécifier les attributs à inclure dans la clause WHERE lors d'une extraction. Vous pouvez également spécifier des opérateurs et des valeurs d'attribut.

La syntaxe des informations spécifiques à l'application pour les instructions Retrieve et RetrieveByContent est indiquée ci-dessous :

```
[condition_variable conditional_operator @ [...]:[..]attribute_name [, ...]]
```

où :

condition_variable	Nom de la colonne de base de données.
conditonal_operator	Opérateur pris en charge par la base de données, par exemple =, >, OR, AND et IN (*value1*, *value2*).
@	Variable remplacée par la valeur extraite par getAttrValue(*attribute_name*). La substitution est positionnelle. En d'autres termes, le connecteur remplace le premier signe @ par la valeur de la première variable *attribute_name* spécifiée après le délimiteur :.
..	L'attribut spécifié dans la variable attribute_name appartient à l'objet métier parent immédiat. Si cette valeur est manquante, l'attribut appartient à l'objet métier courant.
attribute_name	Nom de l'attribut dont la valeur est utilisée par le connecteur pour remplacer @.

Pour comprendre la syntaxe de cette propriété, supposons qu'un objet métier Item soit doté d'un attribut item_id dont la valeur est XY45 et d'un attribut Color dont la valeur est RED. Supposons également que vous spécifiiez la propriété AppSpecificInfo de l'instruction Retrieve comme suit :

```
Color='RED'
```

La valeur des informations spécifiques à l'application ci-dessus oblige le connecteur à générer la clause WHERE suivante lors d'une extraction :

```
where item_id=XY45 and Color = 'RED'
```

Voici un exemple plus complexe : supposons que l'objet métier Customer dispose de l'attribut customer_id dont la valeur est 1234 et de l'attribut creation_date dont la valeur est 01/01/90. Supposons également que le parent de l'objet métier dispose d'un attribut quantity dont la valeur est 20.

Supposons également que vous spécifiiez la propriété AppSpecificInfo de l'instruction Retrieve comme suit :

```
creation_date > @ OR quantity = @ AND customer_status IN
('GOLD', 'PLATINUM') : creation_date, ..quantity
```

La valeur des informations spécifiques à l'application ci-dessus oblige le connecteur à générer la clause WHERE suivante lors d'une extraction :

```
where customer_id=1234 and creation_date > '01/01/90'
OR quantity = 20 AND customer_status IN ('GOLD', 'PLATINUM')
```

Le connecteur extrait la valeur de date ('01/01/90') de l'attribut creation_date dans l'objet métier courant. Il extrait la valeur de quantité (20) de l'attribut quantity dans l'objet métier parent (comme indiqué par ..quantity dans les informations spécifiques à l'application.

Après avoir analysé les informations spécifiques à l'application pour l'instruction Retrieve, le connecteur ajoute le texte à la clause WHERE de l'instruction RETRIEVE élaborée à partir des clés primaires ou étrangères de l'objet métier. Le connecteur ajoute ensuite l'opérateur AND à la clause WHERE. La valeur des informations spécifiques à l'application doit correspondre à une syntaxe SQL valide. Dans le cas d'une instruction RetrieveByContent, les informations spécifiques à l'application sont ajoutées à la clause WHERE de l'instruction RETRIEVE élaborée à partir des attributs de l'objet métier dont les valeurs sont renseignées.

La clause WHERE peut également faire référence à des attributs d'espace réservé plutôt qu'aux attributs réels dans l'objet métier parent. Ces espaces réservés ne possèdent aucune information spécifique à l'application. Un attribut peut être un espace réservé s'il satisfait l'une des conditions suivantes pour ses informations spécifiques à l'application :

1. attribut simple avec ASI=null ou '' ;
2. attribut simple avec ASI=PH=TRUE.

Par exemple, un objet métier Order contient un objet métier de ligne d'articles de type cardinalité multiple et seule l'extraction de lignes d'articles spécifiques est nécessaire. Cette extraction peut être réalisée via un attribut d'espace réservé dans l'objet métier Order. Cet espace réservé est requis dans l'objet parent car les objets enfant ont tous été élagués. L'attribut d'espace réservé peut être renseigné au moment de l'exécution par le courtier d'intégration avec une liste de lignes d'articles spécifiques, séparées par une virgule (,).

Pour cet exemple, vous pouvez ajouter les informations suivantes à la clause WHERE pour l'instruction Retrieve sur l'objet métier enfant des lignes d'articles :

```
line_item_id in(@,@,@):..placeholder1,..placeholder2,..placeholder3
```

où line_item_id désigne l'ID dans l'objet métier enfant et placeholder, l'attribut dans le parent. Si placeholder contient les valeurs 12,13,14, la requête sélectionnera les éléments suivants dans la clause WHERE :

```
line_item_id in(12,13,14)
```

où SELECT:..FROM:..WHERE *x* in (1,2,3) désigne une syntaxe SQL de base de données standard.

Dans l'instruction RetrieveByContent, si la longueur de la clause WHERE est 0, le connecteur utilisera les informations spécifiques à l'application dans la clause WHERE de l'instruction RETRIEVE. Grâce à cette fonction, l'utilisateur peut envoyer un objet métier sans valeur d'attribut indiquée et spécifier les informations spécifiques à l'application pour l'instruction RetrieveByContent. Le connecteur générera alors la clause WHERE en fonction des éléments spécifiés dans les informations d'application de l'instruction uniquement.

Chapitre 4. Création de définitions d'objets métier à l'aide d'IBM ODA for Manugistics

Ce chapitre présente le composant IBM ODA for Manugistics, un agent ODA (Object Discovery Agent), qui crée des définitions d'objets métier pour le connecteur pour JDBC. Etant donné que le connecteur utilise des objets basés sur des tables ou sur des vues, le composant ODA for Manugistics utilise des tables de base de données, des vues, des procédures stockées et des synonymes/alias pour rechercher les exigences des objets métier spécifiques à sa source de données JDBC.

Remarque : Etre familiarisé avec les bases de données et les pilotes JDBC (à des fins de configuration) peut aider à comprendre le fonctionnement du composant ODA for Manugistics.

Ce chapitre contient les sections suivantes :

Installation et utilisation

Cette section aborde les points suivants :

Installation d'ODA for Manugistics

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

- `ODA\Manugistics\BIA_ManugisticsODA.jar`
- `ODA\messages\BIA_ManugisticsODAAgent.txt`
- `ODA\messages\BIA_ManugisticsODAAgent_ll_TT.txt` (fichiers de messages spécifiques à une langue (*ll*) et à un pays ou un territoire (*TT*).
- `ODA\Manugistics\start_ManugisticsODA.bat` (Windows uniquement)
- `ODA/Manugistics/start_ManugisticsODA.sh` (UNIX uniquement)
- `bin\CWODAEnv.bat` (Windows uniquement)
- `bin/CWODAEnv.sh` (UNIX uniquement)

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

Avant d'utiliser ODA for Manugistics

Pour pouvoir exécuter ODA for Manugistics, vous devez effectuer les opérations suivantes :

1. Installez le pilote JDBC correspondant. Suivez les instructions contenues dans le document *System Installation Guide for UNIX* ou *for Windows*.

   Important : ODA for Manugistics peut être connecté à n'importe quelle base de données utilisant un pilote JDBC qui prend en charge JDBC 2.0 ou suivant.

2. Etant donné qu'ODA for Manugistics crée des noms d'objet métier et des noms d'attribut à partir des noms de tables de base de données et des colonnes correspondantes, et que ces noms doivent être au format ISO Latin-1, vérifiez que les noms des composants de la base de données sont au format Latin-1. Dans le cas contraire, vous avez le choix entre les opérations suivantes :
   - Créez la définition de l'objet métier manuellement dans Business Object Designer.
   - Modifiez la définition créée par ODA for Manugistics de sorte que tous les noms des objets métier et des attributs soient au format Latin-1.
3. Modifiez le shell UNIX ou le fichier de traitement par lots Windows et configurez les valeurs décrites dans le tableau 12.

Tableau 12. Variables de configuration du shell et du fichier de traitement par lots

Variable	Explication	Exemple
AGENTNAME	Nom de l'ODA	**UNIX :** AGENTNAME=ManugisticsODA **Windows :** set AGENTNAME=ManugisticsODA
AGENT	Nom du fichier jar de l'ODA	**UNIX :** AGENT=$CROSSWORLDS/ODA/Manugistics/ BIA_ManugisticsODA.jar **Windows :** set AGENT= %CROSSWORLDS%\ODA\Manugistics\ BIA_ManugisticsODA.jar
DRIVERPATH	Chemin d'accès à la bibliothèque du pilote JDBC. ODA for Manugistics utilise les classes du pilote pour établir une connexion à une base de données spécifique	**UNIX :** DRIVERPATH=$CROSSWORLDS/lib/ \ xwutil.jar:$CROSSWORLDS/lib/ \ xwbase.jar:$CROSSWORLDS/lib/ \ xwsqlserver.jar:$CROSSWORLDS/lib/ \ spy/lib/spy.jar **Windows :** set DRIVERPATH=%CROSSWORLDS%\ / lib\xwutil.jar;%CROSSWORLDS%\lib\ / xwbase.jar;%CROSSWORLDS%\lib\ / xwsqlserver.jar;%CROSSWORLDS%\lib\ / spy\lib\spy.jar

Tableau 12. Variables de configuration du shell et du fichier de traitement par lots (suite)

Variable	Explication	Exemple
DRIVERLIB	Chemin d'accès aux bibliothèques natives utilisées par le pilote JDBC	**UNIX :** DRIVERLIB=$CROSSWORLDS/lib/db2jdbc.so **Windows :** DRIVERLIB=%CROSSWORLDS%\bin\db2jdbc.dll

Une fois que vous avez installé le pilote JDBC et défini les valeurs de configuration dans le shell ou le fichier de traitement par lots, vous devez procéder comme suit pour créer les objets métier :

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

Les sections suivantes décrivent chaque étape en détail.

Lancement d'ODA for Manugistics

Vous pouvez lancer ODA for Manugistics à l'aide du script de démarrage correspondant au système d'exploitation.

UNIX :

```
start_ManugisticsODA.sh
```

Windows :

```
start_ManugisticsODA.bat
```

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

Exécution d'instances multiples d'ODA for Manugistics

Il est recommandé de changer le nom de l'ODA lorsque vous exécutez plusieurs instances de ce composant. Pour créer d'autres instances d'ODA for Manugistics portant chacune un nom unique :

- Créez un script ou un fichier de traitement par lots pour chaque instance.
- Indiquez un nom unique dans la variable AGENTNAME de chaque script ou chaque fichier de traitement par lots.

Il est recommandé d'indiquer un préfixe à chaque nom de machine hôte lorsque vous exécutez des instances ODA sur différentes machines.

La figure 6 à la page 76 illustre la fenêtre dans Business Object Designer à partir de laquelle vous sélectionnez l'ODA à exécuter.

Utilisation des fichiers de messages d'erreur et de trace

Les fichiers de messages d'erreur et de trace (la valeur par défaut est `BIA_ManugisticsODAAgent.txt`) résident dans le répertoire `\ODA\messages\`, situé sous le répertoire produit. Ces fichiers utilisent la convention d'appellation suivante :

AgentName`Agent.txt`

Si vous créez plusieurs instances du script ou du fichier de traitement par lots de l'ODA et que vous indiquez un nom unique pour chaque ODA représenté, vous pouvez avoir un fichier de messages pour chaque instance d'ODA. Des noms différents d'ODA peuvent également utiliser le même fichier de messages. Il existe deux manières d'indiquer un fichier de messages correct :

- Si vous modifiez le nom d'un ODA et que vous ne créez pas de fichier de messages associé, vous devez changer le nom du fichier de messages dans Business Object Designer dans le cadre de la configuration de l'ODA. Business Object Designer fournit un nom pour le fichier de messages mais ne crée pas le fichier. Si le fichier affiché dans le cadre de la configuration de l'ODA n'existe pas, changez la valeur afin qu'elle désigne un fichier existant.
- Vous pouvez copier le fichier de messages existant associé à un ODA et le modifier, le cas échéant. Business Object Designer présume que vous nommez chaque fichier conformément à la convention d'appellation. Par exemple, si la variable AGENTNAME indique `ManugisticsODA1`, l'outil présume que le nom du fichier de messages associé est `ManugisticsODA1Agent.txt`. Par conséquent, lorsque Business Object Designer fournit le nom de fichier à des fins de vérification pendant la configuration de l'ODA, le nom du fichier est basé sur le nom de l'ODA. Vérifiez que le fichier de messages par défaut est correctement nommé, et corrigez-le si nécessaire.

Important : Si vous n'indiquez pas de nom pour le fichier de messages lorsque vous configurez l'ODA, ce dernier s'exécute sans messages. Pour plus d'informations sur l'indication du nom du fichier de messages, voir «Configuration des propriétés d'initialisation» à la page 76.

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

- le nom du fichier dans lequel le composant ODA for Manugistics enregistre les messages d'erreurs et les informations de trace ;
- le niveau de suivi, compris entre `0` et `5`.

Le tableau 13 décrit ces valeurs.

Tableau 13. Niveaux de suivi

Niveau de trace	Description
0	Consigne toutes les erreurs
1	Effectue un suivi de tous les messages d'entrée et de sortie pour la méthode
2	Effectue un suivi des propriétés de l'ODA et de leurs valeurs
3	Trace le nom de tous les objets métier
4	Trace le détail des unités d'exécution générées
5	• Indique les valeurs d'initialisation de l'ODA pour toutes ses propriétés • Trace un état détaillé de chaque unité d'exécution générée par ODA for Manugistics • Trace le cliché de la définition de l'objet métier

Pour plus d'informations sur l'emplacement dans lequel vous voulez configurer ces valeurs, voir «Configuration des propriétés d'initialisation» à la page 76.

Utilisation d'ODA for Manugistics dans Business Object Designer

Cette section explique comment utiliser ODA for Manugistics dans Business Object Designer pour créer des définitions d'objet métier. Pour plus d'informations sur l'exécution de Business Object Designer, voir le document *Business Object Development Guide*. Vous pouvez télécharger ce document sur l'InfoCenter d'IBM WebSphere Business Integration Adapters : http://www.ibm.com/websphere/integration/wbiadapters/infocenter

Une fois que vous avez lancé un ODA, vous devez démarrer Business Object Designer afin de le configurer et l'exécuter. Dans Business Object Designer, la création d'une définition d'objet métier à l'aide d'un ODA comprend six étapes. Business Object Designer propose un assistant pour vous guider à travers ces étapes.

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

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

   Business Object Designer affiche la première fenêtre de l'assistant, Select Agent. La figure 6 à la page 76 représente cette fenêtre.

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

1. «Sélection de l'ODA»
2. «Configuration des propriétés d'initialisation» à la page 76
3. «Développement des noeuds et sélection des objets de la base de données» à la page 78
4. «Confirmation des sélections d'objets de base de données» à la page 80
5. «Création des définitions» à la page 81 et, le cas échéant, «Informations supplémentaires» à la page 81
6. «Enregistrement des définitions» à la page 85

Sélection de l'ODA

La figure 6 représente la première boîte de dialogue de l'assistant de Business Object Designer qui comprend six étapes. Dans cette fenêtre, sélectionnez l'ODA à exécuter.

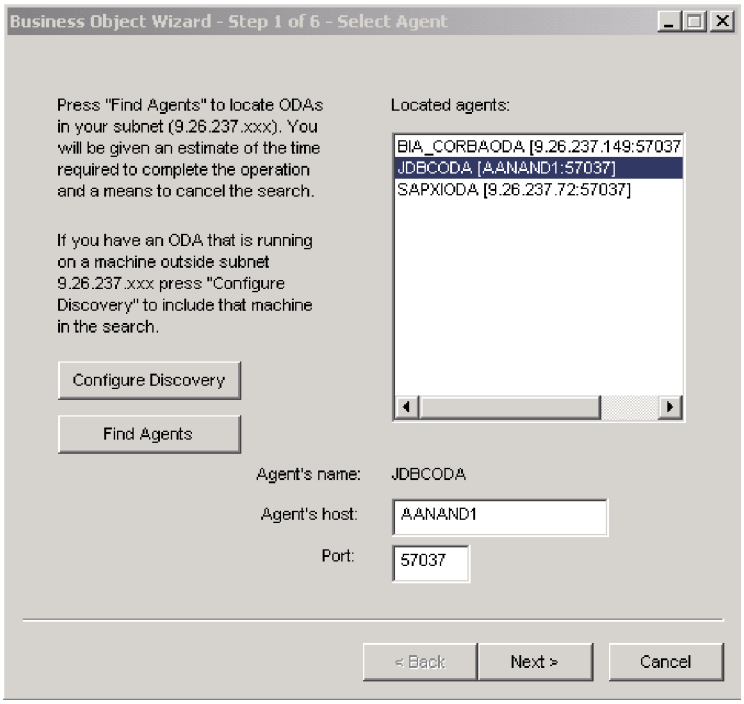


Figure 6. Sélection de l'ODA

Pour sélectionner l'ODA :

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

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

   Business Object Designer affiche votre sélection dans la zone Agent's name.
3. Cliquez sur Next.

Configuration des propriétés d'initialisation

La première fois que Business Object Designer communique avec ODA for Manugistics, il vous invite à entrer les propriétés d'initialisation, comme l'illustre la figure 7. Vous pouvez enregistrer ces propriétés dans un profil nommé de sorte que vous n'ayez plus besoin de les entrer chaque fois que vous utilisez ODA for Manugistics. Pour plus d'informations sur l'indication d'un profil ODA, voir le document *Business Object Development Guide*.

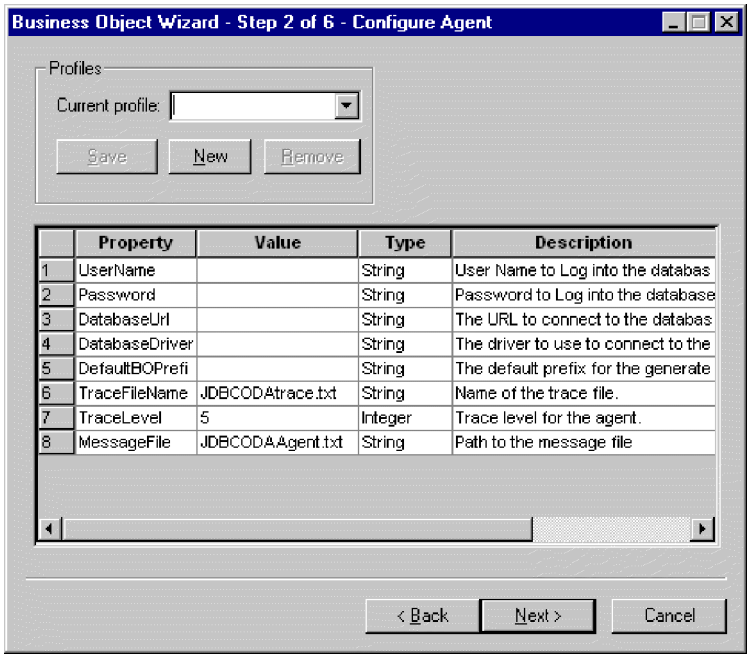


Figure 7. Configuration des propriétés d'initialisation de l'agent

Configurez les propriétés d'ODA for Manugistics, décrites dans le tableau 14.

Tableau 14. Propriétés d'ODA for Manugistics

Numéro de ligne	Nom de la propriété	Type de propriété	Description
1	`UserName`	String	Nom de l'utilisateur qui dispose des autorisations pour se connecter à la base de données.
2	`Password`	String	Mot de passe de l'utilisateur qui dispose des autorisations pour se connecter à la base de données.
3	`DatabaseUrl`	String	URL qui permet de se connecter à la base de données. Par exemple : `jdbc:oracle:thin:@MACHINENAME:1521:SIDNAME`
4	`DatabaseDriver`	String	Nom du pilote utilisé pour établir la connexion. Par exemple : `oracle.jdbc.driver.OracleDriver`
5	`DefaultBOPrefix`	String	Texte préajouté au nom de l'objet métier pour le rendre unique. Vous pouvez modifier le nom par la suite, lorsque Business Object Designer vous demande d'entrer les propriétés de l'objet métier. Pour plus d'informations, voir «Informations supplémentaires» à la page 81.
6	`TraceFileName`	String	Nom du fichier dans lequel le composant ODA for Manugistics enregistre les informations de trace. Si le fichier n'existe pas, ODA for Manugistics le crée dans le répertoire `\ODA\Manugistics`. Si le fichier existe déjà, ODA for Manugistics y ajoute le nom du fichier. ODA for Manugistics nomme le fichier conformément à la convention d'appellation. Par exemple, si l'agent est nommé `ManugisticsODA`, il crée un fichier trace nommé `ManugisticsODAtrace.txt`. Utilisez cette propriété pour indiquer un autre nom pour ce fichier.
7	`TraceLevel`	Integer	Niveau de suivi défini pour ODA for Manugistics. Voir tableau 13 à la page 74.

Tableau 14. Propriétés d'ODA for Manugistics (suite)

Numéro de ligne	Nom de la propriété	Type de propriété	Description
8	`MessageFile`	String	Nom du fichier de messages et d'erreurs. ODA for Manugistics affiche le nom du fichier conformément à la convention d'appellation. Par exemple, si l'agent s'appelle `ManugisticsODA`, la valeur de la propriété du fichier de messages se présente sous la forme `ManugisticsODAAgent.txt`. **Important** : Le fichier de messages et d'erreurs doit résider dans le répertoire `\ODA\messages`. Utilisez cette propriété pour vérifier ou indiquer un fichier existant.

Important

Corrigez le nom du fichier de messages si la valeur par défaut affichée dans Business Object Designer correspond à un fichier qui n'existe pas. Si le nom est incorrect lorsque vous passez à la fenêtre suivante, Business Object Designer affiche un message d'erreur dans la fenêtre dans laquelle l'ODA a été lancé. Le message n'apparaît pas dans Business Object Designer. Si vous n'indiquez pas un fichier de messages correct, l'ODA s'exécute sans message.

Développement des noeuds et sélection des objets de la base de données

Une fois les propriétés d'initialisation configurées pour ODA for Manugistics, Business Object Designer se connecte à la base de données spécifiée et affiche une arborescence contenant tous les noms de schéma existant dans la base de données. Ces noms, représentés par des noeuds dans l'arborescence, peuvent être développés. Cliquez sur l'un deux pour afficher toutes les tables, les vues, les procédures stockées et les synonymes/alias contenus dans chaque schéma. La figure 8 représente cette boîte de dialogue avec certains schémas développés.

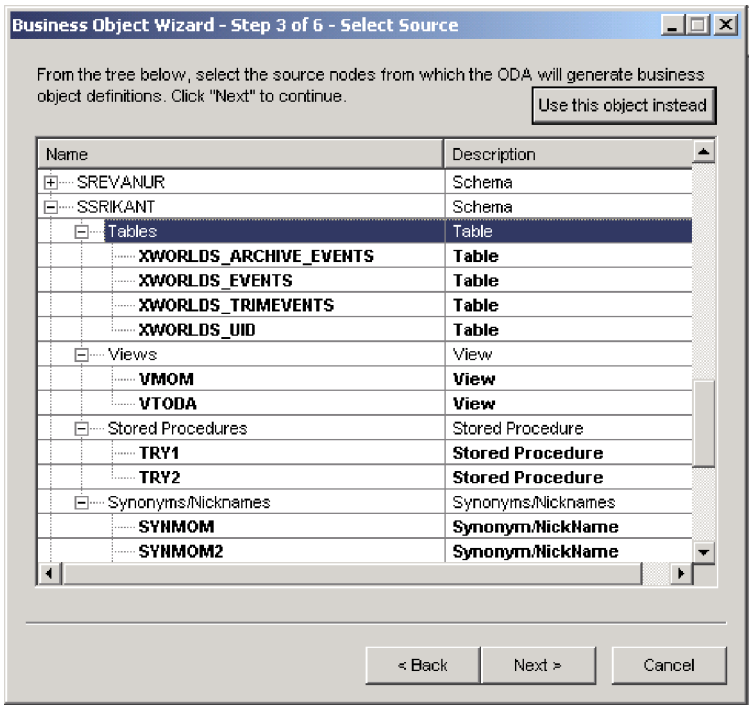


Figure 8. Arborescence des schémas avec les noeuds développés

Pour identifier tous les objets de base de données qui stockent les données de la définition de l'objet métier créée, sélectionnez l'ensemble des tables, vues, procédures stockées et synonymes/alias, puis cliquez sur Next. Pour plus d'informations sur le filtrage des objets renvoyés, voir le document *Business Object Development Guide*.

Le nom de schéma ALL SCHEMAS est utilisé pour faciliter l'extraction des objets comme les tables et les vues des bases de données qui n'ont pas de schéma associé aux objets. Si vous développez ALL SCHEMAS, une arborescence contenant les tables, vues, procédures stockées et les synonymes/alias s'affiche. Lorsque tous les noeuds sont développés, tous les objets correspondants de la base de données s'affichent, quel que soit le schéma auquel l'objet appartient.

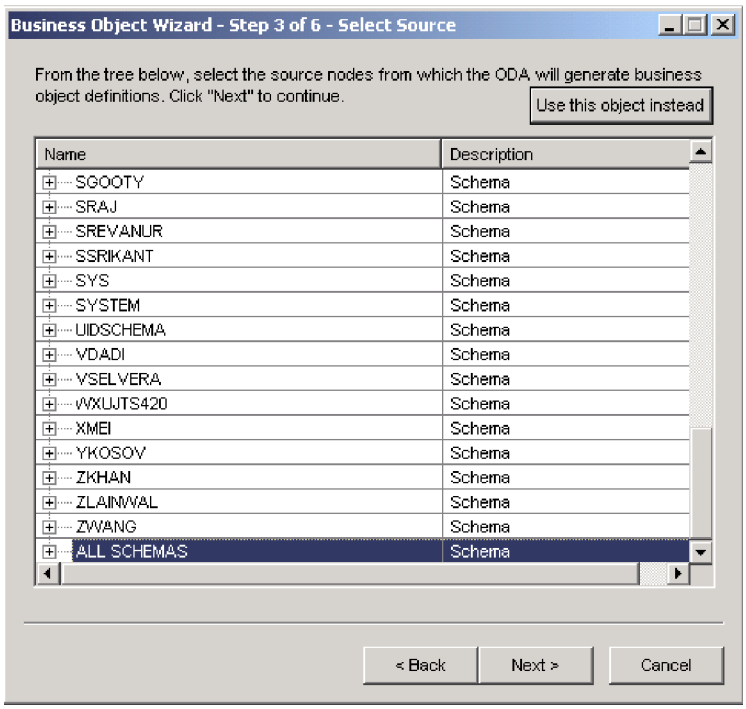


Figure 9. Développement du noeud ALL SCHEMAS pour extraire les objets

Confirmation des sélections d'objets de base de données

Après avoir identifié tous les objets de base de données à associer à la définition de l'objet métier créée, Business Object Designer affiche la boîte de dialogue qui contient uniquement les tables, les vues, les procédures stockées et les synonymes/alias. La figure 10 représente cette boîte de dialogue.

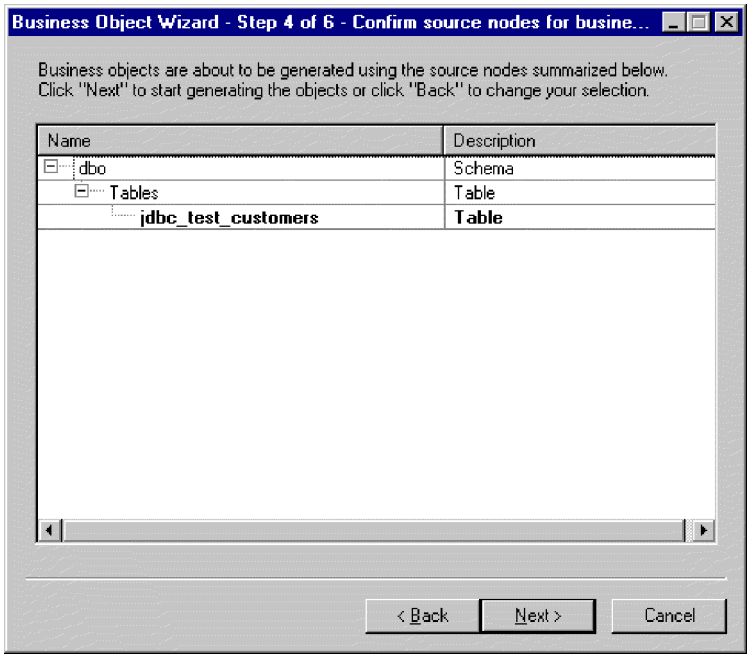


Figure 10. Confirmation de la sélection des objets de base de données

Cette fenêtre contient les options suivantes :

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

Création des définitions

Lorsque vous avez confirmé la sélection des objets de la base de données, une boîte de dialogue vous informe que Business Object Designer procède à la création des définitions.

Informations supplémentaires

Si le composant ODA for Manugistics a besoin d'informations supplémentaires, Business Object Designer affiche la fenêtre BO Properties, qui vous permet d'entrer ces informations.

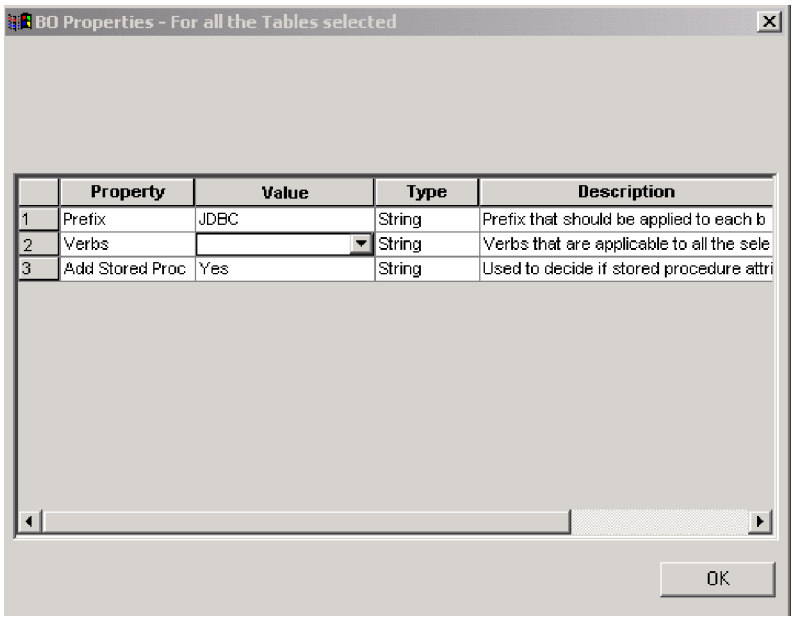


Figure 11. Informations supplémentaires sur les objets de base de données

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

- *Prefix* — Texte préajouté au nom de l'objet métier pour le rendre unique. Si vous êtes satisfait de la valeur que vous avez indiquée pour la propriété *DefaultBOPrefix* dans la fenêtre Configure Agent (figure 7), vous n'avez pas besoin de changer cette valeur.
- *Verbs* — Cliquez dans la zone *Value* et sélectionnez une ou plusieurs instructions dans le menu instantané. Il s'agit d'instructions prises en charge par l'objet métier.
- *Add Stored Procedure* — Cliquez sur `Yes` ou `No` dans la zone Value :
  - Si vous sélectionnez `Yes`, puis cliquez sur OK, ODA for Manugistics affiche une fenêtre contenant la liste des attributs des procédures stockées. Sélectionnez les attributs de procédure stockée que vous voulez ajouter à l'objet métier.
  - Si vous sélectionnez `No`, aucun attribut de procédure stockée n'est ajouté à la définition de l'objet métier.

  La valeur par défaut est `Yes`.

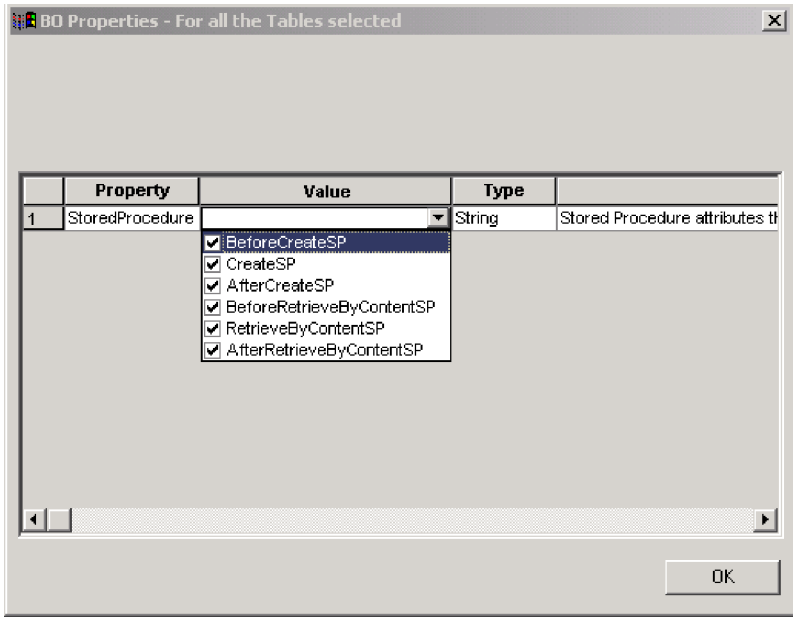


Figure 12. Sélection des attributs de procédure stockée

Remarque : Si une zone dans la boîte de dialogue BO Properties contient plusieurs valeurs, la zone apparaît vide lorsque la boîte de dialogue s'affiche pour la première fois. Cliquez dans la zone pour afficher une liste déroulante contenant ses valeurs.

Les attributs de la procédure stockée à ajouter à l'objet métier peuvent être associés à l'une des procédures stockées dans la base de données de ce schéma. Vous pouvez choisir une procédure stockée dans une liste déroulante contenant toutes les procédures stockées de la base de données de ce schéma, en fonction de chaque attribut de procédure stockée. Ces informations créeront les informations spécifiques à l'application relatives à cet attribut.

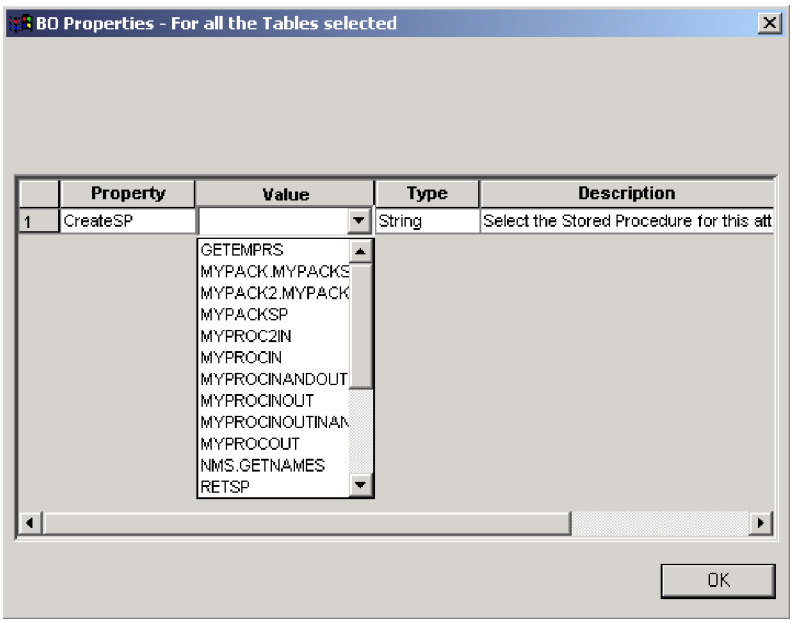


Figure 13. Association des procédures stockées aux attributs de procédure stockée

Les informations spécifiques à l'application relatives à l'objet se présenteront sous la forme `TN=tableName`.

Au niveau de l'attribut, les informations spécifiques à l'application se présenteront sous la forme `CN=ColumnName`.

Si un objet métier est créé à partir d'une procédure stockée, et que les attributs de procédure stockée d'ODA for Manugistics Adapter (comme SPForCreate) y sont associés, l'agent ODA fournit la liste de tous les noms de procédure stockée de ce schéma en fonction des attributs de procédure stockée. Cet ODA permet d'associer la procédure stockée requise à l'objet métier. Cela permet de créer les informations spécifiques à l'application pour l'attribut de la procédure stockée du composant ODA for Manugistics Adapter, comme suit :

```
SPN=stored procedure Name; IN=a1:a2; OUT=b1:b2; IO=c1:c2
```

où IN signifie que le paramètre de la procédure stockée est de type INPUT, OUT signifie que le paramètre est de type OUTPUT et IO de type INPUT/OUTPUT. L'ODA n'affecte pas la valeur "true" ou "false" au paramètre RS dans les informations spécifiques à l'application. Par conséquent, vous devez le faire manuellement.

Les instructions ajoutées à l'objet métier sont les instructions standard, telles que Retrieve, RetrieveByContent, Create, Update et Delete.

Si le paramètre de retour de la procédure stockée est de type ResultSet, l'agent ODA analyse le jeu de résultats et crée un objet métier, faisant ainsi des colonnes du jeu de résultats les attributs de l'objet métier. Les informations spécifiques à l'application pour les colonnes de la procédure stockée auront la valeur `CN=StoredProcedureColumnName`. L'ODA définit les attributs de clé basés sur les métadonnées JDBC renvoyées par le pilote. Si aucune valeur n'est renvoyée, l'ODA n'identifie pas les attributs par défaut comme des clés. Tous les autres attributs, comme les attributs de longueur et de type, sont définis comme pour les attributs créés à partir des tables.

Enregistrement des définitions

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

Contenu de la définition créée

La définition de l'objet métier que l'agent ODA for Manugistics crée contient les informations suivantes :

- un attribut pour chaque colonne dans les tables de base de données, les vues, les procédures stockées et les synonymes/alias indiqués ;
- les instructions figurant dans la fenêtre BO Properties ;
- les informations spécifiques à l'application :
  - sur l'objet métier
  - pour chaque attribut
  - pour chaque instruction

Cette section aborde les points suivants :

- «Propriétés de l'objet métier»
- «Propriétés des attributs» à la page 86
- «Instructions» à la page 88

Propriétés de l'objet métier

ODA for Manugistics génère les informations suivantes sur l'objet métier :

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

Les informations spécifiques à l'application relatives à l'objet métier vous permettent d'effectuer les opérations suivantes :

- indiquer le nom de la table de la base de données correspondante ;
- fournir les informations nécessaires à la réalisation d'une suppression logique ou physique.

Au niveau de l'objet métier, les informations spécifiques à l'application comportent des paramètres séparés par un point-virgule (;). Le nom du paramètre et la valeur associée sont séparés par un signe deux-points (:). La syntaxe est la suivante :

`TN=`*`TableName`*`; SCN=`*`StatusColumnName`*`:`*`StatusValue`*

où `TableName` désigne une table de la base de données, *`StatusColumnName`* correspond au nom de la colonne de la base de données utilisée pour effectuer les suppressions logiques, et *`StatusValue`* correspond à la valeur qui indique si un objet métier est inactif ou supprimé.

Le paramètre `AppSpecificInfo` généré par ODA for Manugistics à ce niveau contient une valeur uniquement pour le nom de la table de base de données, la vue, la procédure stockée ou le synonyme/alias. Pour plus d'informations sur

l'indication d'une valeur pour la colonne correspondant à l'état, voir «Informations spécifiques à l'application au niveau de l'objet métier» à la page 60.

Propriétés des attributs

Cette section décrit les propriétés générées par ODA for Manugistics pour chaque attribut. Pour plus d'informations sur les attributs, voir «Propriétés des attributs d'objet métier» à la page 57.

Propriété Name

ODA for Manugistics extrait la valeur du nom de l'attribut du nom de la colonne dans la table de base de données, la vue, la procédure stockée ou le synonyme/alias.

Propriété Data type

Lorsque le composant ODA for Manugistics définit le type d'un attribut, il convertit le type de données d'une colonne dans la table ou la vue en type d'objet métier IBM WebSphere Business Integration Adapter correspondant. Cette conversion a lieu en deux étapes. Tout d'abord, le type de données dans la base de données est converti en type de données JDBC. Ensuite, le type de données JDBC est converti en type d'objet métier IBM WebSphere Business Integration Adapter. La première conversion est effectuée par le pilote JDBC que vous utilisez. Pour plus d'informations sur le mappage d'un type de base de données en type de données JDBC, voir les spécifications JDBC (2.0 et suivant). Le tableau 14 représente la conversion du type JDBC en objet métier IBM WebSphere Business Integration Adapter.

Tableau 15. Correspondance des types de données

Type JDBC	Objet métier WebSphere Business Integration Adapter
BIT	BOOLEAN
CHAR	STRING
VARCHAR	STRING
LONGVARCHAR	STRING
INTEGER	INTEGER
NUMERIC	INTEGER
SMALLINT	INTEGER
TINYINT	INTEGER
BIGINT	INTEGER
DATE	DATE
TIME	DATE
TIMESTAMP	DATE
DECIMAL	STRING
DOUBLE	DOUBLE
FLOAT	DOUBLE
REAL	FLOAT

Tableau 15. Correspondance des types de données (suite)

Type JDBC	Objet métier WebSphere Business Integration Adapter
BINARY	STRING, ajouter BYTEARRAY=TRUE à AppSpecificInfo
VARBINARY	STRING, ajouter BYTEARRAY=TRUE à AppSpecificInfo

Remarque : Si le type de données d'une colonne ne fait pas partie des types indiqués dans le tableau 15 à la page 86, ODA for Manugistics ignore la colonne et affiche un message indiquant que la colonne ne peut pas être traitée.

Propriété Cardinality

ODA for Manugistics affecte à la propriété de cardinalité de tous les attributs la valeur 1.

Propriété MaxLength

ODA for Manugistics extrait la longueur d'une chaîne à partir de la longueur indiquée pour le type de données `varchar`, `char` ou `text`.

Propriété IsKey

Si la colonne est une clé primaire dans la table, ODA for Manugistics la désigne comme un attribut clé. Toutefois, si une vue, une procédure stockée ou un synonyme/alias, au lieu d'une table, est sélectionné comme noeud source pour créer des objets métier, ODA for Manugistics ne marque pas la colonne comme étant un attribut clé. Dans ce cas, l'attribut clé doit être défini manuellement.

Propriété IsForeignKey

ODA for Manugistics ne définit pas la propriété `IsForeignKey`. Vous pouvez la définir dans Business Object Designer.

Propriété IsRequired

Si une zone est désignée par la valeur `not null` dans la table, la vue, la procédure stockée ou le synonyme/alias, ODA for Manugistics la marque comme étant `un` attribut obligatoire. Toutefois, ODA for Manugistics ne marque pas la zone clé comme étant obligatoire car une séquence peut lui être associée, ou elle peut constituer une colonne d'identité.

Propriété AppSpecificInfo

ODA for Manugistics comprend deux paramètres pour la propriété `AppSpecificInfo` au niveau de l'attribut. La syntaxe des paramètres indiqués est la suivante :

- `CN=`*`ColumnName`*

où ColumnName correspond au nom de la colonne dans la table de la base de données, la vue, la procédure stockée ou le synonyme/alias associé à l'attribut.

- BYTEARRAY=*true*|*false*

  ODA for Manugistics identifie les colonnes contenant des données binaires et crée un attribut de type String avec une propriété AppSpecificInfo de type BYTEARRAY=true.

Remarque : Vous pouvez définir d'autres paramètres AppSpecificInfo dans Business Object Designer. Pour plus d'informations sur ces paramètres, voir «Informations spécifiques à l'application au niveau de l'attribut» à la page 61.

Instructions

ODA for Manugistics génère les instructions indiquées dans la fenêtre BO Properties. Il crée une propriété AppSpecificInfo pour chaque instruction mais n'indique pas de valeur. Pour plus d'informations, voir «Format des informations spécifiques à l'application pour les instructions» à la page 69.

Exemple de fichier de définition d'objet métier

La sortie suivante représente un exemple de définition d'objet métier :

```
[BusinessObjectDefinition]
Name = CUSTOMER
Version = 1.0.0
AppSpecificInfo = TN=ra_customers;SCN=

            [Attribute]
               Name = customer_id
               Type = Integer
               Cardinality = 1
               MaxLength = 0
               IsKey = true
               IsForeignKey = false
               IsRequired = false
               AppSpecificInfo = CN=customer_id
               DefaultValue =
               [End]

               *******Other attributes *********

[Attribute]
               Name = ObjectEventId
               Type = String
               Cardinality = 1
               MaxLength = 0
               IsKey = false
               IsForeignKey = false
               IsRequired = false
               AppSpecificInfo =
               DefaultValue =
               [End]

               [Verb]
               Name = Delete
               AppSpecificInfo =
               [End]

               [Verb]
               Name = Update
               AppSpecificInfo =
               [End]
```

```
                [Verb]
                Name = Create
                AppSpecificInfo =
                [End]

                [Verb]
                Name = Retrieve
                AppSpecificInfo =
                [End]
[End]
```

Insertion d'attributs contenant des objets métier enfant

Utilisez Business Object Designer pour insérer des attributs qui représentent des objets métier de type cardinalité simple ou multiple. Pour plus d'informations, voir le document *Business Object Development Guide*.

Ajout d'informations à une définition d'objet métier

Dans la mesure où les tables de base de données, les vues, les procédures stockées et les synonymes/alias sont susceptibles de ne pas contenir toutes les informations que requiert une définition d'objet métier, il peut s'avérer nécessaire d'ajouter des informations à la définition de l'objet métier créée par ODA for Manugistics. Pour plus d'informations, voir Chapitre 3, «Présentation des objets métier pour le connecteur», à la page 35.

Pour examiner la définition de l'objet métier ou ajouter des informations, vous pouvez utiliser Business Object Designer ou un éditeur de texte. Pour recharger une définition que vous avez revue dans le référentiel d'IBM WebSphere Business Integration Adapter, vous pouvez utiliser Business Object Designer ou la commande `repos_copy` si InterChange Server est le courtier d'intégration.

Chapitre 5. Identification et résolution des erreurs

Le présent chapitre décrit les problèmes que vous êtes susceptible de rencontrer lorsque vous démarrez ou exécutez le connecteur JDBC. Il contient les sections suivantes :

Problèmes de démarrage

Si vous avez des difficultés à démarrer le connecteur, vérifiez que le courtier d'intégration est actif.

Traitement des événements

Si la table des événements contient des événements, et que ceux-ci ne sont pas traités pendant l'exécution du connecteur, vérifiez les éléments suivants :

- Le processus métier correspondant est exécuté.
- Le nom de l'objet métier dans la table des événements correspond à celui de l'objet métier indiqué pour le port du processus métier.

Mappage (InterChange Server Integration Broker uniquement)

Cette section aborde les points suivants :

Problèmes de mappage

Si les objets métier ne sont pas mappés ou que le mappage n'est pas appelé, vérifiez que les mappes ont été installées dans le répertoire correspondant.

Conversion de dates

Remarque : Cette procédure de conversion des dates concerne uniquement les versions du connecteur antérieures à la version 1.5.0.

Utilisez les mappes pour convertir les données stockées au format `Date` dans la base de données par le format `String` utilisé par un objet métier WebSphere Business Integration Adapter.

Par exemple, supposons que vous vouliez convertir la date suivante, stockée dans une base de données Oracle :

```
Sun Jan 01 00:00:00 CEST 1999
```

par la chaîne suivante, traitée dans un objet métier WebSphere Business Integration Adapter for JDBC :

```
Jan 01 1999 00:00:00
```

Pour effectuer cette conversion, utilisez les constructeurs `DtpDate()` et `DtpSplitString()` définis pour la conversion des données dans le mappage. Pour connaître la syntaxe à utiliser et obtenir une description de ces constructeurs ainsi que les classes qui construisent les objets, voir le document *Map Development Guide*.

Pour utiliser une mappe afin de convertir la valeur `Date` au format `String`, procédez comme suit :

1. Utilisez `DtpSplitString()` avec un espace afin de découper la chaîne en six parties et la réorganiser pour que le constructeur `DtpDate` puisse l'utiliser. Pour convertir l'exemple de date, utilisez la commande suivante :

   ```
   DtpSplitString OurSplitString = new DtpSplitString
      ("Sun Jan 01 00:00:00 CEST 1999"," ");
   ```

   Dans l'instruction ci-dessus, `OurSplitString` est une variable définie par l'utilisateur du type `DtpSplitString`, qui contient un espace comme séparateur.
2. Utilisez la méthode `nextElement()` de la classe `DtpSplitString` pour effectuer une itération sur la variable `OurSplitString` qui vient d'être créée, en plaçant les six éléments de la variable dans un tableau dont les éléments sont de type `String`. L'exemple suivant spécifie la variable `OurStringPieces` comme tableau de sortie :

   ```
   String[] OurStringPieces = new String[6];
   for (i=0;i<=5;i=i+1){
      OurStringPieces[i]=OurSplitString.nextElement();
   }
   ```

   Cette itération génère les éléments de tableau suivants :

   ```
   OurStringPieces[0] = Sun
   OurStringPieces[1] = Jan
   OurStringPieces[2] = 01
   OurStringPieces[3] = 00:00:00
   OurStringPieces[4] = CEST
   OurStringPieces[5] = 1999
   ```
3. Concaténez les éléments de la chaîne requise pour l'entrée `DtpDate`. L'exemple de conversion utilise le format d'entrée `"M D Y h:m:s"` pour le constructeur `DtpDate`, dans lequel la chaîne convertie a le format `"Jan 01 1999 00:00:00"`. Cet exemple de format `String` utilise les éléments 1, 2, 5 et 3 du tableau `OurStringPieces` :

   ```
   OurConcatenatedString =
   OurStringPieces[1]+OurStringPieces[2]+OurStringPieces[5]+OurStringPieces[3];
   ```
4. Utilisez votre nouvelle chaîne concaténée comme entrée dans `DtpDate` :

   ```
   DtpDate OurDtpDate = new DtpDate(OurConcatenatedString,"M D Y h:m:s");
   ```

Une fois que vous avez mis la valeur `Date` au format `DtpDate`, vous pouvez utiliser la date dans votre mappe.

Gestion et consignation des erreurs

Le connecteur enregistre un message d'erreur chaque fois qu'il rencontre une condition qui entraîne l'échec du traitement d'un objet métier et d'une instruction. Lorsqu'une erreur survient, le connecteur imprime également une représentation textuelle de l'objet métier qui a échoué tel qu'il a été reçu. Il enregistre le texte

dans le fichier journal du connecteur ou le flot de sortie standard, selon sa configuration. Vous pouvez utiliser le texte pour vous aider à identifier la source de l'erreur.

Types d'erreur

Le tableau 16 présente les types de message de suivi que le connecteur produit à chaque niveau de trace. Ces messages viennent compléter les messages de suivi générés par l'architecture d'IBM WebSphere Business Integration Adapter, comme l'encapsuleur d'exécution de connecteur Java et l'interface de messages de WebSphere MQ.

Tableau 16. Messages de suivi de Connector

Niveau de trace	Messages de suivi
Niveau 0	Message qui identifie la version du connecteur. Aucun autre suivi n'est réalisé à ce niveau. Il s'agit de la valeur par défaut.
Niveau 1	• Messages d'état • Messages qui contiennent des informations d'identification (clé) pour chaque objet métier traité • Messages transmis chaque fois que la méthode pollForEvents est exécutée
Niveau 2	• Messages du gestionnaire d'objets métier qui contiennent des informations comme les tableaux et les objets métier enfant que le connecteur rencontre ou extrait pendant le traitement d'un objet métier • Messages journalisés chaque fois qu'un objet métier est transmis au courtier d'intégration, depuis la méthode `gotApplEvent()` ou `executeCollaboration()` • Messages qui indiquent qu'un objet métier a été reçu en tant que requête du courtier d'intégration
Niveau 3	• Messages de traitement d'une clé étrangère qui contiennent des informations telles que l'heure à laquelle le connecteur a détecté ou défini une clé étrangère dans un objet métier • Messages qui contiennent des informations sur le traitement de l'objet métier. Par exemple, ces messages sont transmis lorsque le connecteur détecte une correspondance entre des objets métier, ou qu'il identifie un objet métier dans un tableau d'objets métier enfant
Niveau 4	• Messages d'informations spécifiques à l'application, par exemple, des messages indiquant les valeurs renvoyées par les fonctions qui analysent les zones d'informations spécifiques à l'application de l'objet métier • Messages qui identifient lorsque le connecteur accède ou quitte une fonction, ce qui permet de suivre le flot de traitement du connecteur • Tous les messages spécifiques aux unités d'exécution. Si le connecteur engendre plusieurs unités d'exécution, un message signale la création de chaque nouvelle unité d'exécution

Tableau 16. Messages de suivi de Connector (suite)

Niveau de trace	Messages de suivi
Niveau 5	• Messages qui indiquent l'initialisation du connecteur, par exemple, des messages indiquant la valeur de chaque propriété de configuration extraite du courtier d'intégration • Messages qui contiennent des instructions exécutées dans l'application. A ce niveau de trace, le fichier journal du connecteur contient toutes les instructions exécutées dans l'application cible ainsi que la valeur des variables qui sont remplacées. • Messages qui comprennent la représentation d'un objet métier avant son traitement (en affichant son état tel que le connecteur le reçoit) et après son traitement (en affichant son état tel que le connecteur le renvoie) • Messages qui contiennent un cliché de l'objet métier • Messages qui indiquent l'état de chaque unité d'exécution que le connecteur engendre pendant son exécution

Messages d'erreur

Fichier de messages du connecteur

Tous les messages d'erreur générés par le connecteur sont stockés dans un fichier de messages nommé JDBCConnector.txt ou JDBCConnector_*II_TT*.txt (où *ll* correspond à une langue et *TT* à un pays ou territoire). Chaque erreur est associée à un numéro d'erreur suivi du message d'erreur. Par exemple :

```
20017
Connector Infrastructure version does not match.

20018
Connection from {1} to the Application is lost! Please enter 'q'
to stop the connector, then restart it after the problem is fixed.

20019
Error: ev_id is NULL in pollForEvent().
```

Interruption de la connexion à l'application

Si le connecteur ne parvient pas à établir la connexion, il envoie le message `FAIL` au courtier d'intégration et s'arrête.

Si la propriété `AutoCommit` a la valeur `false` et que la commande `PingQuery` échoue, le connecteur tentera à nouveau d'établir la connexion avec la base de données. S'il y parvient, il continuera le traitement. Dans le cas contraire, il renverra un message `APPRESPONSETIMEOUT`, qui arrêtera le connecteur.

Extraction en rupture de séquence

Vous devez affecter à la propriété `AutoCommit` la valeur false lorsque vous utilisez la base de données Oracle 8.0 et 8.1 avec Sun Solaris ou Oracle 8.1 avec Windows 2000. Dans le cas contraire, vous recevrez un message d'erreur de type ORA-01002 (extraction en rupture de séquence). Cette erreur ne se produit pas dans les versions précédentes des bases de données Oracle. Pour améliorer les performances, affectez la valeur ″false″ à la propriété `AutoCommit`.

Erreur de type Ressource occupée

Remarque : Ce connecteur rencontre cette erreur uniquement lorsqu'il est exécuté sur une base de données Oracle.

Lorsqu'il extrait ou modifie des données dans une application, il rencontre parfois une erreur du type :

```
[Time: 2001/05/29 16:30:07.356] [System: ConnectorAgent] [SS: SOVTConnector]
[Type: Trace] [Mesg: Select CLIENT,COUNTRY,STRT_CODE,CITY_CODE,CITYP_CODE,
STRTYPEAB,COMMU_CODE,REGIOGROUP,TAXJURCODE from ADRSTREET where CLIENT='100'
and COUNTRY='DE' and STRT_CODE='000001114136' FOR UPDATE NOWAIT]
[Time: 2001/05/29 16:30:07.526] [System: ConnectorAgent] [SS: SOVTConnector]
[Type: Trace ] [Mesg: :logMsg]
[Time: 2001/05/29 16:30:07.536] [System: ConnectorAgent] [SS: SOVTConnector]
 [Type: Error ] [MsgID: 37002]
[Mesg: Execution of Retrieve statement failed : java.
sql.SQLException: ORA-00054: Versuch, mit NOWAIT eine bereits
belegte Ressourceanzufordern.]
```

Cette erreur survient lorsque le connecteur tente de mettre à jour un enregistrement verrouillé. L'enregistrement est peut-être verrouillé par un autre processus ou si le connecteur a plusieurs unités d'exécution, il peut être verrouillé par le connecteur lui-même.

Notez que les enregistrements doivent être verrouillés pendant le processus de mise à jour. Le connecteur tente d'extraire une image postérieure de l'objet reçu par le courtier d'intégration et, dans le processus, verrouille l'objet entier dans la base de données afin de préserver l'intégrité des données.

Pour résoudre cet incident, vous pouvez arrêter le processus qui empêche le connecteur d'obtenir un verrou sur l'enregistrement, ou adapter la propriété de configuration RetryCountInterval au connecteur.

Le composant ODA for Manugistics ne se comporte pas normalement en raison d'un pilote JDBC non pris en charge

Si le pilote JDBC ne prend pas en charge une caractéristique du composant ODA for Manugistics, l'agent ODA (Object Discovery Agent) ne fonctionne pas correctement. Par exemple, si le pilote ne prend pas en charge tous les appels de méthode utilisés par le composant ODA for Manugistics, le fichier journal de l'ODA indique le processus qui a échoué. Voici un exemple de fichier journal :

```
[Time: 2002/05/15 17:00:55.147] [System: Object Discovery Agent] [SS: null]
[Type: 6] [Mesg: A SQL Error occurred in getting Schema Names from Database.
Reason [ProductName][ODBC ProductName Driver]Optional feature not
implemented]
```

Dans cette situation, vous devez utiliser un autre pilote JDBC.

Gestion des erreurs lors de l'utilisation d'IGP

Lorsqu'une opération d'insertion ou de mise à jour est réalisée par la procédure stockée définie sur la table d'interface, les erreurs sont placées dans une table d'erreur. La notification immédiate des échecs n'est pas possible pendant les insertions et les mises à jour dans les tables d'interface.

Le composant Integration Engineer devra créer des collaborations ou des flux de travaux pour connaître la sortie de ces types d'opérations en consultant les tables

d'erreur (ERR) appropriées. Ces erreurs peuvent également être corrigées en activant la notification d'événements sur les tables ERR.

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 couvrent 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 courtiers de messages WebSphere (WMQI).
- WebSphere Application Server (WAS)

Tous les connecteurs n'utilisent pas l'ensemble de ces propriétés standard. Lorsque vous sélectionnez un courtier d'intégration à partir de Connector Configurator, vous voyez la liste des propriétés standard que vous devez configurer pour que votre adaptateur s'exécute avec ce courtier.

Pour plus d'informations sur les propriétés spécifiques au connecteur, voir le guide d'utilisateur de l'adaptateur approprié.

Remarque : Dans ce document, les barres obliques inverses (\) sont utilisées comme convention pour les chemins d'accès aux répertoires. Pour les installations UNIX, remplacez les barres obliques inverses par des barres obliques (/) et suivez les conventions pour chaque système d'exploitation.

Propriétés nouvelles et supprimées

Ces propriétés standard ont été ajoutées dans cette version.

Nouvelles propriétés

- XMLNameSpaceFormat

Propriétés supprimées

- RestartCount

Configuration des propriétés standard du connecteur

Les connecteurs de l'adaptateur ont deux types de propriétés de configuration :

- les propriétés de configuration standard ;
- les propriétés spécifiques au connecteur.

Cette section décrit les propriétés de configuration standard. Pour plus d'informations sur les propriétés de configuration spécifiques à un connecteur, reportez-vous au guide d'utilisateur de l'adaptateur approprié.

Utilisation 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, reportez-vous à l'annexe sur Connector Configurator.

Remarque : 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.

Définition et mise à jour des valeurs des propriétés

La longueur par défaut d'une propriété est de 255 caractères.

Le connecteur utilise l'ordre suivant pour déterminer la valeur d'une propriété (le nombre le plus élevé remplace les autres valeurs) :

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

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

- **Dynamique**
  Les modifications prennent effet immédiatement après leur enregistrement dans System Manager. Si le connecteur s'exécute en mode autonome (indépendamment de System Manager), avec par exemple l'un des courtiers de message WebSphere, vous pouvez uniquement modifier les propriétés via le fichier de configuration. Dans ce cas, une mise à jour dynamique n'est pas possible.
- **Redémarrage de l'agent (ICS uniquement)**
  Les modifications prennent effet uniquement une fois que vous avez arrêté et redémarré le composant spécifique à l'application.
- **Redémarrage du composant**
  Les modifications prennent effet uniquement après que le connecteur a été arrêté et redémarré dans System Manager. Vous n'avez pas besoin d'arrêter et de redémarrer le composant spécifique à l'application ou le courtier d'intégration.
- **Redémarrage du serveur**
  Les modifications prennent effet uniquement une fois que vous avez arrêté et redémarré le composant spécifique à l'application et le courtier d'intégration.

Pour déterminer la manière dont une propriété spécifique est mise à jour, reportez-vous à la colonne **Update Method** dans la fenêtre Connector Configurator ou à la colonne Update Method dans le tableau 17 à la page 99 plus bas.

Récapitulatif des propriétés standard

Le tableau 17 contient un récapitulatif des propriétés de configuration standard du connecteur. Tous les connecteurs n'utilisent pas toutes ces propriétés, et les paramètres des propriétés peuvent différer d'un courtier d'intégration à l'autre, dans la mesure où les dépendances des propriétés standard sont basées sur `RepositoryDirectory`.

Vous devez définir les valeurs de certaines de ces propriétés avant d'exécuter le connecteur. Pour une explication sur chacune des propriétés, reportez-vous à la section suivante.

Remarque : Dans la colonne "Remarques" du tableau 17, la phrase "Le répertoire du référentiel est REMOTE" indique que le courtier est WebSphere InterChange Server. Lorsque le courtier est WMQI ou WAS, le répertoire du référentiel est LOCAL.

Tableau 17. Récapitulatif des propriétés de configuration standard

Nom de propriété	Valeurs possibles	Valeur par défaut	Méthode de mise à jour	Remarques
AdminInQueue	Nom de file d'attente JMS valide	*CONNECTORNAME* /ADMININQUEUE	Redémarrage composant	Le protocole de transfert est JMS
AdminOutQueue	Nom de file d'attente JMS valide	*CONNECTORNAME*/ADMINOUTQUEUE	Redémarrage composant	Le protocole de transfert est JMS
AgentConnections	1-4	1	Redémarrage composant	Le protocole de transfert est MQ ou IDL : le répertoire du référentiel est <REMOTE> (courtier ICS)
AgentTraceLevel	0-5	0	Dynamique	
ApplicationName	Nom d'application	Valeur indiquée pour le nom de l'application du connecteur	Redémarrage composant	
BrokerType	ICS, WMQI, WAS		Redémarrage composant	
CharacterEncoding	ascii7, ascii8, SJIS, Cp949, GBK, Big5, Cp297, Cp273, Cp280, Cp284, Cp037, Cp437 **Remarque :** Il s'agit d'un sous-ensemble de valeurs prises en charge.	ascii7	Redémarrage composant	
ConcurrentEventTriggeredFlows	1 à 32,767	1	Redémarrage composant	Le répertoire du référentiel est <REMOTE> (courtier ICS)
ContainerManagedEvents	No value ou JMS	No value	Redémarrage composant	Le protocole de transfert est JMS
ControllerStoreAndForwardMode	true ou false	true	Dynamique	Le répertoire du référentiel est <REMOTE> (courtier ICS)
ControllerTraceLevel	0-5	0	Dynamique	Le répertoire du référentiel est <REMOTE> (courtier ICS)

Tableau 17. Récapitulatif des propriétés de configuration standard (suite)

Nom de propriété	Valeurs possibles	Valeur par défaut	Méthode de mise à jour	Remarques
DeliveryQueue		*CONNECTORNAME*/DELIVERYQUEUE	Redémarrage composant	Transfert JMS uniquement
DeliveryTransport	MQ, IDL ou JMS	JMS	Redémarrage composant	Si le répertoire du référentiel est local, alors la valeur est JMS (uniquement)
DuplicateEventElimination	true ou false	false	Redémarrage composant	Transfert JMS uniquement : les événements gérés par le conteneur doivent être <NONE>
FaultQueue		*CONNECTORNAME*/FAULTQUEUE	Redémarrage composant	Transfert JMS uniquement
jms.FactoryClassName	CxCommon.Messaging.jms .IBMMQSeriesFactory ouCxCommon.Messaging .jms.SonicMQFactory ou n'importe quel nom de classe Java	CxCommon.Messaging. jms.IBMMQSeriesFactory	Redémarrage composant	Transfert JMS uniquement
jms.MessageBrokerName	Si FactoryClassName a la valeur IBM, utilisez crossworlds.queue. manager. Si FactoryClassName a la valeur Sonic, utilisez localhost:2506.	crossworlds.queue.manager	Redémarrage composant	Transfert JMS uniquement
jms.NumConcurrentRequests	Entier positif	10	Redémarrage composant	Transfert JMS uniquement
jms.Password	Tout mot de passe valide		Redémarrage composant	Transfert JMS uniquement
jms.UserName	Tout nom valide		Redémarrage composant	Transfert JMS uniquement
JvmMaxHeapSize	Taille de segment en mégaoctets	128 Mo	Redémarrage composant	Le répertoire du référentiel est <REMOTE> (courtier ICS)
JvmMaxNativeStackSize	Taille de la pile en kilo-octets	128 Ko	Redémarrage composant	Le répertoire du référentiel est <REMOTE> (courtier ICS)
JvmMinHeapSize	Taille de segment en mégaoctets	1 Mo	Redémarrage composant	Le répertoire du référentiel est <REMOTE> (courtier ICS)
ListenerConcurrency	1-100	1	Redémarrage composant	Le protocole de transfert doit être MQ

Tableau 17. Récapitulatif des propriétés de configuration standard (suite)

Nom de propriété	Valeurs possibles	Valeur par défaut	Méthode de mise à jour	Remarques
Locale	en_US, ja_JP, ko_KR, zh_CN, zh_TW, fr_FR, de_DE, it_IT, es_ES, pt_BR **Remarque :** Il s'agit d'un sous-ensemble des environnements locaux pris en charge.	en_US	Redémarrage composant	
LogAtInterchangeEnd	true ou false	false	Redémarrage composant	Le répertoire du référentiel doit être <REMOTE> (courtier ICS)
MaxEventCapacity	1-2147483647	2147483647	Dynamique	Le répertoire du référentiel doit être <REMOTE> (courtier ICS)
MessageFileName	Chemin d'accès ou nom de fichier	CONNECTORNAMEConnector.txt	Redémarrage composant	
MonitorQueue	Tout nom de file d'attente valide	*CONNECTORNAME*/MONITORQUEUE	Redémarrage composant	Transfert JMS uniquement : DuplicateEvent doit avoir la valeur true
OADAutoRestartAgent	true ou false	false	Dynamique	Le répertoire du référentiel doit être <REMOTE> (courtier ICS)
OADMaxNumRetry	Nombre positif	1000	Dynamique	Le répertoire du référentiel doit être <REMOTE> (courtier ICS)
OADRetryTimeInterval	Nombre positif en minutes	10	Dynamique	Le répertoire du référentiel doit être <REMOTE> (courtier ICS)
PollEndTime	HH:MM	HH:MM	Redémarrage composant	
PollFrequency	Nombre positif en millisecondes no (pour désactiver l'interrogation) key (pour interroger uniquement lorsque la lettre p est entrée dans la fenêtre Command Prompt du connecteur)	10000	Dynamique	
PollQuantity	1-500	1	Redémarrage de l'agent	Transfert JMS uniquement : Les événements gérés du conteneur sont indiqués
PollStartTime	HH:MM(HH va de 0 à 23, MM va de 0 à 59)	HH:MM	Redémarrage composant	

Tableau 17. Récapitulatif des propriétés de configuration standard (suite)

Nom de propriété	Valeurs possibles	Valeur par défaut	Méthode de mise à jour	Remarques
RepositoryDirectory	Emplacement du référentiel des métadonnées		Redémarrage de l'agent	ICS : <REMOTE> Courtier de messages MQ WebSphere et WAS : référentiel C:\crossworlds\
RequestQueue	Nom de file d'attente JMS valide	*CONNECTORNAME*/REQUESTQUEUE	Redémarrage composant	Le protocole de transfert est JMS
ResponseQueue	Nom de file d'attente JMS valide	*CONNECTORNAME*/RESPONSEQUEUE	Redémarrage composant	Le protocole de transfert est JMS uniquement si le répertoire du référentiel est <REMOTE>
RestartRetryCount	0-99	3	Dynamique	
RestartRetryInterval	Valeur positive sensible en minutes : 1 - 2147483547	1	Dynamique	
RHF2MessageDomain	mrm, xml	mrm	Redémarrage du composant	Uniquement si le protocole de transfert est JMS et que le format WF est CwXML.
SourceQueue	Nom WebSphere MQ valide	*CONNECTORNAME*/SOURCEQUEUE	Redémarrage de l'agent	Uniquement si le protocole de transfert est JMS et que les événements gérés du conteneur sont indiqués
SynchronousRequestQueue		*CONNECTORNAME*/ SYNCHRONOUSREQUESTQUEUE	Redémarrage composant	Le protocole de transfert est JMS
SynchronousRequestTimeout	0 - n'importe quel nombre (millisecondes)	0	Redémarrage composant	Le protocole de transfert est JMS
SynchronousResponseQueue		*CONNECTORNAME*/ SYNCHRONOUSRESPONSEQUEUE	Redémarrage composant	Le protocole de transfert est JMS
WireFormat	CwXML, CwBO	CwXML	Redémarrage de l'agent	CwXML si le répertoire du référentiel n'est pas <REMOTE> : CwBO si le répertoire du référentiel est <REMOTE>
WsifSynchronousRequestTimeout	0 - n'importe quel nombre (millisecondes)	0	Redémarrage composant	WAS uniquement
XMLNameSpaceFormat	short, long	short	Redémarrage de l'agent	Courtiers de messages WebSphere MQ et WAS uniquement

Propriétés de configuration standard

Cette section répertorie et définit chacune des propriétés de configuration standard du connecteur.

AdminInQueue

File d'attente utilisée par le courtier d'intégration pour envoyer des messages administratifs au connecteur.

La valeur par défaut est `CONNECTORNAME/ADMININQUEUE`.

AdminOutQueue

File d'attente utilisée par le connecteur pour envoyer des messages administratifs au courtier d'intégration.

La valeur par défaut est `CONNECTORNAME/ADMINOUTQUEUE`.

AgentConnections

Applicable uniquement si `RepositoryDirectory` a la valeur <REMOTE>.

La propriété AgentConnections contrôle le nombre de connexions ORB (Object Request Broker) ouvertes par `orb.init[]`.

La valeur par défaut de cette propriété est 1. Vous pouvez la modifier le cas échéant.

AgentTraceLevel

Niveau des messages de trace pour le composant spécifique à l'application. La valeur par défaut est `0`. Le connecteur fournit tous les messages de trace applicables au niveau de trace défini ou à un niveau inférieur.

ApplicationName

Nom qui identifie de manière unique l'application du connecteur. Ce nom permet à l'administrateur système de contrôler l'environnement système WebSphere Business Integration. Vous devez attribuer une valeur à cette propriété avant d'exécuter le connecteur.

BrokerType

Identifie le type de courtier d'intégration que vous utilisez. Les options sont ICS, les courtiers de messages WebSphere (WMQI, WMQIB ou WBIMB) ou WAS.

CharacterEncoding

Indique le jeu de codes de caractères utilisé pour mapper un caractère (comme une lettre de l'alphabet, une représentation numérique ou un signe de ponctuation) à une valeur numérique.

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

Par défaut, un sous-ensemble des codages de caractères pris en charge uniquement s'affiche dans la liste déroulante. Pour ajouter à cette liste d'autres valeurs prises

en charge, vous devez modifier manuellement le fichier \Data\Std\stdConnProps.xml dans le répertoire produit. Pour plus d'informations, voir l'annexe sur Connector Configurator.

ConcurrentEventTriggeredFlows

Applicable uniquement si RepositoryDirectory a la valeur <REMOTE>.

Détermine le nombre d'objets métier pouvant être traités simultanément par le connecteur pour la transmission des événements. La valeur de cet attribut doit correspondre au nombre d'objets métier que vous souhaitez mapper et transmettre simultanément. Par exemple, affectez à cette propriété la valeur 5 pour que cinq objets métier soient traités simultanément. La valeur par défaut est 1.

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 la collaboration en attribuant à sa propriété Maximum number of concurrent events une valeur suffisamment élevée pour qu'elle utilise plusieurs unités d'exécution ;
- vérifier que le composant d'application cible peut exécuter 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 traiter pour plusieurs processus. Attribuez une valeur supérieure à 1 à la propriété de configuration Parallel Process Degree.

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

ContainerManagedEvents

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

Il n'y a pas de valeur par défaut.

Lorsque ContainerManagedEvents est défini sur JMS, vous devez configurer les propriétés suivantes pour activer la transmission d'événement garantie :

- PollQuantity = 1 à 500
- SourceQueue = /SOURCEQUEUE

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

Ces propriétés sont spécifiques à l'adaptateur, mais voici des valeurs d'**exemple** :

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

Les zones correspondant à ces valeurs dans l'onglet Data Handler s'affichent uniquement si vous avez défini ContainerManagedEvents sur JMS.

Remarque : Lorsque ContainerManagedEvents a la valeur JMS, le connecteur n'appelle *pas* sa méthode pollForEvents(), désactivant ainsi la fonctionnalité de celle-ci.

Cette propriété apparaît uniquement si la propriété DeliveryTransport a la valeur JMS.

ControllerStoreAndForwardMode

Applicable uniquement si RepositoryDirectory a la valeur <REMOTE>.

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.

La valeur par défaut est true.

ControllerTraceLevel

Applicable uniquement si RepositoryDirectory a la valeur <REMOTE>.

Niveau des messages de trace pour le contrôleur du connecteur. La valeur par défaut est 0.

DeliveryQueue

Applicable uniquement si DeliveryTransport a la valeur JMS.

File d'attente utilisée par le connecteur pour envoyer des objets métier au courtier d'intégration.

La valeur par défaut est CONNECTORNAME/DELIVERYQUEUE.

DeliveryTransport

Spécifie le mécanisme de transfert pour la transmission des événements. La valeurs possibles sont `MQ` pour WebSphere MQ, `IDL` pour CORBA IIOP ou `JMS` pour Java Messaging Service.

- Si `RepositoryDirectory` est distant, la valeur de la propriété `DeliveryTransport` peut être MQ, IDL ou JMS et la valeur par défaut est IDL.
- Si `RepositoryDirectory` est un répertoire local, la valeur associée ne peut être que JMS.

Le connecteur envoie les requêtes d'appel de service et les messages administratifs par CORBA IIOP si la valeur de la propriété `DeliveryTransport` est `MQ` ou `IDL`.

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 permet d'éviter l'écriture d'événements pouvant être 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

Permet la communication entre le connecteur et la structure du connecteur de client à l'aide de Java Messaging Service (JMS).

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

Important : 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
- Lorsqu'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 moins de 768 Mo en taille de segment de processus, IBM recommande que vous définissiez :

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

  Ce script se trouve dans le répertoire \bin sous le répertoire produit. 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 remplacement des pages peut se produire, ce qui peut affecter les performances de votre système.
- La propriété IPCCBaseAddress sur 11 ou 12. Pour plus d'informations sur cette propriété, voir le document *System Installation Guide for UNIX*.

DuplicateEventElimination

Lorsque vous affectez à cette propriété la valeur true, un connecteur activé par JMS peut vérifier que des doublons ne sont pas transmis à la file d'attente de transmission. Pour utiliser cette fonction, le connecteur doit avoir un identificateur d'événement unique défini en tant qu'attribut **ObjectEventId** de l'objet métier dans le code spécifique à l'application. Cette opération s'effectue lors du développement du connecteur.

Cette propriété peut également avoir la valeur false.

Remarque : Lorsque DuplicateEventElimination a la valeur true, vous devez également configurer la propriété MonitorQueue pour activer la transmission d'événement garantie.

FaultQueue

Si le connecteur rencontre une erreur lors du traitement d'un message, il transmet ce message à la file d'attente spécifiée dans cette propriété, accompagné d'un indicateur d'état et d'une description de l'incident.

La valeur par défaut est CONNECTORNAME/FAULTQUEUE.

JvmMaxHeapSize

Taille de segment maximale pour l'agent (en mégaoctets). Cette propriété est applicable uniquement si la valeur RepositoryDirectory est <REMOTE>.

La valeur par défaut est 128 Mo.

JvmMaxNativeStackSize

Espace mémoire natif maximal pour l'agent (en kilo-octets). Cette propriété est applicable uniquement si la valeur RepositoryDirectory est <REMOTE>.

La valeur par défaut est 128 Ko.

JvmMinHeapSize

Taille de segment minimale pour l'agent (en mégaoctets). Cette propriété est applicable uniquement si la valeur RepositoryDirectory est <REMOTE>.

La valeur par défaut est 1 Mo.

jms.FactoryClassName

Indique le nom de classe à instancier pour un fournisseur JMS. Vous *devez* définir cette propriété de connecteur lorsque vous choisissez JMS comme mécanisme de transfert (DeliveryTransport).

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

jms.MessageBrokerName

Indique le nom de courtier à utiliser pour le fournisseur JMS. Vous *devez* définir cette propriété de connecteurs lorsque vous choisissez JMS en tant que mécanisme de transfert (DeliveryTransport).

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

Lorsque vous vous connectez à un courtier de messages distant, cette propriété prend les valeurs (obligatoires) suivantes :
QueueMgrName:<Channel>:<HostName>:<PortNumber>,
où les variables représentent respectivement :
QueueMgrName : le nom du gestionnaire de files d'attente ;
Channel : le canal utilisé par le client ;
HostName : le nom de la machine sur laquelle le gestionnaire de files d'attente doit résider ;
PortNumber : le numéro de port que le gestionnaire de files d'attente doit utiliser pour écouter.

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

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

Indique le mot de passe défini pour le fournisseur JMS. Cette valeur est facultative.

Il n'y a pas de valeur par défaut.

jms.UserName

Indique le nom d'utilisateur défini pour le fournisseur JMS. Cette valeur est facultative.

Il n'y a pas de valeur par défaut.

ListenerConcurrency

Cette propriété prend en charge le traitement de plusieurs unités d'exécution dans MQ Listener lorsque ICS est le courtier d'intégration. Elle permet l'écriture par lots de plusieurs événements sur la base de données, améliorant ainsi les performances du système. La valeur par défaut est 1.

Cette propriété s'applique uniquement aux connecteurs utilisant le transfert MQ. La propriété `DeliveryTransport` doit être définie sur `MQ`.

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	correspond à un code de langue à deux caractères (habituellement en minuscule) ;
TT	correspond à un code territoire ou pays à deux caractères (habituellement en majuscule) ;
codeset	correspond au jeu de codes de caractères associé ; cette partie du nom est souvent facultative.

Par défaut, seul un sous-ensemble d'environnements locaux pris en charge apparaît dans la liste déroulante. Pour ajouter à cette liste d'autres valeurs prises en charges, vous devez modifier manuellement le fichier `\Data\Std\stdConnProps.xml` situé dans le répertoire produit. Pour plus d'informations, voir l'annexe sur Connector Configurator.

La valeur par défaut est `en_US`. Si le connecteur n'a pas été internationalisé, la seule valeur correcte pour cette propriété est `en_US`. Pour déterminer si un connecteur a été internationalisé, consultez la liste des versions de connecteur sur ces sites Web :

http://www.ibm.com/software/websphere/wbiadapters/infocenter ou
http://www.ibm.com/websphere/integration/wicserver/infocenter

LogAtInterchangeEnd

Applicable uniquement si RepositoryDirectory a la valeur <REMOTE>.

Indique s'il faut consigner les erreurs dans la cible du journal du courtier d'intégration. La consignation des erreurs dans la cible du journal du courtier d'intégration active également la notification par courrier électronique qui, lorsque des erreurs ou des erreurs fatales ont lieu, génère des messages électroniques pour le destinataire `MESSAGE_RECIPIENT` spécifié dans le fichier `InterchangeSystem.cfg`.

Par exemple, lorsque la connexion entre un connecteur et son application est interrompue, si `LogAtInterChangeEnd` a la valeur `true`, un courrier électronique est envoyé au destinataire du message spécifié. La valeur par défaut est `false`.

MaxEventCapacity

Nombre maximal d'événements contenus dans la mémoire tampon du contrôleur. Cette propriété est utilisée par le contrôle du flux et est applicable uniquement si la valeur de la propriété `RepositoryDirectory` est <REMOTE>.

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

MessageFileName

Nom du fichier de messages du connecteur. L'emplacement standard de ce fichier de messages est `\connectors\messages` dans le répertoire produit. Indique le nom du fichier du message dans un chemin d'accès absolu si le fichier de messages n'est pas situé à l'emplacement standard.

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 de l'utilisateur de l'adaptateur.

MonitorQueue

File d'attente logique utilisée par le connecteur pour contrôler les événements en double. Elle est uniquement utilisée si la propriété `DeliveryTransport` a la valeur JMS et que `DuplicateEventElimination` a la valeur `TRUE`.

La valeur par défaut est `CONNECTORNAME/MONITORQUEUE`.

OADAutoRestartAgent

Valide uniquement lorsque `RepositoryDirectory` a la valeur <REMOTE>.

Indique si le connecteur utilise la fonction de redémarrage automatique et à distance. Cette fonction utilise le démon d'activation d'objets (OAD, Object Activation Daemon) déclenché par MQ pour redémarrer le connecteur après un arrêt anormal ou pour démarrer un connecteur distant 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 MQ, voir le document *Installation Guide for Windows* ou *for UNIX*.

La valeur par défaut est `false`.

OADMaxNumRetry

Valide uniquement lorsque `RepositoryDirectory` a la valeur <REMOTE>.

Indique le nombre maximal de tentatives de redémarrage automatique du connecteur par l'OAD déclenché par MQ après un arrêt anormal. La propriété `OADAutoRestartAgent` doit avoir la valeur `true` pour prendre effet.

La valeur par défaut est `1000`.

OADRetryTimeInterval

Valide uniquement lorsque `RepositoryDirectory` a la valeur <REMOTE>.

Indique la durée en minutes entre les tentatives de relance pour l'OAD déclenché par MQ. 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 le nombre spécifié par la propriété `OADMaxNumRetry`. La propriété `OADAutoRestartAgent` doit être définie sur `true` pour prendre effet.

La valeur par défaut est `10`.

PollEndTime

Heure d'arrêt de l'interrogation de la file d'attente des événements. Le format est `HH:MM`, où *HH* représente les heures (de 0 à 23) et *MM* représente les secondes (de 0 à 59).

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

PollFrequency

Intervalle compris entre la fin de la dernière interrogation et le début de la suivante. La propriété `PollFrequency` indique la durée (en millisecondes) de l'intervalle compris entre la fin de la dernière interrogation et le début de l'interrogation suivante. Il ne s'agit pas de l'intervalle compris entre chaque interrogation. Plus précisément, la logique est la suivante :

- Emettez une interrogation pour obtenir le nombre d'objets spécifié par la valeur de la propriété `PollQuantity`.
- Traitez ces objets. Pour certains adaptateurs, cela peut être en partie effectué 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.

Affectez à la propriété `PollFrequency` l'une des valeurs suivantes :

- Le nombre de millisecondes entre les interrogations (nombre entier).
- 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.
- Le mot `no`, pour que le connecteur n'émette pas d'interrogation. 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

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 la définition du nombre d'interrogations, la valeur définie dans cette propriété spécifique remplace la valeur de la propriété standard.

FIX

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

Première interrogation : le connecteur va sélectionner 1. le corps du message, étant donné qu'il est aussi considéré comme une pièce jointe. Si aucun gestionnaire de données n'a été spécifié pour ce type mime, il ignorera le corps de texte. 2. le connecteur traite la première pièce jointe du PO. Si un gestionnaire de données est disponible pour ce type mime, le connecteur envoie l'objet métier au Visual Test Connector (VTC). Si le 3. accept in VTC again no BO should come thru deuxième interrogation 1. le connecteur traite la deuxième pièce jointe du PO. Un gestionnaire de données est disponible pour ce type mime, il envoie donc le BO au VTC2. accept in VTC again now the third PO attachment should come through. Il s'agit du comportement correct du connecteur.

PollStartTime

Heure de début 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`, mais elle doit être modifiée.

RequestQueue

File d'attente utilisée par le courtier d'intégration pour envoyer des objets métier au connecteur.

La valeur par défaut est `CONNECTOR/REQUESTQUEUE`.

RepositoryDirectory

Emplacement du référentiel à partir duquel le connecteur lit les schémas XML qui stockent les métadonnées pour la définition des objets métier.

Lorsqu'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.

Lorsqu'un courtier de messages WebSphere ou WAS est le courtier d'intégration, cette valeur doit être définie sur *<local directory>*.

ResponseQueue

Applicable uniquement si `DeliveryTransport` a la valeur JMS et obligatoire uniquement si `RepositoryDirectory` a la valeur <REMOTE>.

Désigne la file d'attente de réponses JMS, qui transmet un message de réponse depuis la structure 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.

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 asservi par le composant spécifique à l'application du connecteur maître.

La valeur par défaut est 3.

RestartRetryInterval

Indique l'intervalle en minutes pendant lequel le connecteur tente de redémarrer. Lorsqu'elle est utilisée pour un connecteur parallèle, cette propriété indique l'intervalle de temps pendant lequel le composant spécifique à l'application du connecteur principal tente de redémarrer le composant spécifique à l'application du connecteur asservi. Les valeurs possibles varient de 1 à 2147483647.

La valeur par défaut est 1.

RHF2MessageDomain

Courtiers de messages WebSphere et WAS uniquement.

Cette propriété vous permet de configurer la valeur du nom de domaine de la zone dans l'en-tête JMS. Lorsque les données sont envoyées au WMQI par transfert JMS, la structure 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 permet aux utilisateurs de suivre le mode de traitement des données du message par le courtier WMQI.

Voici un modèle d'en-tête :

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

La valeur par défaut est `mrm`, mais elle peut également avoir la valeur `xml`. Cette propriété apparaît uniquement lorsque DeliveryTransport a la valeur `JMS` et que le format WF a la valeur `CwXML`.

SourceQueue

Applicable uniquement si `DeliveryTransport` a la valeur JMS et que la propriété ContainerManagedEvents est spécifiée.

Désigne la file d'attente source JMS de la structure du connecteur qui assure la transmission d'événements garantie pour les connecteur activés par JMS qui utilisent un magasin d'événements JMS. Pour plus d'informations, voir «ContainerManagedEvents» à la page 104.

La valeur par défaut est `CONNECTOR/SOURCEQUEUE`.

SynchronousRequestQueue

Applicable uniquement si `DeliveryTransport` a la valeur JMS.

Transmet les messages de requête qui requièrent une réponse synchrone de la structure du connecteur vers le courtier. Cette file d'attente est nécessaire uniquement si le connecteur utilise l'exécution synchrone. Avec l'exécution synchrone, la structure du connecteur envoie un message à la file d'attente

SynchronousRequestQueue et attend une réponse de la part du courtier sur la file d'attente SynchronousResponseQueue. Le message de réponse envoyé au connecteur porte un ID de corrélation qui correspond à l'ID du message d'origine.

La valeur par défaut est CONNECTORNAME/SYNCHRONOUSREQUESTQUEUE.

SynchronousResponseQueue

Applicable uniquement si DeliveryTransport a la valeur JMS.

Transmet les messages de réponse à une requête synchrone entre le courtier et la structure du connecteur. Cette file d'attente est nécessaire uniquement si le connecteur utilise l'exécution synchrone.

La valeur par défaut est CONNECTORNAME/SYNCHRONOUSRESPONSEQUEUE.

SynchronousRequestTimeout

Applicable uniquement si DeliveryTransport a la valeur JMS.

Indique la durée d'attente en minutes 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.

La valeur par défaut est 0.

WireFormat

Format du message lors du transfert.

- Si RepositoryDirectory est un répertoire local, le paramètre est CwXML.
- Si la valeur de RepositoryDirectory est <REMOTE>, le paramètre est CwBO.

WsifSynchronousRequestTimeout

Courtiers d'intégration WAS uniquement.

Indique la durée d'attente en minutes 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.

La valeur par défaut est 0.

XMLNameSpaceFormat

Courtiers de messages WebSphere et courtier d'intégration WAS uniquement.

Propriété stricte permettant à l'utilisateur de spécifier des espaces de nom courts et longs des définitions des objets métier au format XML.

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.

Remarque :

Dans ce document, les barres obliques inverses (\) sont utilisées comme convention pour les chemins d'accès aux répertoires. Pour les installations UNIX, remplacez les barres obliques inverses par des barres obliques (/) et suivez les conventions pour chaque système d'exploitation.

Les sujets traités dans cette annexe sont les suivants :

Présentation de Connector Configurator

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

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

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 116).

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 117 pour configurer un nouveau modèle.

Remarque : 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.

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 et utiliser les fichiers de configuration indépendamment de votre courtier.

Pour ce faire, procédez comme suit :

- Dans **Démarrer>Programmes**, cliquez sur **IBM WebSphere InterChange Server>IBM WebSphere Business Integration Tools>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 122).

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 **Connecteurs**.
3. Dans la barre de menus de System Manager, cliquez sur **Outils>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 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 117.
- 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
  Updated Method
  Description
- **Flags**
  Standard flags
- **Custom Flag**
  Flag

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. Sélectionnez la propriété dans la liste **Edit properties** et cliquez dessus avec le bouton droit.

2. Dans la boîte de dialogue, cliquez sur **Add**.
3. Entrez le nom de 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.

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.

- Dans la fenêtre System Manager, cliquez avec le bouton droit sur le dossier **Connectors** et sélectionnez **Create New Connector**. Connector Configurator affiche la boîte de dialogue **New Connector**.
- En mode autonome : dans Connector Configurator, sélectionnez **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.
- Faites défiler les zones restantes dans la fenêtre **New Connector**, comme indiqué plus loin 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. Cliquez sur **File>New>Connector Configuration**.
2. 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**.
3. 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 parcourir la définition maintenant ou enregistrer le fichier et renseigner les zones ultérieurement.
4. 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.

5. 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**, 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 associées au courtier sélectionné. 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 125.
3. 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 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.

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)

Important : Connector Configurator accepte des valeurs de propriété dans des jeux de caractères anglais et des jeux de caractères qui n'existent pas en anglais. 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.
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** (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.

Définition des propriétés de configuration spécifiques à l'application

Pour les propriétés de configuration spécifiques à l'application, vous pouvez ajouter ou modifier les noms des propriétés, définir des valeurs, supprimer une propriété et chiffrer une propriété. 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.
3. Pour chiffrer une propriété, cochez la case **Encrypt**.
4. Vous pouvez enregistrer ou ignorer les modifications, comme décrit pour «Définition des propriétés standard du connecteur».

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 contenues dans l'annexe *Propriétés de configuration standard pour les connecteurs* dans «Définition et mise à jour des valeurs des propriétés» à la page 98.

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 boîte à liste déroulante affiche 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 uniquement)

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**.

- **Explicit**

  Dans certains cas, vous devez 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)

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.

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 et les zones disponibles dans l'écran des propriétés changent immédiatement, et seules les onglets et les zones appartenant au nouveau courtier sélectionné apparaissent.

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. Prise en charge de la valeur null et des valeurs vides

Cette annexe décrit différents scénarios de succès et d'échec dans lesquels la valeur de la clé d'un objet métier est null ou vide. Elle contient également les modifications fonctionnelles requises pour les valeurs null ou vides d'un objet métier.

Scénarios de succès et d'échec

Si la valeur de clé d'un objet métier est vide ou null dans la base de données, créez la clause where avec le type ″is null″ au lieu de l'opérateur de type ″=″.

IBM recommande d'affecter aux objets métier au moins un attribut de clé ne comportant pas de valeur vide.

Le scénario suivant est un objet parent avec une clé de valeur null. Ces conditions provoquent l'échec du scénario.

Tableau 18. Objet Customer

Attribut	**Type**
cid	Integer (Key)
name	String
comments	String

Le scénario suivant introduit un objet parent possédant deux clés dont une de valeur null. Ces conditions permettent le succès du scénario.

Tableau 19. Objet Customer

Attribut	**Type**
cid	Integer (Key)
name	String
comments	String

Dans le deuxième scénario, créez la requête d'extraction en sélectionnant les attributs cid, name et comments auprès de l'objet Customer, où l'attribut cid=1000 et l'attribut name a une valeur null.

Le scénario suivant introduit un objet parent avec un objet enfant dans un objet conteneur avec une référence de clé étrangère. Ces conditions provoquent l'échec du scénario.

Tableau 20. Objet Customer

Attribut	**Type**
cid	Integer (Key)
name	String (Key)
comments	String

Tableau 20. Objet Customer (suite)

Attribut	Type
Address	Address
Aid	Integer (Key) ASI:FK=cid
Acity	String
Azip	String

Si l'attribut cid contient une valeur null, créez la requête d'extraction en sélectionnant Aid, Acity et Azip à partir de l'objet Address. Affectez à l'attribut Aid la valeur null.

Le scénario suivant introduit un objet parent avec un objet enfant dans un objet conteneur avec deux références de clés. Ces conditions permettent le succès du scénario.

Tableau 21. Objet Customer

Attribut	Type
cid	Integer (Key)
name	String
comments	String
Address	Address
Aid	Integer (Key) ASI:FK=cid
Acity	String (Key) ASI:FK=name
Azip	String

Si l'attribut name contient une valeur null, créez la requête d'extraction en sélectionnant Aid, Acity et Azip à partir de l'objet Address, où l'attribut Aid=Cid et l'attribut Acity a une valeur null.

Fonctionnalité

Si le connecteur rencontre une valeur vide dans une clé, il compare cette valeur avec la valeur UseNull de l'attribut. Si la valeur est true, le connecteur ajoute la valeur null à la requête. Les instructions suivantes s'en trouvent modifiées :

- Retrieve
- RetrieveByContent
- Update
- Delete

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 commercial IBM. Toute référence à un produit, logiciel ou service IBM n'implique pas que seul ce produit, logiciel ou service puisse être utilisé. Tout autre élément fonctionnellement équivalent peut être utilisé, s'il n'enfreint aucun droit d'IBM. Il est de la responsabilité de l'utilisateur d'évaluer et de vérifier lui-même les installations et applications réalisées avec des produits, logiciels ou services non expressément référencés par IBM.

IBM peut détenir des brevets ou des demandes de brevet couvrant les produits mentionnés dans le présent document. La remise de ce document ne vous donne aucun droit de licence sur ces brevets ou demandes de brevet. Si vous désirez recevoir des informations concernant l'acquisition de licences, veuillez en faire la demande par écrit à l'adresse suivante :

IBM EMEA Director of Licensing
IBM Europe Middle-East Africa
Tour Descartes
92066 Paris-La Défense Cedex 50
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

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". IBM DECLINE TOUTE RESPONSABILITE, EXPLICITE OU IMPLICITE, RELATIVE AUX INFORMATIONS QUI Y SONT CONTENUES, Y COMPRIS EN CE QUI CONCERNE LES GARANTIES DE VALEUR 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.

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 modifier sans préavis 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 Burlingame Laboratory Director
IBM Burlingame Laboratory
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.

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.

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.

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 :

IBM
le logo IBM
AIX
CrossWorlds
DB2
DB2 Universal Database
Domino
Lotus
Lotus Notes
MQIntegrator
MQSeries
Tivoli
WebSphere

Microsoft, Windows, Windows NT et le logo Windows sont des marques de Microsoft Corporation aux Etats-Unis et/ou dans certains autres pays.

MMX, Pentium et ProShare sont des marques d'Intel Corporation aux Etats-Unis et/ou dans certains autres pays.

Java et toutes les marques incluant Java sont des marques de Sun Microsystems, Inc. aux Etats-Unis et/ou dans certains autres pays.

D'autres sociétés sont propriétaires des autres marques, noms de produits ou logos qui pourraient apparaître dans ce document.

Le composant Adapter for Manugistics inclut un logiciel développé par Eclipse Project (http://www.eclipse.org)

WebSphere Business Integration Adapter Framework V2.4.0.

IBM WebSphere InterChange Server V4.2.1, IBM WebSphere Business Integration Toolset V4.2.1, IBM WebSphere Business Integration Adapters V2.3.1, IBM WebSphere Business Integration Collaborations V4.2.