Composants IBM WebSphere Business Integration
Adapter

Adaptateur pour Enterprise Java Beans Architecture - Guide d'utilisation

Version 1.2.x

Composants IBM WebSphere Business Integration
Adapter

Adaptateur pour Enterprise Java Beans Architecture - Guide d'utilisation

Version 1.2.x

Consigne

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

Remarque

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

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

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

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

Vous pouvez également consulter les serveurs Internet suivants :

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

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

Table des matières

Avis aux lecteurs canadiens

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

Illustrations

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

Terminologie

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

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

Claviers

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

OS/2 et Windows - Paramètres canadiens

Au Canada, on utilise :

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

Nomenclature

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

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

Brevets

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

Assistance téléphonique

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

A propos de ce document

La famille de produits IBM WebSphere Business Integration Adapter propose une connectivité d'intégration pour les technologies e-business de pointe, les applications d'entreprise, les systèmes existants et 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.

Ce document décrit l'installation, la configuration, le développement d'objets métier et la résolution des incidents pour IBM WebSphere Business Integration Adapter for Enterprise Java Beans Architecture.

Public visé

Ce document s'adresse aux consultants, développeurs et aux administrateurs système qui prennent en charge et gèrent le système d'intégration WebSphere sur les sites clients.

Prérequis pour ce document

Les utilisateurs doivent bien connaître le système d'intégration WebSphere, le développement d'objets métier et de collaboration ainsi que la technologie Enterprise JavaBeans.

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 données 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 WebSphere Business Integration Support Web, http://www.ibm.com/software/integration/websphere/support/. Sélectionnez la zone qui vous intéresse et parcourez les sections Technotes et Flashes. Des informations complémentaires sont également disponibles auprès d'IBM Redbooks à l'adresse http://www.redbooks.ibm.com/.

Conventions typographiques

Ce document utilise les conventions suivantes :

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

Nouveautés de la présente édition

Nouveautés de l'édition 1.2.x

- A partir de l'édition 1.2.x du présent document, le WebSphere Business Integration Adapter for Enterprise Java Beans devient WebSphere Business Integration Adapter for Enterprise Java Beans Architecture.
- L'ODA prend en charge la génération d'objets métier hiérarchiques
- La nouvelle propriété `RetryDuration` limite le nombre de tentatives de connexion de l'adaptateur au serveur d'applications en cas de panne de ce dernier.
- L'adaptateur pour EJB Architecture ne prend plus en charge InterChange Server version 4.1.1.
- La gestion de l'adaptateur pour EJB Architecture par IBM Tivoli License Manager (ITLM) est désormais prise en charge.
- Prise en charge des classes de collection Java.
- L'adaptateur pour EBJ Architecture et l'ODA prend désormais en charge le mappage des méthodes getter/setter sur les attributs d'objet métier.
- Prise en charge de JDK version 1.4.2.

Nouveautés de l'édition 1.1.x

Juin 2004 :

- A partir de l'édition 1.1.x, l'adaptateur pour EJB n'est plus pris en charge sous Solaris 7. Par conséquent, les références à cette version de plateforme ont été supprimées de ce guide.
- La section «Mappage des attributs : Enterprise JavaBeans (EJB) et objet métier», à la page 34 a été mise à jour avec les nouvelles informations de mappage de type de données pour les constructions EJB suivantes :
  - `java.math.BigDecimal`
  - `java.math.BigInteger`

Nouveautés de l'édition 1.0.x

Février 2004 :

- Les informations relatives aux données dépendant des paramètres régionaux ont été déplacées dans la section «Environnement de l'adaptateur», à la page 1.
- La section «Traitement des instructions», à la page 8 a été mise à jour avec des informations sur la différence entre une ASI d'instruction vide pour un bean d'entreprise éloigné et pour un objet Java.
- La section «Structure de fichiers du connecteur», à la page 11 a été mise à jour avec les noms d'autres fichiers qui sont copiés sur votre système pendant l'installation. Si vous avez déjà installé le produit, vous n'avez pas à le réinstaller. Les fichiers ont déjà été copiés sur votre système lorsque vous avez exécuté le programme d'installation.
- La section «Configuration du fichier de démarrage», à la page 24 a été mise à jour avec un nouveau nom de fichier exemple.
- Les informations relatives aux méthodes et attributs d'objet métier de la section «Structure de l'objet métier du connecteur», à la page 28 ont été mises à jour.

- Les informations de la section «ASI de l'attribut», à la page 31, sur les attributs d'objet métier et leurs propriétés y compris les informations spécifiques à l'application (propriété `AppSpecificInfo`), ont été mises à jour.
- Des informations sur l'exécution des exemples de fichier du bean d'entreprise `MusicCart` figurent dans la section «Exécution de l'adaptateur pour appeler des exemples d'objets métier», à la page 35.
- Les informations sur la variable `CLIENT JARPATH` du fichier de démarrage ont été mises à jour dans la section «Démarrage de l'ODA», à la page 39.

Décembre 2003 :

- La version 1.0.x est la première édition de l'adaptateur pour EJB.

Chapitre 1. Présentation

Ce chapitre présente le connecteur pour Enterprise JavaBeans (EJB) Architecture. Il contient les sections suivantes :

Le connecteur pour l'architecture EJB est un composant d'exécution de WebSphere Business Integration Adapter for Enterprise JavaBeans Architecture. L'adaptateur pour architecture EJB contient un connecteur, des fichiers de message, des outils de configuration et un Object Discovery Agent (ODA). Le connecteur autorise le courtier d'intégration WebSphere à échanger des objets métier avec les beans d'entreprise conçus avec l'architecture Enterprise JavaBeans (EJB) et déployés sur un serveur d'applications. Pour ce faire, le connecteur active les processus d'entreprise dans le courtier, de façon à transmettre les données à une ou plusieurs méthodes de bean d'entreprise, et à recevoir les données retournées. L'adaptateur est unidirectionnel dans le sens qu'il fournit uniquement la fonctionnalité de traitement des requêtes. Il ne procède pas à la notification des événements.

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

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

Ce document contient des informations 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.

Environnement de l'adaptateur

Avant d'installer, de configurer et d'utiliser l'adaptateur, vous devez connaître les spécifications inhérentes à son environnement. En complément des indications ci-dessous, les caractéristiques matérielles et logicielles requises sont indiquées dans le document technique disponible à l'adresse :

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

Compatibilité du courtier

L'architecture qu'utilise l'adaptateur doit être compatible avec la version du courtier d'intégration avec lequel l'adaptateur communique. La version 1.2.x de l'adaptateur pour l'architecture Enterprise JavaBeans est prise en charge sur les versions de l'architecture de l'adaptateur et les courtiers d'intégration suivants :

- Architecture de l'adaptateur :
  - WebSphere Business Integration Adapter Framework, version 2.6.0.3, 2.6.0
- Courtiers d'intégration :
  - WebSphere InterChange Server, version 4.2.2, 4.3.x
  - WebSphere MQ Integrator, version 2.1, 5.0
  - 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 les composants requis, voir les documents suivants.

Pour WebSphere InterChange Server (ICS), voir *System Installation Guide for UNIX* ou *for Windows*.

Pour les courtiers de messages (WebSphere MQ Integrator Broker, WebSphere MQ Integrator et WebSphere Business Integration Message Broker), voir *Implementing Adapters with WebSphere Message Brokers*, et la documentation d'installation du courtier de messages. Certaines de ces informations sont disponibles à l'adresse :

http://www.ibm.com/software/integration/mqfamily/library/manualsa/

Pour WebSphere Application Server, voir Implementing Adapters with WebSphere Application Server et la documentation disponible à l'adresse

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

Standards de l'adaptateur

L'adaptateur est écrit d'après la spécification Enterprise JavaBeans 2.1. Il est donc compatible avec un serveur d'applications J2EE basé sur ce standard, ce qui est le cas de WebSphere Application Server, version 5.x.

L'adaptateur prend en charge des beans d'entité et des beans de session. Les beans de messages ne sont pas pris en charge dans cette édition.

L'adaptateur est conforme à Java Authentication and Authorization Service (JAAS), édition 1.0.

Pour plus d'informations sur l'architecture logicielle EJB, voir http://java.sun.com/products/ejb/

Plateformes de l'adaptateur

L'adaptateur fonctionne sur les platesformes suivantes :

- Windows 2000
- Solaris 8, 9
- HP-UX 11i
- AIX 5.2
- Linux RedHat 3.0

Dépendances de l'adaptateur

L'ODA de l'adaptateur pour architecture EJB exige le fichier `j2ee.jar`, fourni avec l'adaptateur.

Données dépendant des paramètres nationaux

Le connecteur a été internationalisé, de sorte qu'il gère l'envoi de jeux de caractères à deux octets à l'interface EJB (laquelle est compatible avec ces jeux), et peut transmettre le texte du message dans la langue indiquée. Lorsque l'adaptateur transfère des données entre deux emplacements qui utilisent des jeux différents de codes de caractères, il effectue la conversion des caractères afin de préserver les informations.

L'environnement d'exécution Java de la machine virtuelle Java (JVM) représente les données dans le jeu de codes de caractères Unicode. Le format Unicode contient des codes pour les caractères présents dans la plupart des jeux de codes de caractères connus (à la fois mono-octet et multi-octets). La plupart des composants du système d'intégration WebSphere sont écrits en Java. Par conséquent, lorsque des données sont transférées ces composants d'intégration, la conversion des caractères est inutile.

Terminologie

Les termes suivants sont utilisés dans ce guide :

- **ASI (Application-Specific Information)** Métadonnées adaptées à une application ou à une technologie particulière. L'ASI existe au niveau de l'attribut et de l'objet métier. Voir également *ASI d'instruction*.
- **BO (objet métier)** Ensemble d'attributs représentant une entité de l'entreprise (telle que les employés) et une action sur les données (telle que l'opération de création ou de mise à jour). Les composants du système d'intégration WebSphere utilisent des objets métier pour échanger des informations et déclencher des actions.
- **Gestionnaire de BO (objet métier)** Composant de connecteur contenant des méthodes qui interagissent avec une application et qui transforment les objets métier de requête en opérations de l'application.
- **Descripteur de déploiement** Fichier XML regroupé dans le fichier JAR (Java Archive), qui fournit des informations structurelles sur l'ensemble de l'application. Il indique quelles classes composent les beans dans le fichier JAR, comment les beans doivent être déployés et comment chaque bean doit être géré lors de l'exécution. Lorsqu'un bean d'entreprise est déployé sur un serveur d'applications, le descripteur de déploiement est lu et les propriétés sont affichées en vue de leur modification. La personne chargée du déploiement peut alors modifier et ajouter des paramètres en fonction des besoins. Lorsqu'elle est

satisfaite des informations de déploiement, elle les utilise pour générer l'ensemble de l'infrastructure de prise en charge nécessaire pour déployer le bean d'entreprise sur le serveur.

- **Conteneur EJB** Gère les interactions entre un bean et le serveur d'applications, en offrant des services de gestion du cycle de vie, de sécurité, de déploiement et d'exécution. Cette entité par programmation réside sur le serveur d'applications et en tant que telle, gère la mise en mémoire cache du côté du serveur plutôt que du côté du client. Le conteneur isole le bean d'entreprise de l'accès direct des applications clients. Lorsqu'une application client appelle une méthode éloignée sur un bean d'entreprise, le conteneur intercepte d'abord l'appel pour s'assurer que la persistance, les transactions et la sécurité sont appliquées correctement à chaque opération effectuée par le client sur le bean. Lorsqu'un bean n'est pas utilisé, le conteneur le place dans un pool afin qu'il puisse être réutilisé par un autre client, ou il l'efface de la mémoire tandis que sa référence éloignée sur le client reste intacte, pour pouvoir la réinsérer si nécessaire. Lorsque le client appelle une méthode sur la référence éloignée, le conteneur EJB réactive simplement le bean pour répondre à la requête, l'opération restant transparente du côté de l'application client. Un bean d'entreprise ne peut fonctionner hors d'un conteneur EJB.
- **Objet local EJB** Objet grâce auquel les opérations de cycle de vie du bean d'entreprise (create, remove, find) peuvent être réalisées. L'objet local EJB implémente l'interface locale du bean d'entreprise, et la classe de l'objet local EJB est générée par les outils de déploiement du conteneur. Le client utilise l'interface JNDI (Java Naming and Directory Interface (JNDI) pour localiser un objet local EJB. Ensuite, le client le référence pour exécuter des opérations de cycle de vie sur un objet EJB.
- **Bean d'entreprise** Composant Enterprise JavaBeans constitué d'une *interface locale*, d'une *interface éloignée* et d'une *classe de bean*. L'interface locale représente les méthodes de cycle de vie du composant (create, destroy, find), tandis que l'interface éloignée représente les méthodes du bean. La classe de bean implémente les méthodes métier définies dans l'interface éloignée, c'est par conséquent un élément clé du bean.

  Les beans d'entreprise sont regroupés dans un fichier JAR, déployés sur un serveur d'applications et gérés par un conteneur EJB. Un fichier JAR EJB contient un ou plusieurs beans d'entreprise et un descripteur de déploiement. Un bean d'entreprise ne peut fonctionner hors d'un conteneur EJB.
- **Classe de bean d'entreprise** Implémente les méthodes d'un bean d'entreprise.
- **Enterprise JavaBeans (EJB)** Permet de développer de façon rapide et simple des applications Java d'entreprise orientées objet, réparties, sécurisées et portables. La spécification EJB impose un modèle de programmation, c'est-à-dire des conventions ou protocoles et l'ensemble de classes et d'interfaces qui composent l'API EJB. Le modèle de programmation EJB procure aux développeurs de beans d'entreprise et de serveurs une plateforme de développement commune.
- **Bean d'entité** Bean qui modélise un concept d'entreprise pouvant être exprimé par un nom. `Customer`, `Item` et `Vendor` sont des exemples de beans d'entité puisqu'ils modélisent des objets réels. Ces objets sont habituellement des enregistrements persistants, figurant dans une base de données. Voir également *bean de session*.
- **Clé étrangère** Attribut simple dont la valeur identifie de façon unique un objet métier enfant. En général, cet attribut identifie un objet métier enfant auprès de son parent, en contenant la valeur de clé primaire de l'enfant. Le connecteur de l'architecture EJB utilise la clé étrangère pour préciser les objets de connexion organisables en pool.

- **Interface locale** Définit les méthodes de cycle de vie du bean : méthodes de création, suppression et recherche dans l'interface éloignée du bean et méthodes métier locales non spécifiques à une instance de bean particulière.
- **Java Authentication and Authorization Service (JAAS)** Architecture de sécurité qui permet à des services d'authentifier et d'appliquer des contrôles d'accès en fonction de l'identité de l'utilisateur. Elle implémente une version Java de l'architecture Pluggable Authentication Module (PAM) standard, et prend en charge des droits d'accès en fonction de l'utilisateur. L'autorisation JAAS permet d'accorder des droits d'accès non seulement basés sur le type de code exécuté, mais aussi en fonction de qui l'exécute.
- **Java Naming and Directory Interface (JNDI)** Extension Java standard qui fournit une API uniforme permettant d'accéder à un choix de services plus large. Le connecteur pour l'architecture EJB utilise l'interface JNDI pour localiser les beans d'entreprise déployés sur un serveur d'applications d'entreprise. Le serveur doit prendre en charge l'interface JNDI en organisant les beans d'entreprise dans une arborescence, et en fournissant un pilote JNDI nommé fournisseur de services pour accéder à cette arborescence.
- **ODA (Object Discovery Agent)** Outil qui génère automatiquement une définition d'objet métier en examinant les entités spécifiées dans l'application et en "reconnaissant" les éléments de ces entités qui correspondent aux attributs d'objet métier. Lorsque vous installez l'adaptateur, l'ODA est automatiquement installé. Business Object Designer fournit une interface graphique pour accéder à l'ODA et l'utiliser de façon interactive.
- **Pool d'objet par appel** Entité par programmation dédiée au stockage des objets qui doivent passer d'une méthode à la suivante au cours d'un même appel de méthode `doVerbFor`. Les objets conservés peuvent être des objets (enfants ou non enfants) ou des attributs simples.
- **Interface éloignée** Définit les méthodes métier du bean susceptibles d'être appelées par un client, c'est-à-dire les méthodes présentées par un bean au monde extérieur pour exécuter ses travaux. Une interface éloignée est centrée sur les actions métier et n'inclut pas de méthode pour les opérations système telles que persistance, sécurité, simultanéité et transactions. Pendant le traitement, le connecteur accède au bean d'entreprise par le biais de sa définition d'objet éloignée ou de son mandataire.
- **Bean de session** Bean d'entreprise créé par un client en tant qu'extension de son application. Un bean de session est responsable de la gestion des tâches et processus du client, tels que calculs ou accès à une base de données, et n'existe que pour la durée d'une même session client/serveur. Bien qu'un bean de session puisse être transactionnel, il n'est pas récupérable en cas de panne du système. Les objets de bean de session peuvent être sans état ou maintenir un état conversationnel entre les méthodes et les transactions. Si une session maintient un état, le conteneur EJB le gère si l'objet doit être supprimé de la mémoire. Toutefois, l'objet de bean de session lui-même doit gérer ses propres données persistantes. Voir également *bean d'entité*.
- **ASI (application-specific information) d'instruction** Pour une instruction donnée, l'ASI d'instruction précise de quelle façon le connecteur doit traiter l'objet métier lorsque l'instruction est active. L'ASI de l'instruction contient le nom de la méthode à appeler pour traiter l'objet métier de requête en cours.

Traitement des requêtes dans l'adaptateur de l'architecture EJB

Cette section explique comment le connecteur pour architecture EJB gère le traitement des requêtes, comme indiqué dans la figure 1.

Le scénario de traitement des requêtes décrit ici suppose que :

- Les beans d'entreprise sont déployés sur un serveur d'applications d'entreprise conforme J2EE, tel que WebSphere Application Server.
- Le serveur d'applications est installé et en cours d'exécution.
- Le connecteur a été initialisé, ce qui a pour effet de charger les propriétés `ProviderURL` et `InitialContextFactory` spécifiques au connecteur. Ces propriétés aident à obtenir le contexte initial de l'interface JNDI, exigé pour démarrer la connexion au serveur EJB et localiser l'interface locale du bean d'entreprise. Les propriétés sont mises en mémoire cache et réutilisées durant tout le cycle de vie du connecteur. Pour plus d'informations sur les propriétés spécifiques au connecteur, voir «Propriétés spécifiques au connecteur», à la page 16. Pour savoir comment le connecteur gère la sécurité, voir «Configuration de la sécurité», à la page 21.

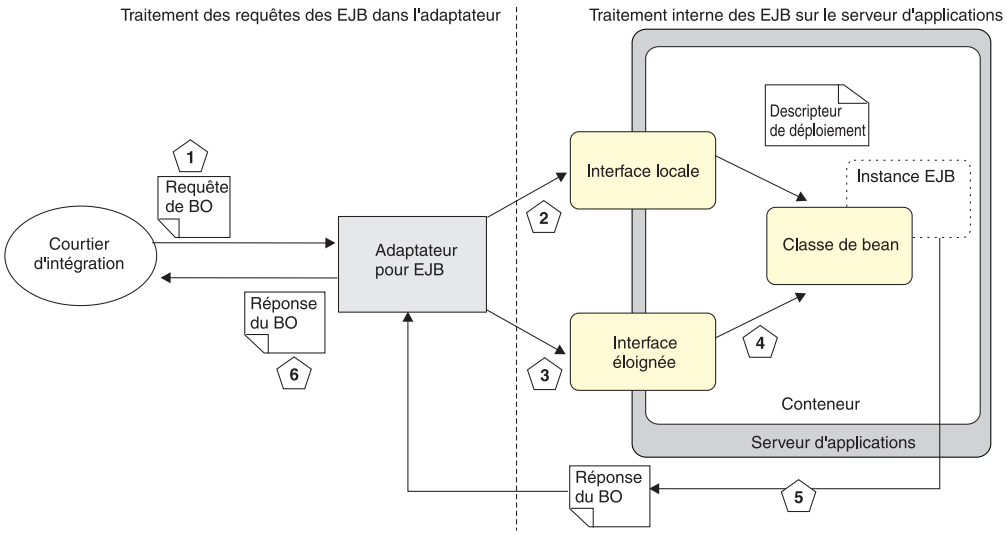


Figure 1. Traitement de requête du connecteur pour l'architecture EJB

Le connecteur pour architecture EJB traite les requêtes d'objet métier de la façon suivante :

1. Le connecteur reçoit une requête d'objet métier du courtier d'intégration. L'objet métier contient les informations relatives au bean d'entreprise correspondant, y compris son nom JNDI et le nom de classe de ses interfaces locale et éloignée.
2. Le connecteur utilise ses propriétés `InitialContextFactory` et `ProviderURL` pour commencer le processus de localisation et d'accès aux beans d'entreprise déployés sur le serveur d'intégration. `InitialContext` fait partie d'une API JNDI plus large, prise en charge par le serveur d'applications. Il s'agit du point de départ de toute recherche JNDI réalisée par un client (tel que le connecteur). `ProviderURL` précise le fournisseur de service, qui est un pilote JNDI du serveur d'applications utilisé pour localiser l'interface locale du bean d'entreprise. L'interface locale offre des méthodes de création et de recherche de l'interface éloignée du bean d'entreprise.

3. Après avoir situé l'interface locale, le connecteur recherche l'interface éloignée qui détermine les méthodes métier du bean d'entreprise qu'un client (dans ce cas, le connecteur) peut appeler.

   Pour les beans d'entité, le connecteur utilise une méthode de type creator ou finder, en fonction de celle qui a été implémentée par le développeur de bean d'entreprise (le connecteur sait lequel utiliser d'après les métadonnées de la définition de l'objet métier. Ces méthodes sont définies dans l'interface locale et appelées par un client pour obtenir une référence d'interface locale/éloignée à une instance de bean d'entreprise.
4. Une fois que le connecteur a localisé l'interface éloignée, il peut commencer à appeler des méthodes de bean d'entreprise. L'objet métier parent envoyé par le connecteur contient un objet métier enfant pour chaque méthode définie dans l'interface éloignée. Les attributs de l'objet métier enfant sont mappés sur les paramètres de la méthode éloignée des beans d'entreprise correspondants. Les beans d'entreprise étant chargés de façon dynamique, les méthodes sont reconnues et appelées par le biais de réflexions. Voici les détails de cette étape :
   - Le gestionnaire BO du connecteur recherche une ou plusieurs listes de noms d'attributs dans l'ASI de l'instruction de l'objet métier parent. Une liste est une série ordonnée de noms d'attributs, séparés par un point-virgule.
   - Chaque attribut de l'objet métier parent contient un objet métier enfant représentant une méthode à appeler sur l'interface éloignée. En d'autres termes, l'ASI de l'instruction n'est pas une liste de méthodes, mais une liste d'attributs ayant chacun un objet enfant qui représente une méthode à appeler.
   - Le connecteur exécute les méthodes indiquées dans l'ASI de l'instruction, dans l'ordre dans lequel elles apparaissent dans la liste.
5. Une fois les méthodes exécutées et les valeurs retournées par le serveur d'applications EJB, le connecteur charge les données d'objet EJB dans l'objet métier.
6. Le connecteur retourne l'objet métier au courtier d'intégration, avec des valeurs complétées à partir du serveur d'applications EJB.

   Le connecteur retourne également un message au courtier d'intégration, indiquant si la requête d'objet d'origine a réussi ou non (statut FAIL). Si la requête a réussi, le connecteur retourne également au courtier l'objet métier mis à jour.

Prise en charge de la transaction EJB

Notez que le connecteur ne prend pas en charge les appels de transaction des méthodes EJB. Si un processus métier exige des transactions, créez un bean d'encapsuleur léger qui expose une méthode non transactionnelle au connecteur, mais qui appelle en interne les méthodes EJB cibles dans le cadre d'une transaction du serveur d'applications.

Sécurité de l'adaptateur de l'architecture EJB

Bien qu'il s'agisse d'une fonctionnalité facultative, lorsque la sécurité EJB est configurée dans le connecteur, elle exige que le connecteur fournisse des données d'authentification et de contrôle d'accès avant de pouvoir accéder aux beans sécurisés déployés sur le serveur. Le connecteur pour architecture EJB utilise Java Authentication and Authorization Service (JAAS) pour mettre en oeuvre la sécurité EJB.

JAAS est une architecture de sécurité qui permet à des services d'authentifier et appliquer des contrôles d'accès en fonction de l'identité de l'utilisateur. Elle implémente une version Java de l'architecture Pluggable Authentication Module (PAM) standard et prend en charge des droits d'accès en fonction de l'utilisateur. L'autorisation JAAS permet d'accorder des droits d'accès, non seulement en fonction du type de code exécuté mais également en fonction de qui l'exécute.

L'authentification s'effectue de façon connectable. Ceci permet aux applications Java de rester indépendantes des technologies d'authentification sous-jacentes. Les technologies d'authentification, nouvelles ou mises à jour, peuvent être connectées à une application sans qu'il ne soit nécessaire de modifier l'application elle-même.

Pour plus d'informations sur la sécurité, voir «Configuration de la sécurité», à la page 21.

Traitement des instructions

Le connecteur traite les objets métier qui lui sont transmis par un courtier en fonction de l'instruction de chaque objet métier.

Lorsque l'architecture du connecteur reçoit une requête du courtier, elle appelle la méthode `doVerbFor()` de la classe du gestionnaire d'objet métier associée à la définition d'objet métier de l'objet de la requête. Le rôle de la méthode `doVerbFor()` est de déterminer le traitement d'instruction a effectuer, en fonction de l'instruction active de l'objet métier de requête. Il obtient les informations auprès de l'objet métier de requête pour concevoir et envoyer des requêtes d'opération à l'application.

Lorsque l'architecture du connecteur transmet l'objet métier de requête à `doVerbFor()`, cette méthode extrait l'ASI d'objet métier et appelle le gestionnaire de BO, qui lit à son tour l'ASI de l'instruction et la traduit en une série de fonctions appelables. L'ASI d'instruction est une liste ordonnée des méthodes qui doivent être appelées pour cette instruction. L'ordre dans lequel les appels sont lancés est essentiel à la réussite de traitement de l'objet.

Si l'ASI de l'instruction est vide pour un bean d'entreprise éloigné, alors le gestionnaire de BO recherche une méthode creator (`Create()`) sans argument dans l'interface locale, l'appelle puis appelle la première méthode dont les paramètres sont renseignés. L'ASI d'attribut de ce constructeur doit être `method_name=create`.

Si l'ASI d'instruction est vide pour un objet Java, le gestionnaire de BO recherche un constructeur et une méthode dont les paramètres sont renseignés. L'ASI d'attribut du constructeur doit être `method_name=CONSTRUCTOR`.

Sur un bean d'entreprise éloigné et un objet Java, une seule méthode peut être renseignée. Dans le cas contraire, si l'ASI d'instruction est vide, le connecteur consigne une erreur et retourne un code FAIL. Pour plus d'informations sur le traitement des erreurs, voir «Gestion des erreurs», à la page 55. Pour plus d'informations sur l'ASI d'instruction, voir «ASI de l'instruction», à la page 30.

Le connecteur ne prend pas en charge d'instruction spécifique, mais l'ODA permet à l'utilisateur de configurer des instructions personnalisées. Les instructions standard déjà présentes sont Create, Retrieve, Update et Delete. Vous pouvez leur donner la signification sémantique de votre choix via l'Object Discovery Agent (ODA) exécuté dans Business Object Designer. Pour plus d'informations sur l'utilisation de l'ODA pour affecter une séquence d'appels de méthode à une instruction, voir Chapitre 5, «Création et modification des objets métier», à la page 39.

Traitement du gestionnaire de données

L'utilisation du gestionnaire de données est facultative. Si vous avez intégré un gestionnaire de données à l'architecture de votre connecteur, celui-ci doit accéder à n'importe quelle classe de gestionnaire de données nécessaire pour convertir des valeurs d'objet métier en paramètres de bean d'entreprise (tel qu'indiqué dans l'ASI EJB). Une méthode de bean d'entreprise peut choisir un document XML, EDI ou un autre type de document pris en charge par le gestionnaire de données WBI en tant qu'argument vers une méthode EJB éloignée.

Lorsque le connecteur reçoit un objet métier, il évalue l'ASI de l'objet métier pour déterminer si l'objet métier doit être converti en données à l'aide du gestionnaire de données. L'ASI de l'objet métier d'un message pris en charge par un gestionnaire de données doit contenir la valeur `object_type=dataHandlerObject; mime_type=<text_value>`, où `<text_value>` est le type mime correct défini pour le gestionnaire de données (tel que précisé dans le méta-objet du gestionnaire de données) que l'adaptateur doit utiliser pour convertir les données.

Si le connecteur trouve un objet métier de méthode avec pour paramètre un document pris en charge par le gestionnaire de données, il appelle le gestionnaire de données pour convertir l'objet métier dans le document correspondant. Ensuite, il appelle une méthode de bean d'entreprise éloignée en transmettant le document généré par le gestionnaire de données en tant qu'argument à la méthode. De même, si une méthode retourne un document qui doit être traité avec le gestionnaire de données, le connecteur convertit la chaîne retournée par la méthode en un objet métier, à l'aide des valeurs du gestionnaire de données. Par exemple, si une méthode de bean d'entreprise retourne un document XML ou EDI, le gestionnaire de données doit être appelé pour le convertir en un objet métier enfant.

Pour prendre en charge un gestionnaire de données, vous devez configurer la propriété `DataHandlerConfigMO` spécifique au connecteur. Pour plus d'informations sur cette propriété et d'autres qui sont spécifiques au connecteur, voir «Propriétés spécifiques au connecteur», à la page 16. Pour plus d'informations sur le développement d'un gestionnaire de données, voir *Data Handler Guide*.

Chapitre 2. Installation de l'adaptateur

Le présent chapitre explique comment installer le connecteur. Il contient les sections suivantes :

- «Présentation des tâches d'installation»
- «Structure de fichiers du connecteur»
- «Après l'installation», à la page 13

Présentation des tâches d'installation

Pour installer l'adaptateur pour EJB, vous devez vérifier que les conditions requises par l'adaptateur sont satisfaites par votre environnement, installer le courtier d'intégration et exécuter l'installation de l'adaptateur.

Vérification des conditions requises par l'adaptateur

Avant d'installer l'adaptateur, vérifiez que toutes les conditions d'environnement requises pour l'installation et l'exécution de l'adaptateur sont réunies sur votre système. Pour plus d'informations, voir «Environnement de l'adaptateur», à la page 1.

Installation du courtier d'intégration

L'installation du courtier d'intégration, qui comprend l'installation du système d'intégration WebSphere et le démarrage du courtier, est décrite dans la documentation de votre courtier. Pour plus d'informations sur les courtiers pris en charge par le connecteur pour EJB, voir «Compatibilité du courtier», à la page 2.

Pour plus d'informations sur l'installation du courtier, voir la documentation d'implémentation du courtier que vous utilisez.

Installation de l'adaptateur pour EJB et des fichiers associés

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

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

Structure de fichiers du connecteur

L'installation du connecteur copie sur votre système les fichiers standard associés. Elle installe le connecteur dans le répertoire *ProductDir*\connectors\EJB et ajoute un raccourci dans le menu Démarrer. Notez que *ProductDir* est le répertoire d'installation du connecteur.

Le tableau 1, à la page 12 décrit la structure de fichiers utilisée par le connecteur, et indique les fichiers qui sont automatiquement installés lorsque vous choisissez d'installer le connecteur à l'aide du programme d'installation.

Tableau 1. Structure de fichier du connecteur

Sous-répertoire de *ProductDir*	**Description**
\connectors\EJB\BIA_EJB.jar	Contient uniquement les classes utilisées par le connecteur EJB
\connectors\EJB\start_EJB.bat	Script de démarrage du connecteur générique (Windows 2000)
\connectors\EJB\start_EJB.sh	Script de démarrage du connecteur générique (Unix)
\connectors\messages\BIA_EJBConnector.txt	Fichier de message du connecteur
\connectors\EJB\samples\BIA_EJBConnector.cfg	Exemple de fichier de configuration du connecteur EJB. Pour plus d'informations sur l'utilisation de ce fichier, voir «Exécution de l'adaptateur pour appeler des exemples d'objets métier», à la page 35.
\connectors\EJB\samples\BIA_PortConnector.cfg	Exemple de fichier de configuration du connecteur de port. Pour plus d'informations sur l'utilisation de ce fichier, voir «Exécution de l'adaptateur pour appeler des exemples d'objets métier», à la page 35.
\connectors\EJB\samples\SampleBOs\	Exemple de définitions d'objet métier. Pour plus d'informations sur l'utilisation de ces fichiers, voir «Exécution de l'adaptateur pour appeler des exemples d'objets métier», à la page 35.
\connectors\EJB\samples\SampleMusicCartEJB\src	Fichiers source EJB des exemples de beans. Pour plus d'informations sur l'utilisation de ces exemples, voir «Exécution de l'adaptateur pour appeler des exemples d'objets métier», à la page 35.
\connectors\EJB\samples\SampleMusicCartEJB\ BIA_MusicBeanSample.jar	Fichier JAR EJB qui contient les beans d'entreprise déployés sur le serveur d'applications lors de l'exécution des fichiers exemples. Pour plus d'informations sur l'utilisation de ces exemples, voir «Exécution de l'adaptateur pour appeler des exemples d'objets métier», à la page 35.
\repository\EJB\EJBConnectorTemplate	Modèle de fichier pour la définition du connecteur.
\connectors\EJB\dependencies\j2ee.jar	Fichier JAR requis par l'ODA EJB.
\ODA\EJB\BIA_EJBODA.jar	ODA EJB
\ODA\EJB\start_EJBODA.bat	Fichier de démarrage de l'ODA (Windows 2000)
\ODA\EJB\start_EJBODA.sh	Fichier de démarrage de l'ODA (Unix)
\ODA\messages\BIA_EJBODAAgent.txt	Fichier de message de l'ODA
\ODA\messages\BIA_EJBODAAgent_de_DE.txt	Fichier de message de l'ODA (chaînes de texte en allemand)
\ODA\messages\BIA_EJBODAAgent_en_US.txt	Fichier de message de l'ODA (chaînes de texte en anglais US)
\ODA\messages\BIA_EJBODAAgent_es_ES.txt	Fichier de message de l'ODA (chaînes de texte en espagnol)
\ODA\messages\BIA_EJBODAAgent_fr_FR.txt	Fichier de message de l'ODA (chaînes de texte en français)
\ODA\messages\BIA_EJBODAAgent_it_IT.txt	Fichier de message de l'ODA (chaînes de texte en italien)
\ODA\messages\BIA_EJBODAAgent_ja_JP.txt	Fichier de message de l'ODA (chaînes de texte en japonais)
\ODA\messages\BIA_EJBODAAgent_ko_KR.txt	Fichier de message de l'ODA (chaînes de texte en coréen)
\ODA\messages\BIA_EJBODAAgent_pt_BR.txt	Fichier de message de l'ODA (chaînes de texte en portugais brésilien)
\ODA\messages\BIA_EJBODAAgent_zh_CN.txt	Fichier de message de l'ODA (chaînes de texte en chinois simplifié)
\ODA\messages\BIA_EJBODAAgent_zh_TW.txt	Fichier de message de l'ODA (chaînes de texte en chinois traditionnel)

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

Après l'installation

Une fois l'adaptateur installé, et avant de l'exécuter, vous devez le configurer. Pour plus d'informations, voir Chapitre 3, «Configuration de l'adaptateur», à la page 15.

Chapitre 3. Configuration de l'adaptateur

Le présent chapitre explique comment configurer l'adaptateur. Il contient les sections suivantes :

Présentation des tâches de configuration

Après l'installation et avant le démarrage, vous devez configurer le connecteur et configurer vos objets métier.

- **Configuration du connecteur**: Comprend configuration et paramétrage. Pour plus d'informations, voir «Configuration du connecteur».
- **Configuration des objets métier** : Réalisée via un ODA (Object Discovery Agent). L'ODA permet de générer des définitions d'objet métier. Une définition d'objet métier est un modèle d'objet métier. L'ODA examine les objets d'application spécifiés, "reconnaît" les éléments de ces objets qui correspondent aux attributs d'objet métier, et génère des définitions d'objet métier pour représenter les informations. Business Object Designer apporte une interface graphique permettant d'accéder à l'Object Discovery Agent et de l'utiliser de façon interactive.

  Pour plus d'informations sur l'utilisation de l'ODA, voir Chapitre 5, «Création et modification des objets métier», à la page 39.

Configuration du connecteur

Les connecteurs ont deux types de propriétés de configuration : les propriétés standard et les propriétés spécifiques. Vous devez définir les valeurs de ces propriétés à l'aide de Connector Configurator avant d'exécuter le connecteur. Pour plus d'informations, voir l'Annexe B, «Connector Configurator», à la page 85.

Un connecteur obtient ses valeurs de configuration lors du démarrage. En cours d'exécution, vous pourrez avoir besoin de changer la valeur d'une ou de plusieurs propriétés. Les modifications apportées à certaines propriétés de configuration, telles que `AgentTraceLevel`, sont a effet immédiat. Les modifications apportées aux autres propriétés du connecteur exigent de redémarrer le système ou le composant du connecteur. Pour déterminer si une propriété est dynamique (à effet immédiat) ou statique (exigeant le redémarrage du système ou d'un composant de connecteur), voir la colonne Update Method de la fenêtre Connector Properties de System Manager.

Propriétés standard du connecteur

Les propriétés standard de configuration fournissent des informations destinées aux connecteurs. Voir l'Annexe A, «Propriétés de configuration standard pour les connecteurs», à la page 59 pour plus d'informations sur ces propriétés.

Notez que, bien que les propriétés suivantes figurent dans l'Annexe A, «Propriétés de configuration standard pour les connecteurs», à la page 59, le connecteur pour EJB *ne les utilise pas* :

- `DuplicateEvent Elimination`
- `PollEndTime`
- `PollFrequency`
- `PollStartTime`

Notez également certaines informations concernant les propriétés ci-dessous :

- `Locale` : Ce connecteur est internationalisé, vous pouvez donc modifier la valeur de la propriété `Locale`
- `ApplicationName`: Avant d'exécuter le connecteur, vous devez indiquer une valeur pour la propriété de configuration `ApplicationName`.

Propriétés spécifiques au connecteur

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

Pour configurer les propriétés spécifiques au connecteur, utilisez Connector Configurator. Cliquez sur l'onglet **Application Config Properties** pour ajouter ou modifier les propriétés de configuration. Pour plus d'informations, voir l'Annexe B, «Connector Configurator», à la page 85.

Le tableau 2 dresse la liste des propriétés de configuration spécifiques au connecteur, ainsi que leurs descriptions et leurs valeurs possibles. Notez que ces propriétés sont toutes des chaînes non hiérarchiques. Voir les sections qui suivent pour obtenir des informations sur les propriétés, y compris une image des propriétés en figure 2, à la page 18.

Tableau 2. Propriétés de configuration spécifiques au connecteur

Nom	Valeurs possibles	Valeur par défaut
InitialContextFactory	Nom de classe de la classe d'objets de contexte initiale.	`com.ibmwebsphere.naming.WsnInitialContextFactory`
ProviderURL	URL du fournisseur de service JNDI. Le fournisseur de services est un pilote, sur le serveur d'applications, qui permet d'accéder aux répertoires du serveur dans lesquels les beans d'entreprise sont conservés.	`corbaloc:iiop:localhost:2809`

Tableau 2. Propriétés de configuration spécifiques au connecteur (suite)

Nom	Valeurs possibles	Valeur par défaut
DataHandlerConfigMO	Méta-objet du gestionnaire de données. Utilisé pour prendre en charge un gestionnaire de données, si vous en avez défini un pour le connecteur.	`MO_DataHandler_Default`
LoginConfiguration	Nom de la classe LoginModule de la sécurité JAAS.	La valeur par défaut est déterminée par votre serveur d'applications. Pour WebSphere Application Server, la classe est `WSLogin`. Vous ne pouvez utiliser cette propriété que si votre serveur d'applications prend en charge JAAS et si vous choisissez d'implémenter des beans d'accès sécurisé.
CallBackHandlerClass	L'interface CallBackHandler implémentée par l'utilisateur est utilisée pour la sécurité JAAS.	Pour WebSphere Application Server, la classe est `com.ibm.websphere.security.auth.callback.WSCallbackHandlerImpl` Vous ne pouvez utiliser cette propriété que si votre serveur d'applications prend en charge JAAS et si vous choisissez d'implémenter des beans d'accès sécurisé.
JAASUserName	Nom d'utilisateur de la sécurité JAAS.	Aucune Ne configurez cette propriété que si votre serveur d'applications prend en charge JAAS et si vous choisissez d'implémenter des beans d'accès sécurisé.
JAASPassword	Mot de passe de sécurité JAAS.	Aucune Ne configurez cette propriété que si votre serveur d'applications prend en charge JAAS et si vous choisissez d'implémenter des beans d'accès sécurisé.
JAASRealm	Nom de domaine de sécurité JAAS.	Aucune Ne configurez cette propriété que si votre serveur d'applications prend en charge JAAS et si vous choisissez d'implémenter des beans d'accès sécurisé.
RetryDuration	Valeur numérique (en secondes)	Aucune. Configurez cette propriété pour limiter le nombre de tentatives de connexion de l'adaptateur au serveur d'applications, en cas de panne. Si cette propriété n'est pas définie, l'adaptateur tentera de se reconnecter au serveur d'applications jusqu'à ce que celui-ci devienne disponible.

La figure 2, à la page 18 illustre les relations hiérarchiques entre les propriétés spécifiques au connecteur.

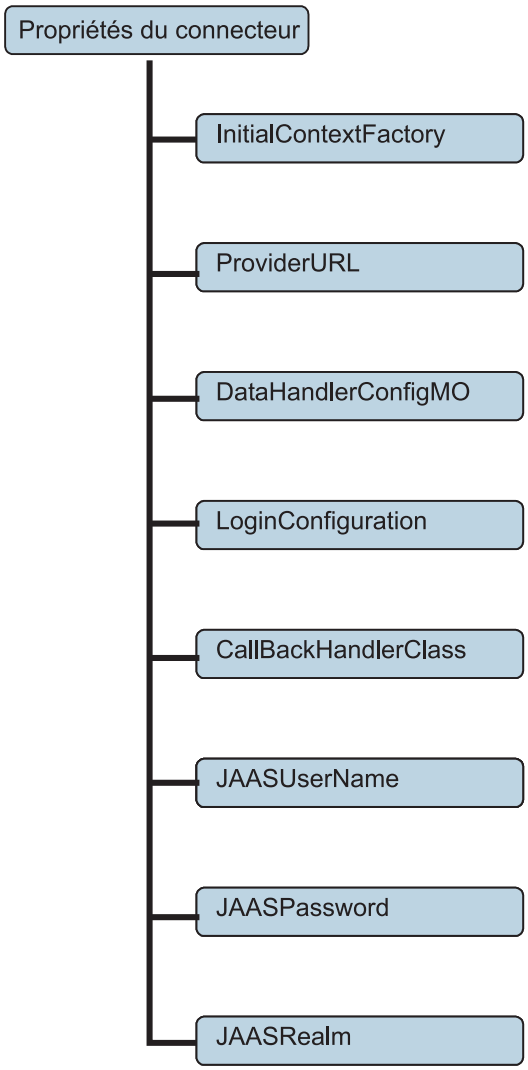


Figure 2. Hiérarchie des propriétés spécifiques au connecteur

InitialContextFactory

Nom de classe de la classe d'objets de contexte initiale.`InitialContext`, une interface JNDI requise pour initialiser la connexion au serveur d'applications d'entreprise et localiser l'interface locale du bean d'entreprise. C'est le point de départ de toute recherche de client d'un bean d'entreprise. Pour obtenir le contexte initial JNDI, le connecteur utilise les propriétés `InitialContextFactory` et `ProviderURL`.

ProviderURL

URL du fournisseur JNDI. JNDI permet au connecteur d'accéder aux beans d'entreprise par le nom. Le connecteur utilise cette URL pour se connecter à distance au serveur JNDI exécuté sur le serveur EJB. Une fois connecté au serveur EJB, le connecteur peut localiser l'interface locale du bean d'entreprise.

DataHandlerConfigMO

Nom du méta-objet du gestionnaire de données de niveau supérieur. Si vous avez intégré un gestionnaire de données à l'architecture de votre connecteur, celui-ci doit accéder à n'importe quelle classe de gestionnaire de données nécessaire pour

convertir des valeurs d'objet métier en paramètres de bean d'entreprise (tel qu'indiqué dans l'ASI EJB). Une méthode de bean d'entreprise peut choisir un document XML, EDI ou un autre type de document pris en charge par le gestionnaire de données WBI, en tant qu'argument vers une méthode EJB éloignée. Si le connecteur trouve un objet métier de méthode avec pour paramètre un document pris en charge par le gestionnaire de données, il appelle le gestionnaire de données pour convertir l'objet métier dans le document correspondant. Ensuite, il appelle une méthode de bean d'entreprise éloignée en transmettant en argument le document généré par le gestionnaire de données.

Si cette propriété n'est pas renseignée, le connecteur ne peut trouver le méta-objet dont il a besoin pour appeler le gestionnaire de données approprié. Pour plus d'informations sur l'utilisation d'un gestionnaire de données avec le connecteur, voir «Traitement du gestionnaire de données», à la page 9.

LoginConfiguration

Classe chargée de l'implémentation de l'interface `LoginModule` du fournisseur technologique d'authentification. Le fournisseur de la technologie d'authentification implémente `LoginModule` pour offrir un type d'authentification particulier par le biais d'un module appelable, sans avoir à modifier l'application en elle-même. En général, l'implémentation de `LoginConfiguration` et de `LoginModule` est fournie dans la classe de connexion du serveur d'applications. Si votre serveur d'applications prend en charge les beans d'accès sécurisé avec JAAS, et si vous choisissez d'implémenter ce service pour la sécurité EJB, définissez cette propriété sur le nom de la classe LoginModule de votre serveur d'applications.

Pour les serveurs d'application compatibles JAAS, le connecteur active le processus d'authentification en instanciant un objet LoginContext, qui précise le LoginModules qui gérera l'authentification JAAS. Au cours de l'authentification du client, le LoginModule demande et vérifie le nom d'utilisateur et le mot de passe, définis dans les propriétés `JAASUserName` et `JAASPassword`.

Bien que JAAS ne soit pas exigé par le connecteur, si vous précisez une valeur dans la propriété `LoginConfiguration`, le connecteur suppose que vous implémentez la sécurité bean, auquel cas les propriétés `CallBackHandlerClass`, `JAASUserName`, `JAASPassword` et `JAASRealm` sont toutes obligatoires.

Pour plus d'informations sur la configuration de la sécurité, voir «Configuration de la sécurité», à la page 21 et «Sécurité de l'adaptateur de l'architecture EJB», à la page 7.

CallBackHandlerClass

Une interface JAAS, déterminée par le serveur d'applications, qui permet à un client de transmettre les données d'authentification au serveur d'applications. Cette interface implémente un gestionnaire CallBack transmis aux services de sécurité sous-jacents, afin qu'ils puissent interagir avec l'application pour extraire du client (dans ce cas, le connecteur) certaines données d'authentification telles que le nom d'utilisateur et le mot de passe. Le `LoginModule` utilise le gestionnaire CallBack pour communiquer avec le client en vue d'obtenir les données d'authentification exigées. Un nom d'utilisateur et un mot de passe pour le connecteur sont définis dans les propriétés `JAASUserName` et `JAASPassword`.

Si votre serveur d'applications prend en charge les beans d'accès sécurisé avec JAAS, et si vous choisissez d'implémenter ce service pour la sécurité EJB, définissez cette valeur sur l'interface `CallBackHandler` implémentée par l'utilisateur.

Bien que JAAS ne soit pas exigé par le connecteur, si vous précisez une valeur dans la propriété LoginConfiguration, le connecteur suppose que vous implémentez la sécurité bean, auquel cas les propriétés CallBackHandlerClass, JAASUserName, JAASPassword et JAASRealm sont toutes obligatoires.

Pour plus d'informations sur la configuration de la sécurité, voir «Configuration de la sécurité», à la page 21 et «Sécurité de l'adaptateur de l'architecture EJB», à la page 7.

JAASUserName

Si votre serveur d'applications prend en charge les beans d'accès sécurisé avec JAAS, et si vous choisissez d'implémenter ce service pour la sécurité EJB, définissez cette valeur sur le nom d'utilisateur JAAS configuré sur votre serveur d'applications.

Bien que JAAS ne soit pas exigé par le connecteur, pour configurer correctement la sécurité avec JAAS vous devez préciser des valeurs dans les propriétés JAASUserName, LoginConfiguration, CallBackHandlerClass, JAASPassword et JAASRealm.

Pour plus d'informations sur la configuration de la sécurité, voir «Configuration de la sécurité», à la page 21 et «Sécurité de l'adaptateur de l'architecture EJB», à la page 7.

JAASPassword

Si votre serveur d'applications prend en charge les beans d'accès sécurisé avec JAAS, et si vous choisissez d'implémenter ce service pour la sécurité EJB, définissez cette valeur sur le mot de passe JAAS configuré sur votre serveur d'applications.

Bien que JAAS ne soit pas exigé par le connecteur, pour configurer correctement la sécurité avec JAAS vous devez préciser des valeurs dans les propriétés JAASPassword, LoginConfiguration, CallBackHandlerClass, JAASUserName et JAASRealm.

Pour plus d'informations sur la configuration de la sécurité, voir «Configuration de la sécurité», à la page 21 et «Sécurité de l'adaptateur de l'architecture EJB», à la page 7.

JAASRealm

Si votre serveur d'applications prend en charge les beans d'accès sécurisé avec JAAS, et si vous choisissez d'implémenter ce service pour la sécurité EJB, définissez cette valeur sur le nom de domaine de sécurité JAAS. Un domaine est un mappage JAAS d'un ou plusieurs User Groups sur un ensemble de privilèges ou droits d'accès.

Bien que JAAS ne soit pas exigé par le connecteur, pour configurer correctement la sécurité avec JAAS vous devez préciser des valeurs dans les propriétés JAASRealm, LoginConfiguration, CallBackHandlerClass, JAASUserName et JAASPassword.

Pour plus d'informations sur la configuration de la sécurité, voir «Configuration de la sécurité», à la page 21 et «Sécurité de l'adaptateur de l'architecture EJB», à la page 7.

RetryDuration

RetryDuration indique en secondes le temps pendant lequel l'adaptateur tentera de se connecter à un serveur Application qui ne répond pas. Par défaut, l'adaptateur de l'architecture EJB tentera plusieurs fois de se reconnecter à un serveur Application qui ne répond pas, jusqu'à ce qu'il finisse par répondre.

Configuration de la sécurité

La sécurité EJB du connecteur est facultative, et n'est disponible que si le serveur d'applications sur lequel les beans d'entreprise sont déployés prend en charge les beans et le protocole de sécurité choisis. Si vous activez la sécurité, l'utilisateur (dans ce cas, le connecteur) doit fournir des données d'authentification au serveur d'applications pour accéder à tout bean déployé. La sécurité ne peut être activée dans le connecteur EJB avec JAAS (Java Authentication and Authorization Service) que si votre serveur d'applications prend en charge cette technologie de sécurité.

Affectation de droits d'accès de méthode aux rôles de sécurité

Le connecteur utilise le contrôle d'accès pour s'assurer que les utilisateurs n'accèdent qu'aux ressources pour lesquelles ils disposent de droits. Le contrôle d'accès applique des règles de sécurité qui réglementent ce que l'utilisateur a le droit ou non de faire dans un système. Les rôles de sécurité des beans d'entreprise du fichier JAR EJB indiquent les méthodes des interfaces locales et éloignées que chaque rôle de sécurité est autorisé à appeler. Le fichier de description du déploiement inclut des balises qui déclarent quels rôles logiques sont autorisés à accéder à quelles méthodes de bean lors de l'exécution.

L'exemple de code suivant montre comment des droits d'accès de méthodes sont affectés aux rôles de sécurité dans le descripteur de déploiement. Notez que pour chaque rôle indiqué dans le descripteur de déploiement, les méthodes qui peuvent être appelées sont groupées par nom de bean.

```
<method-permission>
        <role-name>payroll_department</role-name>
        <method>
                <ejb-name>EmployeeService</ejb-name>
                <method-name>*</method-name>
        </method>
</method-permission>
<method-permission>
        <role-name>employee</role-name>
        <method>
                <ejb-name>AardvarkPayroll</ejb-name>
                <method-name>findByPrimaryKey</method-name>
        </method>
        <method>
                <ejb-name>AardvarkPayroll</ejb-name>
                <method-name>getEmployeeInfo</method-name>
        </method>
        <method>
                <ejb-name>AardvarkPayroll</ejb-name>
                <method-name>updateEmployeeInfo</method-name>
        </method>
</method-permission>
<method-permission>
```

Utilisation du Java Authentication and Authorization Service (JAAS)

Si votre serveur d'applications prend en charge JAAS (Java Authentication and Authorization Service), vous pouvez l'utiliser pour implémenter la sécurité EJB. Lorsque l'utilisateur ou le client (dans ce cas le connecteur) tente d'accéder à un bean sécurisé déployé sur un serveur d'applications, JAAS vérifie l'utilisateur et ses droits d'accès. De cette façon, JAAS apporte un niveau de sécurité supplémentaire au serveur d'applications sur lequel le bean d'entreprise est déployé. JAAS impose de préciser un `LoginContext`, `LoginModule` et d'autres classes J2EE qui décrivent les méthodes basiques utilisées pour authentifier un client. Pour plus d'informations sur ces classes, voir «LoginConfiguration», à la page 19.

Puisque le connecteur pour EJB agit comme un client du serveur, lorsque la sécurité est activée en environnement EJB, le connecteur doit apporter des données d'authentification utilisateur sous la forme d'un ID utilisateur (propriété de configuration `JAASUserName` du connecteur) et d'un mot de passe (propriété de configuration `JAASPassword`). Le conteneur EJB examine l'identité ou le rôle transmis par le connecteur, et le compare avec la liste des objets d'identité associés à la méthode, telle que définie dans le descripteur de déploiement. Si l'identité de l'appelant du connecteur correspond à une identité d'appelant associée à la méthode, alors la méthode peut être invoquée.

L'authentification valide l'identité de l'utilisateur (dans ce cas le connecteur). Une fois que l'utilisateur a franchi le système d'authentification, il est libre d'accéder et d'appeler des méthodes pour lesquelles il dispose d'autorisations.

Les propriétés du connecteur qui contrôlent la sécurité EJB à l'aide de JAAS sont `LoginConfiguration`, `CallBackHandlerClass`, `JAASUserName`, `JAASPassword` et `JAASRealm`. Les valeurs que vous indiquez pour ces propriétés dépendent totalement de votre serveur d'applications. Pour plus d'informations sur l'utilisation de ces propriétés par le connecteur, voir «Propriétés spécifiques au connecteur», à la page 16. Pour plus d'informations sur les valeurs à entrer dans les propriétés, voir la documentation du serveur d'applications.

Configuration de JAAS dans le fichier de démarrage du connecteur

Pour que le connecteur fonctionne avec JAAS, vous devez apporter les modifications suivantes au fichier de démarrage du connecteur (`start_EJB.bat` ou `start_EJB.sh`) :

- Entrez la commande suivante dans le fichier de démarrage, pour préciser le nom et l'emplacement du fichier de configuration JAAS. Cette commande doit être l'une des dernières lignes du fichier.

  ```
  -Djava.security.auth.login.config=login.conf
  ```

  Dans l'exemple de commande fourni ici, le fichier de configuration JAAS porte le nom `login.conf`, qui dépend de l'environnement du serveur d'applications. Pour plus d'informations sur le nom et l'emplacement du fichier de configuration pour votre environnement, voir la documentation du serveur d'applications.
- Pour utiliser l'API JAAS avec le connecteur, supprimez la mise en commentaire de la section `SECURITY_SETTINGS` dans le fichier de démarrage du connecteur.

  Notez que l'API JAAS est disponible dans le fichier `jaas.jar`, fourni par tout serveur d'applications pris en charge par EJB. Veillez à indiquer `jaas.jar` en tant que chemin de classe du connecteur.

Création de plusieurs instances de connecteur

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

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

Création d'un répertoire

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

```
ProductDir\connectors\connectorInstance
```

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

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

```
ProductDir\repository\connectorInstance
```

Création de définitions d'objet métier

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

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

   ```
   ProductDir\repository\initialConnectorInstance
   ```

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

Création d'une définition de connecteur

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

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

Création d'un script de démarrage

Pour créer un script de démarrage, procédez comme suit :

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

   dirname

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

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

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

Configuration du fichier de démarrage

Avant de démarrer le connecteur pour EJB, vous devez configurer le fichier de démarrage du connecteur.

Pour terminer la configuration du connecteur pour les plateformes Windows, vous devez modifier le fichier `start_EJB.bat` :

1. Ouvrez le fichier `start_EJB.bat`.
2. Accédez à la section commençant par "`SET JCLASSES...`"
3. Modifiez la variable `JCLASSES` pour pointer sur le fichier JAR créé par l'ODA. Par exemple, si le fichier JAR créé par l'ODA est `c:\WebSphereAdapters\connectors\EJB\SampleBeans.jar`, alors définissez la variable `JCLASSES` sur :
   ```
   JCLASSES=.;%J_CLASSES%;c:\WebSphereAdapters\connectors\EJB\
   SampleBeans.jar
   ```

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 3.

Tableau 3. 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 [-cconfigFile]`
  - 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.

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.

Utilisation des fichiers de trace et de journalisation

Les composants du connecteur apportent plusieurs niveaux de traçage et de consignation. Le connecteur utilise l'architecture de l'adaptateur pour consigner les messages d'erreur, d'information et de trace. Messages d'information et d'erreur sont enregistrés dans le fichier journal. Les messages de trace et leurs niveaux de trace correspondants (de 0 à 5) sont enregistrés dans un fichier de trace. Pour plus d'informations sur la consignation et les niveaux de trace, voir Chapitre 6, «Identification et résolution des erreurs», à la page 55.

Utilisez Connector Configuration pour configurer les noms des fichiers de trace et de consignation ainsi que le niveau de trace. Pour plus d'informations sur cet outil, voir l'Annexe B, «Connector Configurator», à la page 85.

Notez que l'ODA ne dispose d'aucune capacité de consignation. Les messages d'erreur sont envoyés directement à l'interface utilisateur. Les fichiers de trace et niveaux de trace sont configurés dans Business Object Designer. La procédure est décrite dans «Configuration de l'agent», à la page 41. Les niveaux de trace ODA sont identiques aux niveaux de trace du connecteur, définis dans «Traçage», à la page 56.

Chapitre 4. Présentation des objets métier

Le présent chapitre décrit la structure des objets métier, la façon dont ils sont traités par le connecteur et les suppositions que le connecteur fait à leur sujet.

Définition des métadonnées

Le connecteur pour EJB est contrôlé par des métadonnées. Dans le système d'intégration WebSphere, les métadonnées sont définies comme étant des informations spécifiques aux applications qui décrivent les structures de données d'un objet. Les métadonnées servent à construire des définitions d'objet métier utilisées lors de l'exécution.

Une fois le connecteur installé, mais avant que vous ne puissiez l'exécuter, vous devez créer les définitions d'objets métier. Les objets métier traités par le connecteur peuvent porter n'importe quel nom, pourvu qu'il soit autorisé par le courtier d'intégration. Pour plus d'informations sur les conventions de dénomination, voir *Naming Components Guide*.

Un connecteur contrôlé par des métadonnées traite chaque objet métier qu'il prend en charge en fonction des métadonnées codées dans la définition de l'objet métier. Ceci permet au connecteur de gérer les définitions d'objet métier, qu'elles soient nouvelles ou modifiées, sans qu'il ne soit nécessaire de modifier le code. Il est possible de créer de nouveaux objets via Object Discovery Agent (ODA) dans Business Object Designer. Pour modifier un objet, utilisez directement Business Object Designer (sans passer par l'ODA).

Les métadonnées spécifiques à l'application incluent l'objet métier et les définitions de ses propriétés d'attribut. Les valeurs des données réelles de chaque objet métier sont transmises dans les objets de message lors de l'exécution.

Le connecteur fait des suppositions sur la structure des objets métier pris en charge, les relations entre les objets métier parent et enfant et le format des données. Il est important que la structure de l'objet métier soit parfaitement identique à la structure définie pour le bean correspondant, sinon le connecteur ne peut pas traiter correctement les objets métier.

Si vous avez besoin de modifier la structure de l'objet métier, faites-le dans le bean d'entreprise correspondant, puis exportez les modifications dans le fichier JAR pour qu'elles entrent dans l'ODA.

Pour plus d'informations sur la modification des définitions d'objet métier, voir *WebSphere Business Integration Adapters Business Object Development Guide.*

Structure de l'objet métier du connecteur

Le connecteur traite les objets métier utilisés par les beans d'entreprise. Cette section aborde les principaux concepts relatifs à la structure des objets métier traités par le connecteur EJB.

Méthodes

Pour chaque méthode définie dans un fichier de classe Java, l'ODA crée un attribut dans l'objet métier (pour plus d'informations, voir «Attributs» et «ASI de l'attribut», à la page 31).

Le type d'attribut créé par l'ODA à partir d'une méthode est un objet métier enfant contenant d'autres attributs qui représentent des paramètres de méthode et des types de retour. Ces attributs apparaissent dans le même ordre que les paramètres de la méthode EJB. L'attribut `Return_Value` (non utilisé si la méthode définie est de type `void`) apparaît toujours en dernier dans l'ordre des arguments. Il représente le résultat de l'appel de la méthode EJB. Les attributs d'objet métier enfants peuvent être de type simple ou de type objet (complexe), en fonction du type du paramètre de méthode ou de la valeur en retour. Les informations spécifiques à l'application (ASI) de ces attributs contiennent le nom de méthode éloignée exposé. Pour plus d'information sur l'ASI des attributs, voir «ASI de l'attribut», à la page 31.

Le connecteur EJB exige que l'objet métier ait un objet métier enfant mappé sur une méthode creator, obtenue dans la liste des méthodes creator de l'interface locale, et une ou plusieurs méthodes d'objet métier extraites de la liste des méthodes d'interface éloignées.

Lorsqu'une méthode éloignée utilise des objets java en tant qu'argument ou type de retour, et lorsque les objets java contiennent plusieurs méthodes getter/setter, l'ODA peut être utilisé pour mapper les méthodes getter/setter sur les attributs, au lieu de créer un objet métier enfant pour chacune d'entre elles.

Notez que tous les caractères non alphanumériques d'une propriété ou d'un nom de méthode sont remplacés dans le nom d'objet métier correspondant par un caractère souligné (_).

Attributs

Pour chaque variable de membre public, attribut et paramètre de méthode d'une méthode de bean d'entreprise ayant été définie dans un fichier de classe Java, un attribut d'objet métier correspondant est généré par l'ODA. Le fichier JAR, qui contient les interfaces, est utilisé par l'ODA pour créer des définitions d'objet métier mappées sur des beans d'entreprise éloignés et locaux.

Si un attribut de la classe bean n'est pas un attribut simple mais un objet, l'attribut d'objet métier (BO) se mappe sur un objet métier dont la définition est identique à celle de la classe Java ou de l'interface EJB correspondante.

Les objets métier peuvent être plats ou hiérarchiques. Un objet métier plat contient uniquement des attributs simples, c'est-à-dire qui représentent une seule valeur (telle qu'une chaîne) et ne pointent pas sur des objets métier enfants. Un objet métier hiérarchique contient des attributs simples ainsi que des objets métier enfants ou des tableaux d'objets métier enfants contenant des valeurs d'attribut.

Pour un objet conteneur, la cardinalité 1 ou la relation de cardinalité simple, indique qu'un attribut contenu dans un objet métier parent ne contient qu'un seul objet métier enfant. Cet objet métier enfant est une collection qui ne peut contenir qu'un seul enregistrement. Le type d'attribut est l'objet métier enfant.

La cardinalité `n`, ou la relation de cardinalité multiple, indique qu'un attribut contenu dans un objet métier parent contient un tableau d'objets métier enfants. Dans ce cas, l'objet métier enfant est une collection qui peut contenir plusieurs enregistrements. Le type de l'attribut est le même que celui du tableau d'objets métier enfants.

Pour plus d'informations sur les attributs d'objet métier, voir «ASI de l'attribut», à la page 31.

Applications spécifiques à l'application

Les informations spécifiques à l'application fournissent au connecteur les instructions qui dépendent de l'application concernant le traitement des objets métier. Si vous développez ou modifiez une définition d'objet métier, vous devez veiller à ce que les informations spécifiques à l'application contenues dans la définition correspondent à la syntaxe attendue par le connecteur.

Les informations spécifiques à l'application sont représentées par une paire nom-valeur et peuvent être précisées pour la totalité de l'objet métier, pour chaque attribut d'objet métier et pour chaque instruction.

ASI de l'objet métier

L'ASI de l'objet apporte des informations fondamentales sur la nature d'un objet métier et sur l'objet qu'il contient. Pour afficher l'ASI d'un objet métier, ouvrez l'objet dans Business Object Designer et cliquez sur l'onglet General. L'ASI de l'objet métier s'affiche dans la zone **Business Object Level Application-specific information**. Pour plus d'informations sur cet écran, voir Chapitre 5, «Création et modification des objets métier», à la page 39.

Le tableau 4 décrit l'ASI d'objet métier de l'objet qui représente les beans d'entreprise.

Remarque : Les noms d'ASI ne sont pas reconnus pour les objets métier qui représentent des méthodes, des paramètres de méthode et des valeurs en retour de méthode. Pour plus d'informations sur les attributs d'objet métier créés pour des méthodes de beans d'entreprise, voir «Méthodes», à la page 28.

Tableau 4. ASI de l'objet métier

ASI de l'objet	Description
`object_type=RemoteEJB`	Indique que l'objet est un bean d'entreprise déployé sur un serveur d'applications (éloigné) plutôt qu'une classe Java standard non déployée sur un serveur d'applications (local).
`jndi_name=<`*`JNDIName`*`>`	Nom JNDI du bean d'entreprise
`home_class=<`*`className`*`>`	Nom de classe de l'interface locale
`proxy_class=<`*`className`*`>`	Nom de classe de l'interface éloignée

L'exemple suivant illustre l'ASI d'objet métier pour un objet métier que le connecteur traite avec des beans d'entreprise déployés sur WebSphere Application Server. Lorsque le connecteur reçoit l'objet métier émis par le courtier, il localise d'abord une instance de l'interface locale (com.ibm.websphere.AccountBookHome) sur le serveur d'applications, en recherchant `WsSamples/Account` dans le contexte JNDI configuré. Le connecteur utilise ensuite cette instance locale pour créer une nouvelle instance de l'interface éloignée (`com.ibm.websphere.AccountBook`), qu'il peut utiliser pour appeler des méthodes sur l'EJB.

```
BO ASI=object_type = RemoteEJBObject
           proxy_class = com.ibm.websphere.AccountBook
           home_class = com.ibm.websphere.AccountBookHome
           jndi_name = WsSamples/Account
```

Dans cet exemple de code :

- `BO ASI = object type` est requis pour montrer que l'objet est un objet éloigné.
- `proxy_class = com.ibm.websphere.AccountBook` représente le nom du mandataire éloigné.
- `home_class = com.ibm.websphere.AccountBookHome` est le nom de l'interface locale.
- `jndi_name = WsSamples/Account` est le nom JNDI.

Un objet Java standard (local) peut avoir un `BO ASI = auto_load_or_write`.

L'ASI d'objet métier d'un message pris en charge par un gestionnaire de données doit contenir la valeur `object_type=dataHandlerObject; mime_type=<text_value>`, où *<text_value>* est le type mime correct défini pour le gestionnaire de données (tel que précisé dans le méta-objet du gestionnaire de données) que l'adaptateur doit utiliser pour convertir les données.

ASI de l'instruction

Chaque objet métier contient une instruction. Elle indique quelles méthodes appeler sur le bean d'entreprise. Pour l'adaptateur pour EJB, la première méthode doit être une méthode creator depuis l'interface locale du bean correspondant, et les méthodes éloignées doivent être les méthodes de processus métier de l'interface éloignée du bean.

L'ASI de l'instruction contient une séquence de noms d'attributs, chacun contenant une méthode pour le gestionnaire d'objet métier à appeler. En général, la méthode à appeler appartient à l'objet métier lui-même (et non à un parent de l'objet métier), auquel cas vous devez indiquer la méthode dans l'ASI de l'instruction de l'objet.

Si la méthode à appeler appartient à un parent dans la hiérarchie de l'objet métier, alors ce parent peut être référencé à partir de l'enfant en utilisant la balise `PARENT` comme préfixe dans le nom de la méthode.

Par exemple, la figure 3, à la page 31 illustre une hiérarchie d'objet métier dans laquelle ContactDetails est un objet enfant de Contact, lui-même enfant de PSRCustomerAccount.

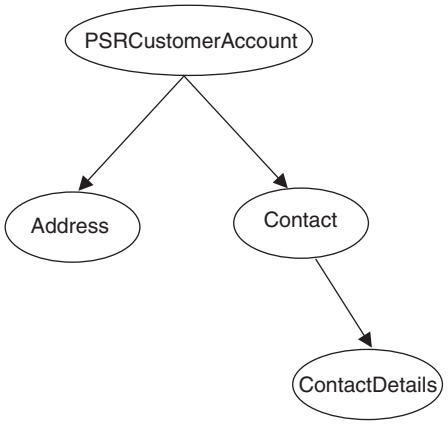


Figure 3. Hiérarchie d'objet métier et ASI de l'instruction

Si une méthode appartenant à PSRCustomerAccount est appelée dans l'objet métier ContactDetails, alors l'ASI de l'instruction de ContactDetails représente la hiérarchie d'objet métier de la façon suivante :

`PARENT.PARENT.<methodName>`

Par contre, si la méthode appartient à l'objet métier Contact, alors l'ASI de l'instruction de ContactDetails doit être définie comme suit :

`PARENT.<methodName>`

Notez qu'il n'est possible d'appeler que les méthodes qui appartiennent aux objets parents de la hiérarchie. Un objet métier parent ne peut appeler de méthode enfant.

Le développeur du connecteur détermine les opérations EJB affectées à l'instruction. Les instructions prises en charge sont les suivantes :

- Create
- Delete
- Retrieve
- Update

Pour un objet donné, vous pouvez préciser les quatre instructions prises en charge (Create, Retrieve, Delete et Update) et affecter `n` méthodes en tant qu'action de chaque instruction, `n` étant le nombre de méthodes du bean d'entreprise correspondant.

Pour afficher l'ASI de l'instruction d'un objet métier, ouvrez un objet métier dans Business Object Designer et cliquez sur l'onglet General. L'ASI de l'instruction figure dans la table **Instructions prises en charge**, qui indique le nom des instructions supportées ainsi que l'ASI correspondant. Pour plus d'informations sur Business Object Designer et les onglets et écrans de l'assistant ODA, voir Chapitre 5, «Création et modification des objets métier», à la page 39.

ASI de l'attribut

Chaque objet métier dispose d'un ensemble d'attributs, décrits dans «Attributs», à la page 28 et mappés sur les méthodes de bean d'entreprise et propriétés correspondantes. Un objet métier plat contient des attributs simples, c'est-à-dire qui représentent une seule valeur (telle qu'une chaîne) et ne pointent pas vers des

objets métier enfants. Un objet métier hiérarchique contient des attributs simples ainsi que des objets métier enfants ou des tableaux d'objets métier enfants contenant les valeurs d'attribut.

L'ASI d'un attribut d'objet métier peut concerner des attributs simples et des attributs complexes contenant des objets enfants. Pour un attribut complexe, l'ASI varie selon que l'enfant contenu est une propriété ou une méthode d'un objet. Le mappage des constructions EJB sur les objets métier générés par ODA est décrit au tableau 8, à la page 34.

Pour afficher l'ASI de l'attribut d'un objet métier, ouvrez un objet métier dans Business Object Designer et cliquez sur l'onglet Attributes. Chaque attribut de l'objet métier apparaît dans la colonne **Name**. Les propriétés d'attribut, décrites au tableau 5, au tableau 6, à la page 33 et au tableau 7, à la page 33, sont indiquées dans les autres colonnes. Parmi ces propriétés, l'ASI figure dans la colonne **Application Specific Information**. Pour plus d'informations sur cet écran, voir Chapitre 5, «Création et modification des objets métier», à la page 39.

Le tableau 5 décrit les propriétés d'objet métier des attributs simples. Ces propriétés incluent l'ASI (propriété `AppSpecificInfo`) de l'attribut. Un attribut simple est toujours différent d'un enfant, c'est-à-dire une valeur booléenne, une chaîne ou un entier.

Tableau 5. Propriétés d'attribut de l'objet métier : attributs simples

Propriété d'attribut	Description
`Name`	Indique le nom de l'attribut de l'objet métier.
`Type`	Indique le type de l'attribut de l'objet métier. Voir tableau 8, à la page 34 pour obtenir des informations sur le mappage des constructions EJB sur les types d'attribut d'objet métier.
`MaxLength`	Non utilisé.
`IsKey`	Chaque objet métier doit avoir au moins un attribut clé, que vous devez spécifier en définissant la propriété key sur `true` pour un attribut. Notez que cet attribut est utilisé par Business Object Designer, plutôt que par le connecteur.
`IsForeignKey`	Indique que le connecteur doit vérifier si l'objet doit être conservé dans le pool d'objets par appel.
`IsRequired`	Non utilisé.

`AppSpecInfo`	Contient le type Java d'origine. La propriété a le format suivant :
	
	`property=<`*`propertyName`*`>, type=<`*`typeName`*`>`
	
	`property` est le nom de l'attribut EJB ou la variable du membre objet. Utilisez cette paire nom-valeur pour capturer le nom de la variable du membre objet EJB d'origine.
	
	`type` est le type Java d'un attribut simple (variable du membre objet EJB). Par exemple, `type=java.lang.String`
	Voir tableau 8, à la page 34 pour plus d'informations sur le mappage des constructions EJB sur les types d'attributs d'objet métier.

`DefaultValue`	Non utilisé.

Le tableau 6, à la page 33 décrit les éléments des attributs complexes contenant des objets enfants qui ne sont pas des méthodes. Trois éléments incluent l'ASI de l'attribut.

Tableau 6. Propriétés d'attribut de l'objet métier : attributs d'objets enfants non-méthode

Propriété d'attribut	Description
Name	Indique le nom de l'attribut de l'objet métier.
type	Type de l'objet contenu. Défini sur proxy si le type est un objet métier.
ContainedObjectVersion	Non utilisé.
Relationship	Indique que l'enfant est un attribut de conteneur. Défini sur Containment.
IsKey	Non utilisé
IsForeignKey	Non utilisé
Is Required	Non utilisé
AppSpecificInfo	Contient le nom de la zone de l'application EJB d'origine. Cette propriété a le format suivant : property=*propertyName*, use_attribute_value=<*(optional)BOName.AttributeName*>, type=<typeName> property est le nom de la variable du membre objet EJB. Utilisez cette paire nom-valeur pour capturer l'attribut du membre objet EJB d'origine. Dans le cas d'un attribut indiquant un argument à une méthode, ne définissez pas de valeur pour property, puisque l'argument n'a pas de nom et est simplement un argument de n'importe quel type standard. use_attribute_value est le nom d'objet métier, au format *BOName.AttributeName*. La définition de cet ASI a pour effet que le connecteur accède à l'attribut depuis le pool d'objets par appel. Notez que cette valeur n'est pas définie dans l'ODA lorsque vous créez l'objet métier, mais dans Business Object Designer. type est le type Java d'une propriété. Voir tableau 8, à la page 34 concernant le mappage de constructions EJB sur des objets métier.)
Cardinality	Défini sur *n* si le type représente un tableau ou un vecteur. Sinon, défini sur 1.

Le tableau 7 décrit les propriétés des attributs complexes contenant des objets enfants de type méthode. Ces propriétés incluent l'ASI de l'attribut.

Tableau 7. Propriétés d'attribut de l'objet métier : attributs d'objets enfants méthode

Propriété d'attribut	Description
Name	Nom de l'attribut de l'objet métier.
type	Objet métier.
Relationship	Défini sur Containmen , indiquant qu'il s'agit d'un objet enfant.
IsKey	Défini sur true si le nom de l'attribut est UniqueName, sinon défini sur false.
IsForeignKey	Défini sur false.
Is Required	Défini sur false.
AppSpecificInfo	Contient le nom de l'interface de méthode éloignée exposé, qui est le nom de l'appel de méthode lancé sur le serveur EJB par le bean d'entreprise. Cet attribut est au format : method_name=<*remoteClassName.RemoteMethodName*> Si la méthode est un constructeur, method_name=CONSTRUCTOR
Cardinality	Défini sur 1.

Notez que les méthodes ont des arguments et des valeurs en retour. Les arguments et les valeurs en retour peuvent être complexes (avec des objets enfants) ou simples.

Mappage des attributs : Enterprise JavaBeans (EJB) et objet métier

Cette section présente une liste des constructions EJB définies dans un fichier JAR et leurs attributs d'objet métier correspondants. Pour tous les attributs d'objet métier qui ne sont pas des objets métier enfants, le type de données est `String`. Dans un objet métier, l'ASI contient le type de données réel de l'attribut et est utilisé lors de l'appel de méthodes en rapport à l'interface éloignée du bean d'entreprise.

Pour plus d'informations sur l'ASI de l'objet métier, voir «Applications spécifiques à l'application», à la page 29.

Tableau 8. Mappage d'objet : bean d'entreprise sur objet métier

Construction EJB	Objet métier	Attribute ASI type=
Toutes les classes référencées dans le fichier JAR	Object	`proxy_class=<remote interface name>`
boolean	Boolean	`type=boolean/Boolean`
char/Character	String	`type=char/Character`
byte/Byte	String	`type=byte/Byte`
java.lang.String	String	`type=string`
short/Short	Integer	`type=short/Short`
int/Integer	Integer	`type=int/Integer`
long/Long	Integer	`type=long/Long`
float/Float	Float	`type=float/Float`
double/Double	Double	`type=double/Double`
java.math.BigDecimal	String	`type=java.math.BigDecimal`
java.math.BigInteger	String	`type=java.math.BigInteger`
class	Object	`proxy_class=<fully qualified class name>`
array	Object Objet métier enfant avec cardinalité multiple	`type=ArrayOf_<datatype>` Par exemple, `type=ArrayOf_int`
method	Object Child BO	`method_name=<methodName>`
method (pas d'argument et type de retour void)	String	`method_name=<methodName>`

Remarque : Si vous ne prévoyez pas de déréférencer l'attribut, utilisez l'ASI `type=PlaceholderOnly`. Ceci indique au connecteur de ne pas renseigner cet attribut. L'attribut peut toujours être utilisé en tant que flux multi-appel s'il est marqué en tant que clé étrangère (`IsForeignKey` est défini sur `true`), ou en tant que l'ASI `use_attribute_value` pointant sur un attribut compatible.

Types de tableaux

Notez les informations suivantes sur les types de `tableaux` :

- Pour utiliser un type `array`, précisez un ASI `type=ArrayOf_<value>`, où *value* est l'une des valeurs ASI d'attribut indiquées au tableau 8, à la page 34. Par exemple, `type=ArrayOf_int` indique un tableau de variables `int`. Elles sont mappées sur un objet métier de cardinalité n contenant l'élément.
- Un tableau d'objets (`Object[]`) en Java a un type ASI correspondant de `ArrayOf_proxy`. Les objets proxy sont traités en fonction de chaque élément du tableau. Si le tableau proxy est un argument vers une fonction, le traitement des instructions s'effectue sur chaque objet du tableau *avant* d'exécuter la méthode. Si le tableau est une valeur de retour, le traitement des instructions s'effectue sur chaque objet du tableau *après* l'exécution de la méthode.
- Un tableau dimensionné peut être utilisé en entrée mais pas en sortie.
- Un SafeArray est pris en charge en tant qu'entrée et en tant que valeur de retour.

Exécution de l'adaptateur pour appeler des exemples d'objets métier

Les rubriques suivantes expliquent comment exécuter les fichiers exemples fournis avec le produit, et illustrent comment le connecteur appelle les services d'un fichier JAR pour créer des objets métier.

- «Exécution des fichiers exemples»
- «Code du fichier JAR EJB», à la page 36
- «Exemple d'objet métier», à la page 37

Exécution des fichiers exemples

Les fichiers exemples figurent dans le répertoire *ProductDir*`\connectors\EJB\samples`, où *ProductDir* est le répertoire d'installation du connecteur.

WebSphere Application Server Thin Client doit impérativement être installé pour que l'adaptateur puisse accéder de façon éloignée aux beans déployés sur le serveur.

Remarque : Le client Application Server Thin ne peut être installé dans un répertoire dont le nom contient des espaces.

Les fichiers exemples illustrent un bean de session simple, `MusicCart`, qui fournit une interface locale, une interface éloignée, un bean et deux classes auxiliaires, `RecordingHelper` et `CustomerHelper`. Les fichiers exemples incluent également deux fichiers de configuration du connecteur : `BIA_EJBConnector.cfg`, qui contient les paramètres des propriétés du connecteur nécessaires à la bonne exécution de l'exemple, et `BIA_PortConnector.cfg`, qui contient les propriétés du connecteur de test.

L'exemple de bean `MusicCart` contient des méthodes dont le paramètre d'entrée est un objet Java et dont la sortie est un objet. Il contient également des méthodes dont l'entrée est un tableau d'objets, pour montrer comment l'adaptateur gère les tableaux. L'exemple montre également comment l'adaptateur traite le gestionnaire de données.

Les fichiers exemples contiennent un fichier d'objet métier (`BIA_SampleMusicCartBO.bo`) et des fichiers source EJB (dans

ProductDir\connectors\EJB\samples\SampleMusicCartEJB\src), qui illustrent le mappage des classes EJB locales et éloignées sur les objets métier qui utilisent l'ODA pour EJB. En envoyant BIA_SampleMusicCartBO.bo depuis le connecteur de test vers le serveur d'applications, vous pouvez voir comment les méthodes addRecording, getFirstRecordInfo, modifyMusicRequestUsingDataHandler et getAllRecordInfo sont exécutées, et de quelle façon le connecteur gère le traitement des requêtes.

Les étapes suivantes supposent que vous exécutez le connecteur pour échanger des objets métier avec les beans d'entreprise déployés sur WebSphere Application Server 5.0.

1. Installez l'adaptateur EJB comme indiqué au Chapitre 2, «Installation de l'adaptateur», à la page 11.
2. Déployez le fichier JAR EJB *ProductDir*\connectors\EJB\samples\SampleMusicCartEJB\BIA_MusicBeanSample.jar sur votre instance de WebSphere Application Server 5.0.
3. Chargez les deux fichiers de configuration du connecteur dans le répertoire *ProductDir*\connectors\EJB\samples\ (BIA_EJBConnector.cfg et BIA_PortConnector.cfg), dans le référentiel de votre courtier d'intégration, par exemple InterChange Server.
4. Chargez les exemples d'objets métier depuis le répertoire *ProductDir*\connectors\EJB\samples\SampleBOs dans le référentiel du courtier.
5. Modifiez le fichier de démarrage du connecteur pour pointer sur l'emplacement du fichier JAR (BIA_MusicBeanSample.jar) déployé à l'étape 2. Pour plus d'informations sur la modification du fichier de démarrage, voir «Configuration du fichier de démarrage», à la page 24.
6. Dans le fichier de démarrage, supprimez la mise en commentaire des commandes des paramétrages de votre serveur d'applications. Ici, vous devez supprimer la mise en commentaire des commandes pour WebSphere Application Server, puisque le présent exemple suppose que vous exécutez le connecteur pour échanger des objets métier avec les beans d'entreprise déployés sur WebSphere Application Server 5.0.
7. Démarrez l'instance de WebSphere Application Server.
8. Commencez à exécuter le connecteur, comme indiqué dans «Démarrage du connecteur», à la page 24.
9. Envoyez le fichier BIA_SampleMusicCartBO.bo depuis le connecteur de test vers le serveur d'applications.

Pendant l'exécution du connecteur, vous pouvez observer comment les méthodes de bean d'entreprise définies dans les fichiers exemples sont exécutées à partir de l'adaptateur, pour appeler des services fournis par le WebSphere Application Server éloigné.

Code du fichier JAR EJB

L'exemple de code suivant est un extrait du fichier JAR EJB qui définit les méthodes de la classe EJB nommée MusicCartBean. Notez la méthode définie à la fin de l'exemple de code : getCustomer.

Remarque : Toutes les méthodes de cette classe ne sont pas définies dans l'exemple de code fourni ici. L'exemple est une partie d'un fichier plus volumineux décrit dans «Exécution des fichiers exemples», à la page 35.

L'objet métier qui correspond à ce code est présenté dans la figure 4, à la page 38.

```
public class MusicCartBean implements SessionBean {
  CustomerHelper customerHelper;
  ArrayList shoppingList;
  RecordingHelper[] recordHelperArr;


   // EJB Methods
   public void ejbRemove() {}
   public void ejbActivate() {}
   public void ejbPassivate() {}
   public void setSessionContext (SessionContext ctx) {}

  public void ejbCreate(String person,String password, String email)
      throws CreateException {
           if (person == null || person.equals("")) {
             throw new CreateException(
               "Name cannot be null or empty.");
           }
           else {
             customerHelper = new
               CustomerHelper(person, password, email);
             customerHelper.setName(person);
             customerHelper.setEmail(email);
             customerHelper.setPassword(password);
           }
           shoppingList = new ArrayList();
  }

   public void ejbCreate(CustomerHelper customerHelper)
      throws CreateException {
          customerHelper.setName(customerHelper.name);
          customerHelper.setEmail(customerHelper.email);
          customerHelper.setPassword(customerHelper.password);
          shoppingList = new ArrayList();
  }
   // Business methods implementation

  public CustomerHelper getCustomer() {
      return customerHelper;
  }
```

Exemple d'objet métier

L'exemple d'écran suivant illustre la structure d'objet métier correspondant au code source présenté dans «Code du fichier JAR EJB», à la page 36. Cet objet métier est créé par l'ODA qui reconnaît les objets et les constructions définis dans le fichier JAR EJB d'origine, et génère les objets métier correspondants. Pour savoir comment utiliser l'ODA pour générer cet exemple, voir Chapitre 5, «Création et modification des objets métier», à la page 39.

Notez le quatrième attribut, `getCustomer`. Il contient un objet de méthode complexe, correspondant à la méthode `getCustomer` définie à la fin du code exemple de la classe `MusicCartBean`, présenté dans «Code du fichier JAR EJB», à la page 36.

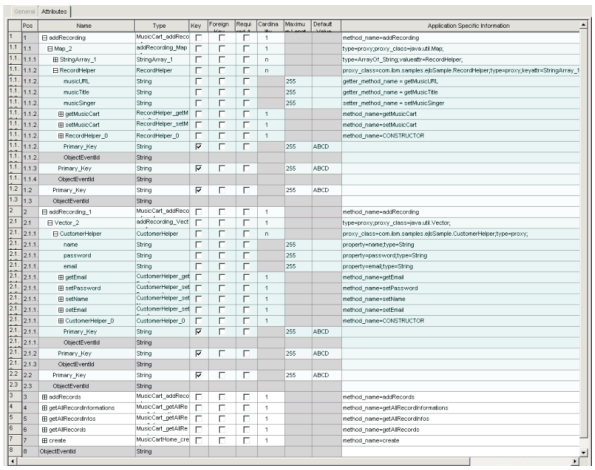


Figure 4. ASI d'objet métier et instructions prises en charge

Génération d'objets métier

A chaque fois qu'un événement se produit pendant l'exécution, une application EJB envoie un objet de message contenant des données d'objet et des informations sur le type de transaction. Le connecteur mappe ces données sur la définition d'objet métier correspondante, pour créer un objet métier spécifique à l'application. Le connecteur envoie ces objets métier au courtier d'intégration pour traitement. Il reçoit également des objets métier depuis le courtier d'intégration, et les transmet à l'application EJB.

Remarque : Si le modèle d'objet de l'application EJB est modifié, utilisez l'ODA pour créer une nouvelle définition. Si les définitions d'objet métier qui figurent dans le référentiel du courtier d'intégration ne sont pas exactement identiques à celles que l'application EJB envoie, le connecteur ne parvient pas à créer d'objet métier et la transaction échoue.

Business Object Designer offre une interface graphique permettant de créer et de modifier les définitions d'objet métier utilisées pendant l'exécution. Pour plus d'informations, voir Chapitre 5, «Création et modification des objets métier», à la page 39.

Chapitre 5. Création et modification des objets métier

Le présent chapitre explique comment créer des objets métier à l'aide de l'ODA (Object Discovery Agent) pour EJB. Il contient les sections suivantes :

- «Présentation de l'ODA pour EJB»
- «Génération de définitions d'objet métier»
- «Téléchargement des fichiers d'objet métier», à la page 53
- «Traitement de l'objet de collection», à la page 46

Présentation de l'ODA pour EJB

Un ODA permet de générer des définitions d'objet métier. Une définition d'objet métier est un modèle d'objet métier. L'ODA examine les objets d'application spécifiés, "reconnaît" les éléments de ces objets qui correspondent aux attributs d'objet métier, et génère des définitions d'objet métier pour représenter les informations. Business Object Designer fournit une interface graphique pour accéder à l'Object Discovery Agent et l'utiliser de façon interactive.

L'ODA pour EJB génère des définitions d'objets métier à partir des métadonnées contenues dans les fichiers JAR EJB. L'assistant Business Object Designer automatise la création de ces définitions. Vous utiliserez l'ODA pour créer des objets métier, et Connector Configurator pour configurer le connecteur nécessaire pour les prendre en charge. Pour plus d'informations sur Connector Configurator, voir l'Annexe B, «Connector Configurator», à la page 85.

Génération de définitions d'objet métier

Cette section explique comment utiliser l'ODA EJB dans Business Object Designer pour créer des définitions d'objet métier. Pour plus d'informations sur le lancement et l'utilisation de Business Object Designer, voir *IBM WebSphere Business Integration Adapters Business Object Development Guide*.

Démarrage de l'ODA

Vous pouvez exécuter l'ODA sur n'importe quelle machine capable de monter le système de fichiers sur lequel réside le référentiel de métadonnées (fichiers JAR). Pour cela, utilisez le fichier de démarrage de l'ODA. Ce fichier contient les paramètres de démarrage, y compris les chemins vers certains fichier EJB requis et fichiers JAR du connecteur. Ces fichiers JAR doivent également être accessibles depuis la machine sur laquelle vous exécutez l'ODA.

Le nom par défaut de l'ODA pour EJB est EJBODA. Vous pouvez le modifier en changeant la valeur de la variable `AGENTNAME` dans le script de démarrage.

Pour démarrer l'ODA, exécutez la commande suivante :

- `start_EJBODA.bat` (Windows 2000)
- `start_EJBODA.sh` (Unix)

La variable `CLIENT JARPATH` du fichier de démarrage ne dispose d'aucune valeur par défaut. Par conséquent, avant d'exécuter la commande `start`, définissez la valeur du répertoire et du nom de fichier du chemin du JAR client, afin que l'ODA puisse inclure le fichier dans son chemin de classe Java. Par exemple :

```
set CLIENTJARPATH=C:\BIA_MusicBeanSample.jar
```

Exécution de Business Object Designer

Business Object Designer dispose d'un assistant pour vous guider à travers les étapes nécessaires pour générer une définition d'objet métier en utilisant l'ODA. La procédure est la suivante :

Sélection de l'agent

Pour sélectionner l'agent, procédez comme suit :

1. Démarrez Business Object Designer.
2. Cliquez sur **File > New Using ODA**. L'écran *Business Object Wizard - Step 1 of 6 - Select Agent* s'affiche.
3. Sélectionnez ODA/AGENTNAME dans la liste **Located agents** et cliquez sur **Next**. (Vous devrez peut-être cliquer sur **Find Agents** si l'agent souhaité n'est pas répertorié.)

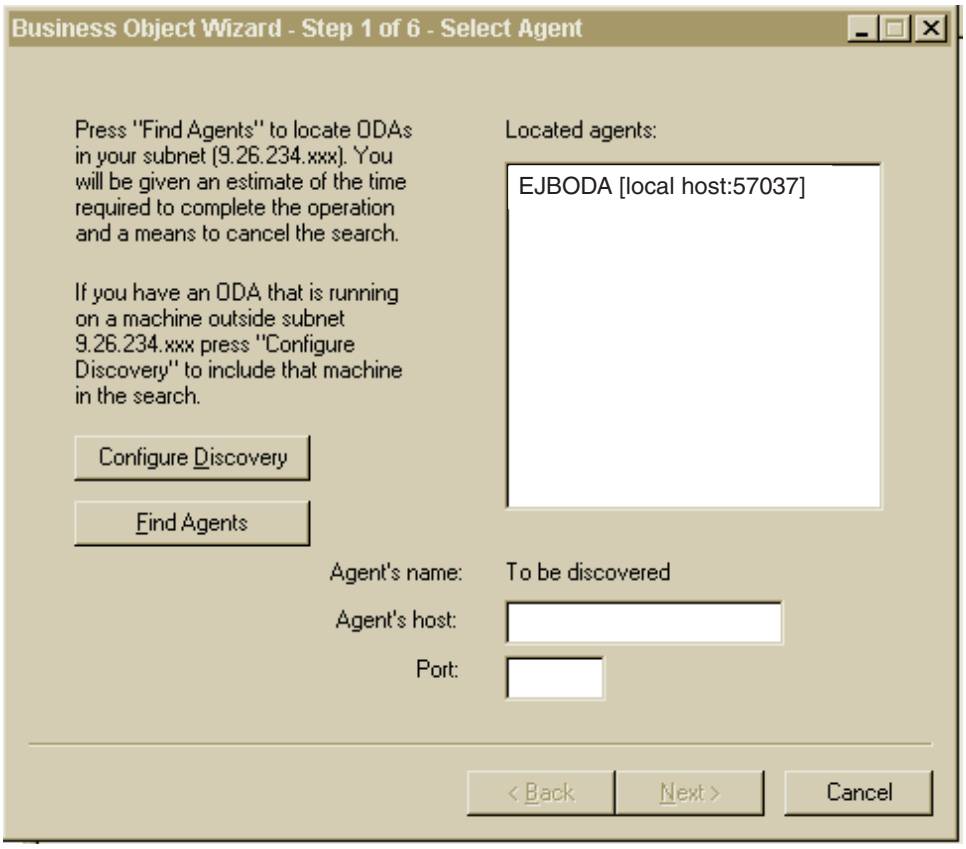


Figure 5. Ecran Select Agent

Configuration de l'agent

Une fois que vous avez cliqué sur **Next** dans l'écran Select Agent, l'écran *Business Object Wizard - Step 2 of 6 - Configure Agent* s'affiche. La figure 6 représente cet écran complété d'exemples de valeurs.

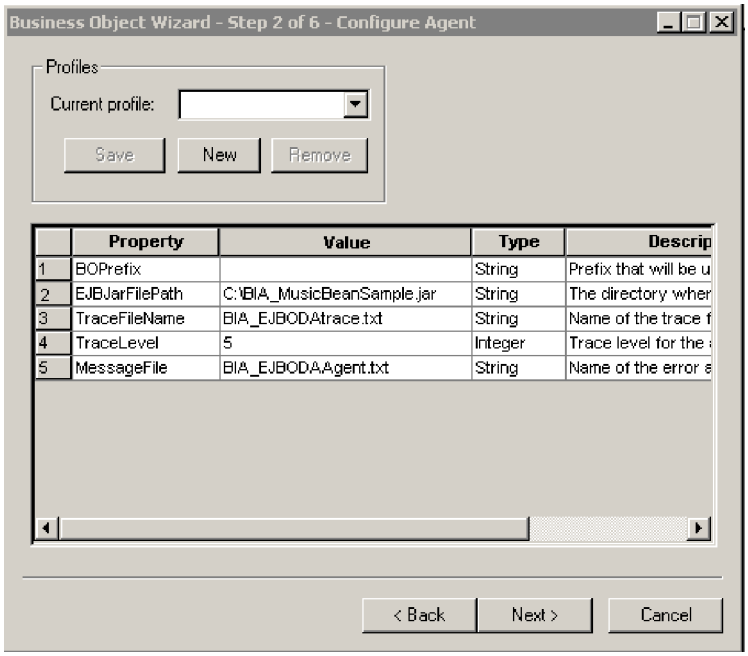


Figure 6. Ecran Configure Agent

Les propriétés définies dans cet écran sont décrites dans le tableau 9. Vous pouvez enregistrer dans un profil toutes les valeurs entrées dans cet écran. Au lieu de saisir de nouveau les données de la propriété la prochaine fois que vous exécuterez l'ODA, vous sélectionnerez un profil dans le menu déroulant Current profile et ré-utiliserez les valeurs enregistrées. Vous pouvez enregistrer plusieurs profils, chacun avec un ensemble de valeurs différent.

Tableau 9. Propriétés de configuration de l'agent

Nom de propriété	Valeur par défaut	Type	Description
`EJBJarFilePath`	Aucune	String	(obligatoire) Chemin complet du fichier JAR EJB. Il s'agit du fichier lu par l'ODA en tant qu'entrée pour générer des objets métier.
`DefaultBOPrefix`	Aucune	String	Préfixe que l'ODA ajoutera aux noms des objets métier qu'il génère
`TraceFileName`	`<agentname>trace.txt`	String	Nom du fichier de messages de trace, par exemple, `BIA_EJBODAtrace.txt`. Notez que le connecteur prend en charge des valeurs internationalisées pour cette propriété. Ainsi, le chemin de fichier et le nom peuvent être internationalisés, contrairement au contenu du fichier.
`TraceLevel`	0	Integer	(obligatoire) Niveau de traçage (de 0 à 5) de l'Agent. Pour plus d'informations sur les niveaux de traçage, voir «Traçage», à la page 56.

Tableau 9. Propriétés de configuration de l'agent (suite)

Nom de propriété	Valeur par défaut	Type	Description
`MessageFile`	`<agentname>Agent.txt`	String	Nom du fichier de messages contenant tous les messages affichés par l'ODA. Pour le connecteur pour EJB, le nom de ce fichier est `BIA_EJBODAAgent.txt`. Si vous n'indiquez pas correctement le nom du fichier de messages, l'ODA s'exécutera sans messages. Notez que le connecteur prend en charge des valeurs internationalisées pour cette propriété. Ainsi, le chemin de fichier et le nom peuvent être internationalisés, contrairement au contenu du fichier.

1. Saisissez la valeur de la propriété dans la colonne **Value** de l'écran Configure Agent, représenté dans la figure 6, à la page 41.
2. Pour créer un nouveau profil, utilisez les boutons **New** et **Save** dans la zone Profiles. Lorsque vous utiliserez de nouveau l'ODA, vous pouvez sélectionner un profil existant.

   Remarque : Si vous utilisez un profil existant, les valeurs de la propriété sont remplies automatiquement, mais vous pouvez les modifier si nécessaire.

Sélection d'un objet EJB

L'écran *Business Object Wizard - Step 3 of 6 - Select Source* s'affiche, tel qu'illustré dans la figure 7, à la page 43. L'écran dresse la liste des objets qui ont été définis dans le fichier JAR EJB. Utilisez cet écran pour sélectionner le nombre voulu d'objets EJB pour lesquels l'ODA générera des définitions d'objet métier.

Notez que la liste contient huit méthodes au total. Les deux premiers objets de la liste sont les méthodes creator de l'interface locale. Les six autres sont les méthodes de l'interface éloignée.

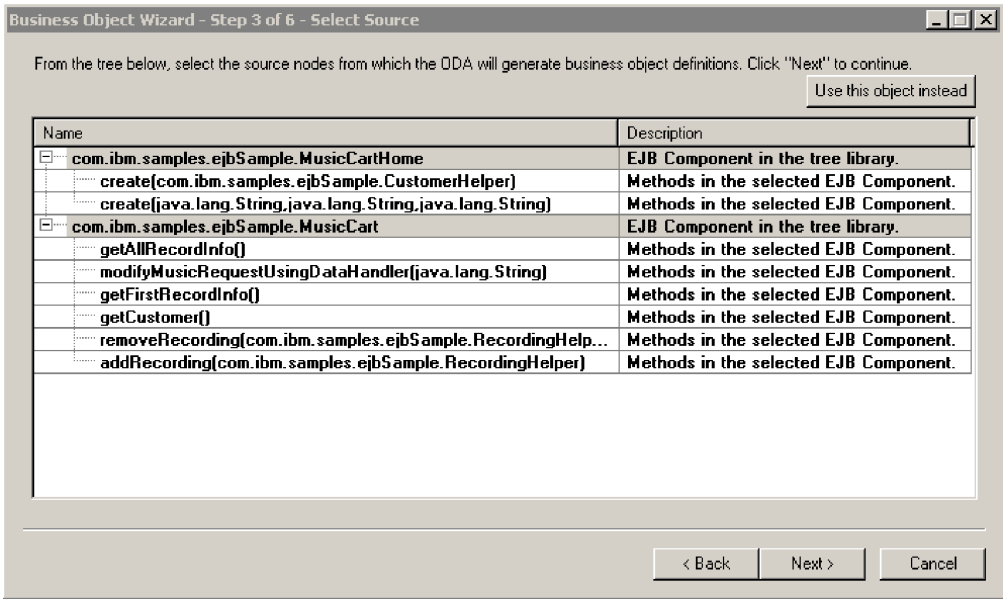


Figure 7. Ecran Select Source

1. Si nécessaire, développez l'objet EJB pour afficher la liste de ses méthodes.
2. Sélectionnez les objets EJB que vous souhaitez utiliser. Par exemple, vous pouvez faire les choix suivants :
   - Un objet EJB *et* quelques unes de ses méthodes.
   - Quelques méthodes d'un objet EJB, mais pas l'objet lui-même.
   - Les méthodes de différents objets EJB.
   - Uniquement un objet d'interface locale. Lorsque seul un objet de l'interface locale est sélectionné, l'ODA génère des objets métier pour toutes les méthodes de l'interface locale.
   - Uniquement un objet de classe de l'interface éloignée. Lorsque seul un objet de l'interface éloignée est sélectionné, l'ODA génère des objets métier pour toutes les méthodes de l'interface éloignée.

   Dans la figure 7, les classes `MusicCartHome` et `MusicCart` sont sélectionnées. Bien que seuls les noms de classe soient sélectionnés, l'ODA crée également les objets des méthodes associées à chacune des classes sélectionnées.
3. Cliquez sur **Next**.

Confirmation de la sélection d'objet

L'écran *Business Object Wizard - Step 4 of 6 - Confirm source nodes for business object definitions* s'ouvre. Il affiche les objets sélectionnés.

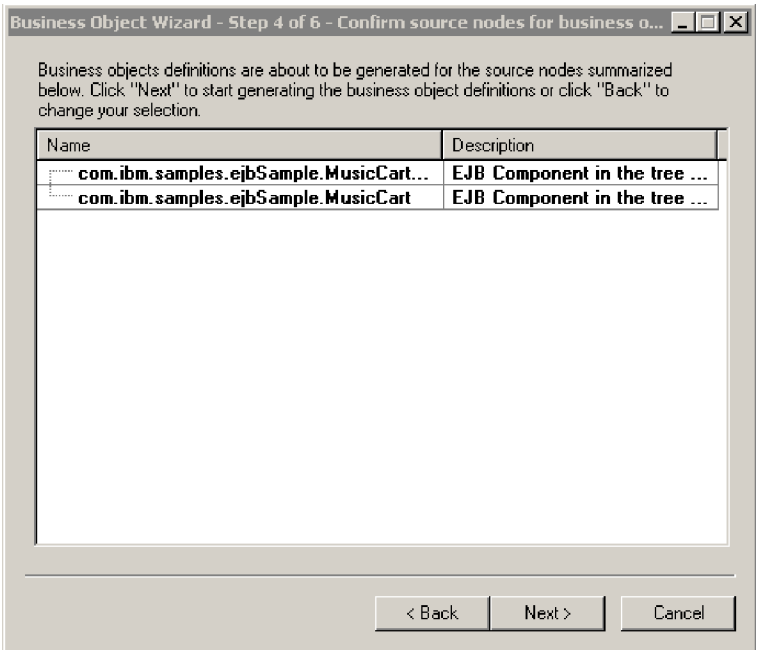


Figure 8. Ecran Confirm source node

Cliquez sur **Back** pour apporter des modifications ou sur **Next** pour confirmer que la liste est correcte.

L'écran *Business Object Wizard - Step 5 of 6 - Generating business objects...* s'ouvre et affiche un message indiquant que l'assistant est en train de générer les objets métier.

Sélection des instructions et affectation du nom JNDI

Une fois que vous avez cliqué sur **Next**, l'écran *Enter JNDI name for this Enterprise Java Bean* s'ouvre. Il s'affiche pour chaque objet source EJB sélectionné. Utilisez-le pour sélectionner les instructions prises en charge pour cet objet et pour affecter un nom JNDI.

Dans l'exemple présenté à la figure 9, à la page 45, le nom JNDI affecté est `MusicCartJNDI`.

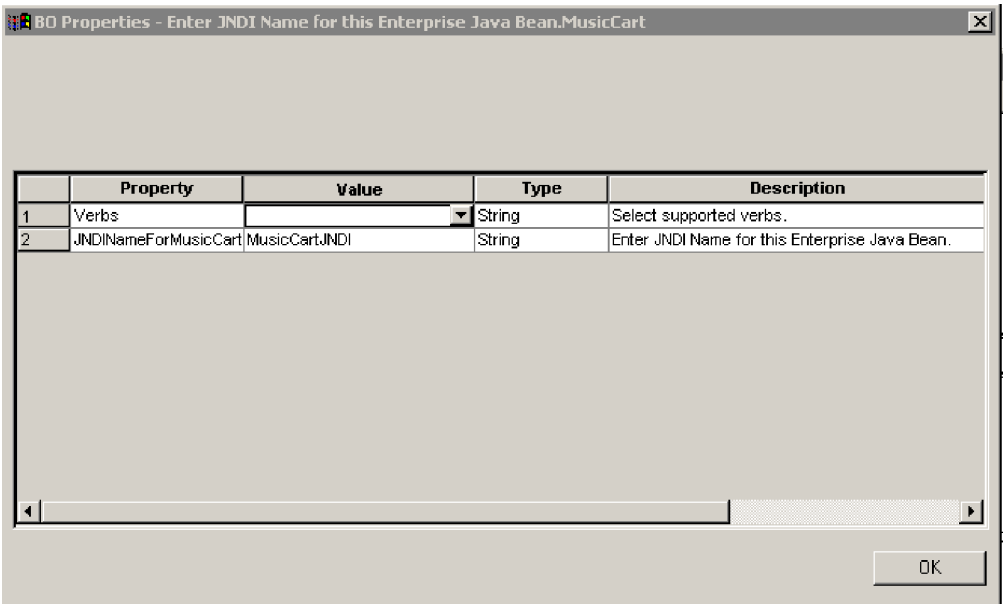
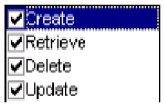


Figure 9. Ecran JNDI Name for Enterprise Java Bean

Dans cet écran, précisez également les instructions prises en charge par l'objet métier. L'ODA permet de préciser les quatre instructions prises en charge (Create, Retrieve, Delete, and Update) et d'affecter `n` méthodes en tant qu'action de chaque instruction, `n` représentant le nombre de méthodes de l'objet EJB correspondant. Pour préciser plus de quatre instructions ou pour modifier les informations sur l'instruction une fois que vous avez créé un objet métier, utilisez Business Object Designer.

Pour plus d'informations sur les instructions d'objet métier pour le connecteur EJB, voir «ASI de l'instruction», à la page 30.

1. Dans la liste **Value** de la propriété `Verbs`, sélectionnez les instructions qui seront prises en charge par l'objet métier. Vous pouvez sélectionner une ou plusieurs instructions. Vous pouvez également désélectionner une instruction à tout moment.



2. Dans la propriété `JNDI name for <className>`, entrez le nom JNDI du bean d'entreprise. Ce nom est défini lorsque le bean est déployé sur le serveur d'applications.
3. Cliquez sur **OK**.

Spécifiez l'ASI de l'instruction

Pour chaque instruction sélectionnée, une fenêtre s'affiche et indique la séquence de méthodes qui doit être exécutée pour l'instruction.

La figure 10, à la page 46 représente cet écran pour l'instruction `Create` de l'objet métier `MusicCart` créé en figure 7, à la page 43 et figure 8, à la page 44.

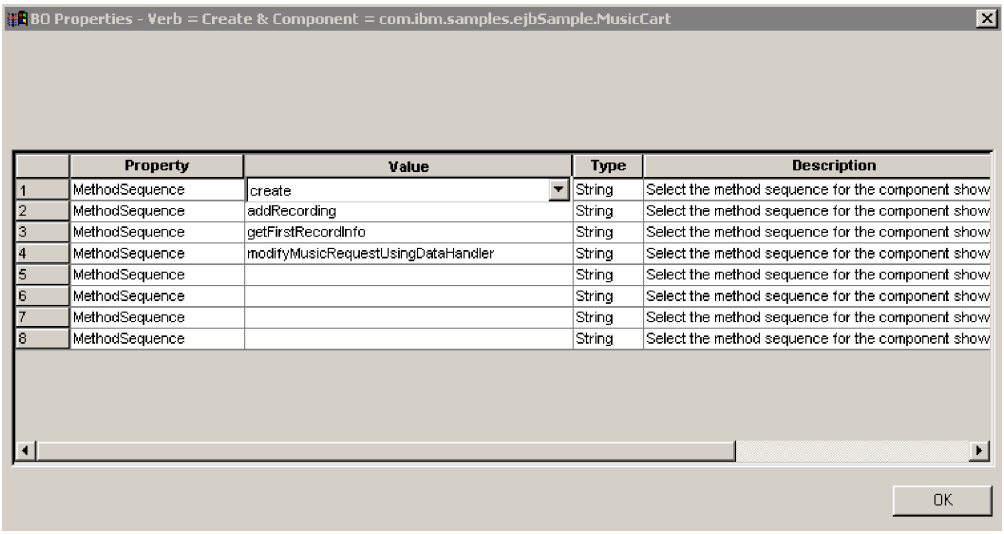


Figure 10. Définition de la séquence de méthodes d'instruction

1. Dans la liste **Value** de la propriété `MethodSequence`, cliquez sur la méthode que l'objet métier exécutera en premier pour l'instruction. Dans la figure 10, la séquence de méthodes est la suivante :
   - La première méthode requise dans la séquence de méthodes de toute instruction d'objet métier EJB est de type creator ou finder. Elle est définie dans l'interface locale du bean d'entreprise. Dans la figure 10, la première méthode est `create`. Le connecteur l'utilise pour obtenir une référence à l'interface éloignée et créer une instance du bean d'entreprise.

     Notez que si la première méthode sélectionnée sur l'ASI de l'instruction n'est pas de type creator ou finder, le connecteur affiche le message d'erreur suivant : `The first method obtained from the list of methods in the ver ASI is not a home creator method. Select a creator method from the home interface and try again.`
   - La seconde méthode de la séquence est `addRecording`. Cette méthode appartient à l'interface éloignée du bean d'entreprise.
   - La troisième méthode de la séquence est `getFirstRecordInfo`. Cette méthode appartient à l'interface éloignée du bean d'entreprise.
   - La dernière méthode de la séquence est `modifyMusicRequestUsingDataHandler`. Cette méthode appartient à l'interface éloignée du bean d'entreprise.

   En précisant une séquence de méthodes pour l'instruction, vous créez l'ASI d'instruction associée à cette instruction. Si nécessaire, vous pourrez modifier ultérieurement cet ASI d'instruction à l'aide de Business Object Designer.
2. Cliquez sur **OK**.

Traitement de l'objet de collection

L'ODA générera des objets métier pour les objets de collection présents dans les listes de paramètres d'entrée et de sortie. Les classes de collection suivantes sont prises en charge :

- Classes de liste : ArrayList, LinkList, Vector, Stack, HashSet
- Classes de mappage : HashMap, TreeMap, WeakHashMap, HashTable, Attributes

Vérification de la classe et de la méthode

Pour chaque objet de collection transmis en tant que type d'argument/retour, l'ODA affichera un message similaire à celui de la figure 11. Il indiquera l'objet de collection en cours de traitement.

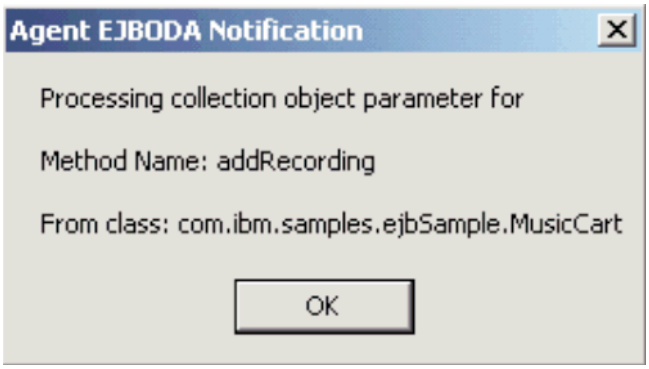


Figure 11. Vérification de la classe et de la méthode

Cliquez sur OK si les informations sur l'objet de collection sont correctes.

Sélection du nombre de types d'objets

Utilisez l'écran Parameter Object Type Info pour entrer le nombre de types d'objets différents que vous souhaitez conserver dans votre objet de collection.

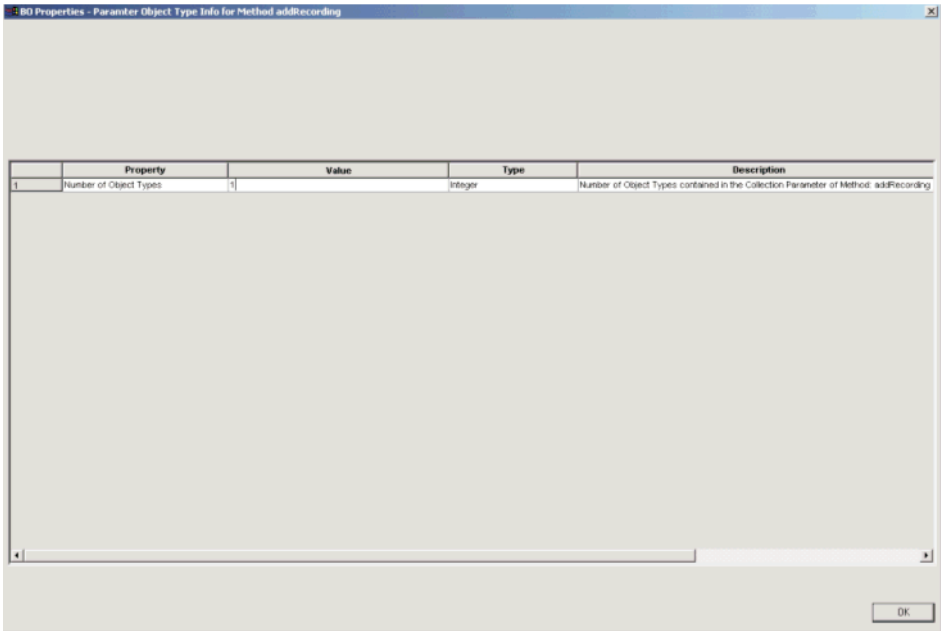


Figure 12. Sélection du nombre de types d'objets

Appuyez sur OK quand vous avez terminé.

L'ODA affichera l'avertissement suivant :

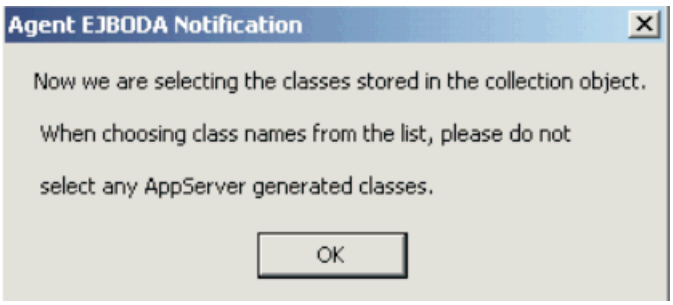


Figure 13. Vérification des paramètres

Vérifiez que vous n'avez pas sélectionné de classes générées par AppServer lorsque vous avez sélectionné votre nom de classe avant de cliquer sur OK.

Création d'objets de mappe

Lorsque l'ODA reconnaît une Map Class, dans laquelle une mappe est un objet qui mappe des clés sur des valeurs, les utilisateurs sont invités à sélectionner un type d'objet pour la clé et un autre pour les valeurs indiquées dans la figure 14. Chaque paire nom/valeur de l'objet de mappe procède au mappage sur deux objets métier : un pour la clé et un pour l'objet de valeur.

Remarque : Une mappe ne peut contenir de clés en double, et chaque clé ne peut se mapper que sur une valeur, au maximum.

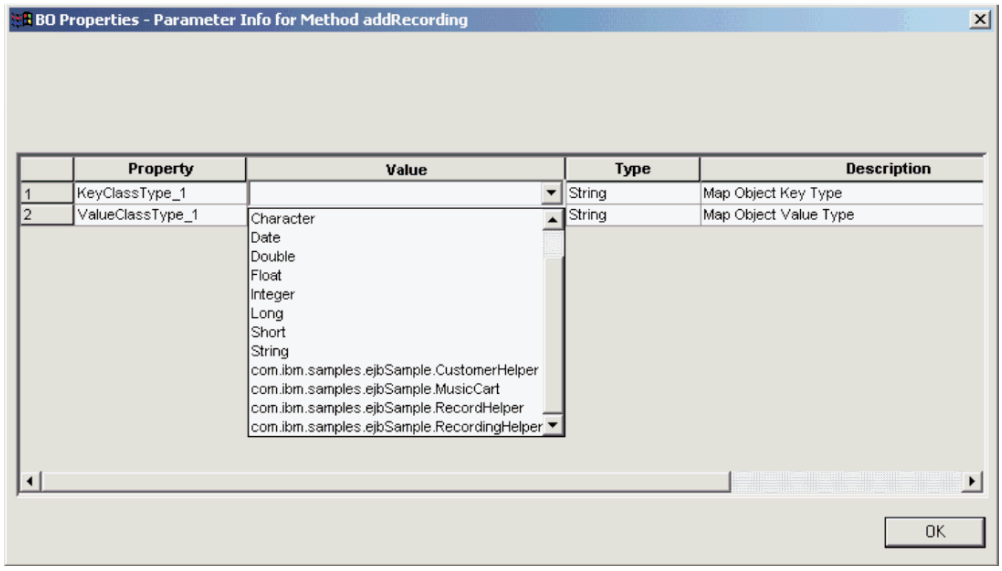


Figure 14. Création d'objets de mappe

Création d'objets de liste

Lorsque l'ODA reconnaît un Vector ou un autre objet de liste, vous devez sélectionner le type des objets que vous voulez conserver dans l'objet de collection, à partir du menu déroulant.

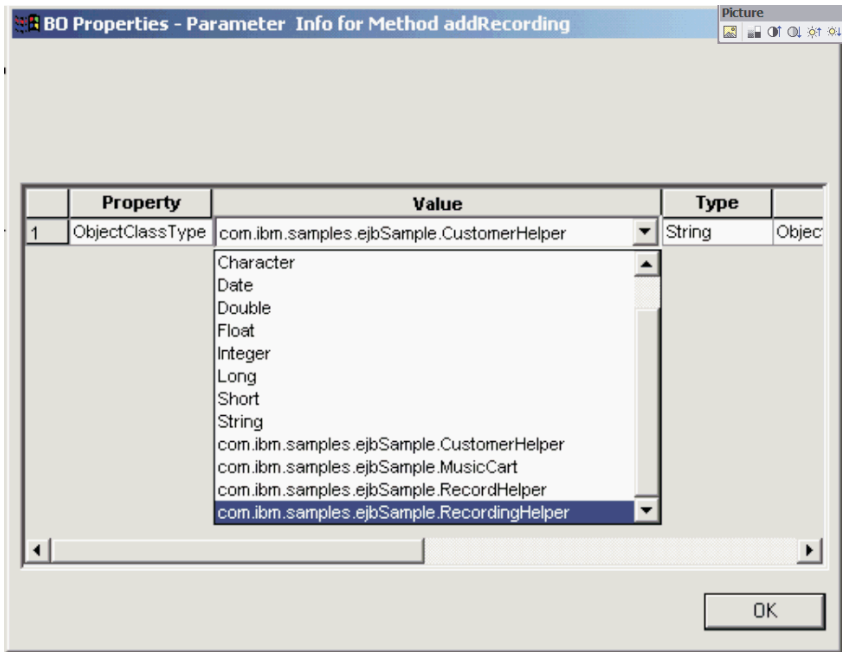


Figure 15. Sélection des types d'objet

Mappage de plusieurs méthodes getter/setter sur un même objet java

Si un JavaBean d'entreprise prend un objet java en tant qu'argument ou type de retour, et si l'objet java a de nombreux types de méthodes getter/setter, l'écran de la figure 16 s'affiche. Utilisez-le pour sélectionner les méthodes getter/setter à mapper sur l'attribut de votre objet métier.

Si vous ne sélectionnez pas getter/setter dans cet écran, les méthodes getter/setter seront mappées sur des objets métier et non sur des attributs d'objet métier.

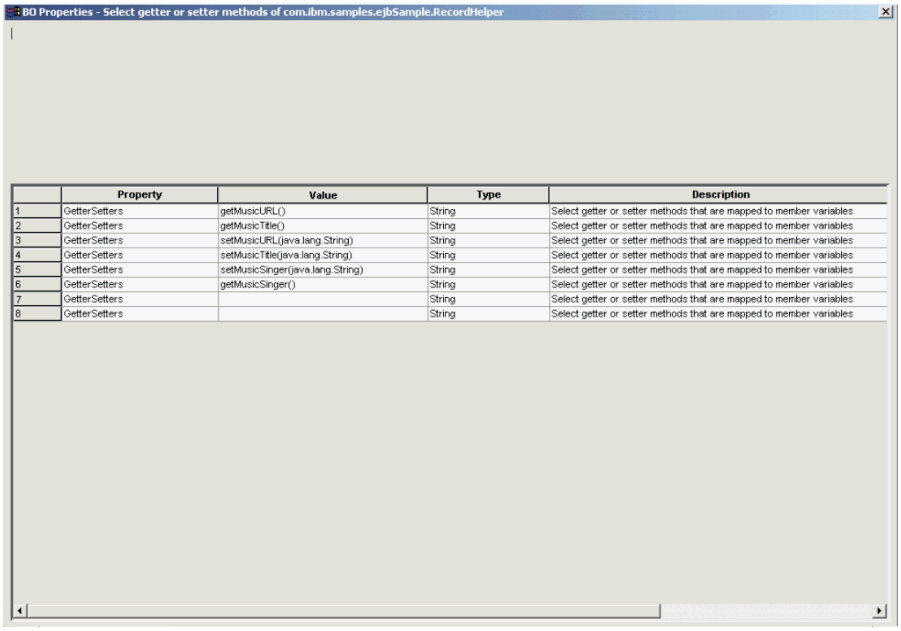


Figure 16. Mappage de plusieurs méthodes getter/setter sur l'attribut d'un objet métier

Ouverture ou enregistrement de l'objet métier

L'écran *Business Object Wizard - Step 6 of 6 - Save business objects* s'affiche.

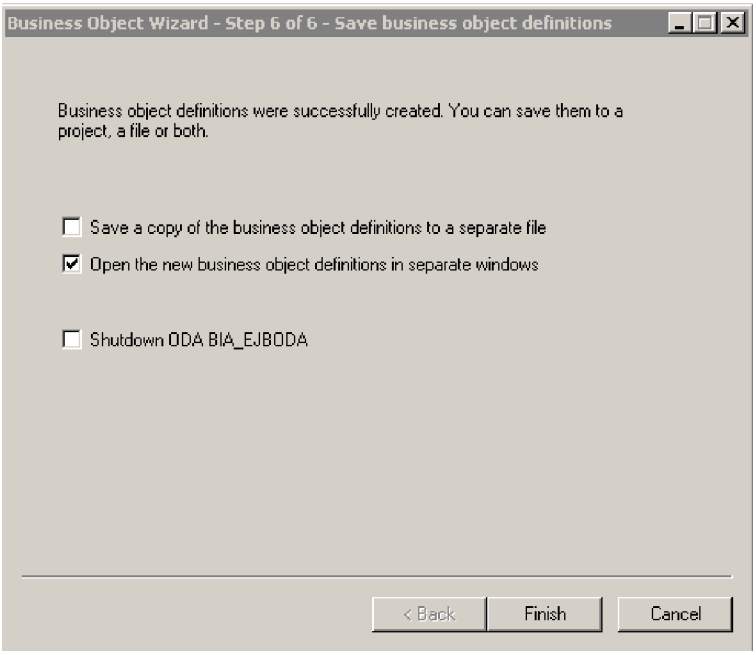


Figure 17. Ecran Save business objects

Vous pouvez aussi ouvrir de nouveaux objets métier dans une fenêtre séparée, dans Business Object Designer. Vous pouvez aussi (après avoir précisé une clé pour un objet métier de niveau supérieur) enregistrer les définitions d'objet métier générées dans un fichier.

Pour ouvrir les objets métier dans une fenêtre séparée :

1. Cochez **Open the new BOs in separate windows**.
2. Cliquez sur **Finish**. Chaque objet métier apparaît dans une fenêtre séparée, dans laquelle vous pouvez afficher et définir les informations ASI pour les objets métier et instructions d'objet métier que vous venez de créer. Pour plus d'informations, voir «Sélection des instructions et affectation du nom JNDI», à la page 44 et «Spécifiez l'ASI de l'instruction», à la page 45.

Pour enregistrer les objets métier dans un fichier (uniquement après avoir indiqué un clé pour l'objet métier parent) :

1. Sélectionnez **Save a copy of the business objects to a separate file**. Une boîte de dialogue s'affiche.
2. Indiquez l'emplacement dans lequel vous souhaitez enregistrer la copie des nouvelles définitions d'objet métier.

Business Object Designer enregistre les fichiers dans l'emplacement spécifié.

Si vous avez fini de travailler avec l'ODA, vous pouvez le fermer en cochant "Shutdown ODA EJBODA" avant de cliquer sur **Finish**.

Indication de l'ASI de l'attribut

Une fois que vous avez défini l'ASI de l'instruction (en modifiant une séquence de méthodes à exécuter pour chaque instruction), Business Object Designer affiche les attributs de l'objet métier. Pour plus d'informations sur l'ASI de l'attribut du connecteur EJB, voir «ASI de l'attribut», à la page 31.

Les attributs sont indiqués dans l'onglet **Attributes**, dans l'ordre dans lequel ils apparaissent dans la structure d'objet EJB, tel que défini par la valeur numérique de la colonne **Pos**. Des attributs d'objet EJB simples sont représentés en tant qu'attributs simples et leur ASI contient le nom et le type de l'attribut EJB d'origine.

L'écran indique le nom, le type et les informations ASI de chaque attribut. La figure 18, à la page 52 présente les attributs (complexes) de la méthode. L'attribut `getAllRecordsInfo` de l'objet métier a un ASI qui mappe l'attribut sur la méthode d'objet EJB d'origine. Dans cet exemple, la méthode d'origine est indiquée dans la colonne **App Spec Info** par l'ASI `method_name=getAllRecordInfo`.

L'attribut `getCustomer` de l'objet métier a un ASI qui mappe l'attribut sur la méthode d'objet EJB d'origine. Dans cet exemple, la méthode d'origine est indiquée dans la colonne **App Spec Info** par l'ASI `method_name=getCustomer`.

De plus, `getCustomer` (objet métier enfant) a l'attribut d'objet métier enfant `Return_Value`, utilisé pour capturer la valeur en retour de la méthode `getCustomer`. Dans le fichier JAR EJB, la méthode est définie comme ayant une valeur en retour de type `CustomerHelper`, c'est-à-dire une méthode. L'ASI d'attribut de `CustomerHelper` est défini sur `type=proxy`, car le type d'objet est une valeur en retour. Notez que si une méthode du fichier JAR EJB ne retourne pas de valeur, l'attribut `Return_Value` n'est pas inclus dans la liste des attributs d'objet métier.

Notez également que l'attribut `Return_Value` de type `CustomerHelper` a trois attributs d'objet enfant non-méthode : `name`, `password` et `email`. Dans le fichier JAR EJB d'origine dans lequel est définie la classe `MusicCartBean`, ils sont tous définis en tant que paramètres de type `String` de la méthode `CustomerHelper`. Dans l'objet

métier, l'ASI d'attribut de ces paramètres indique le nom et le type de la propriété Java (`String`) de chaque fichier JAR d'origine.

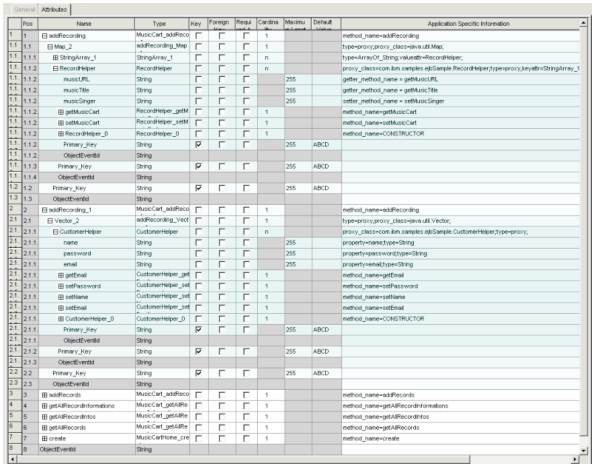


Figure 18. Définition de l'ASI d'attribut

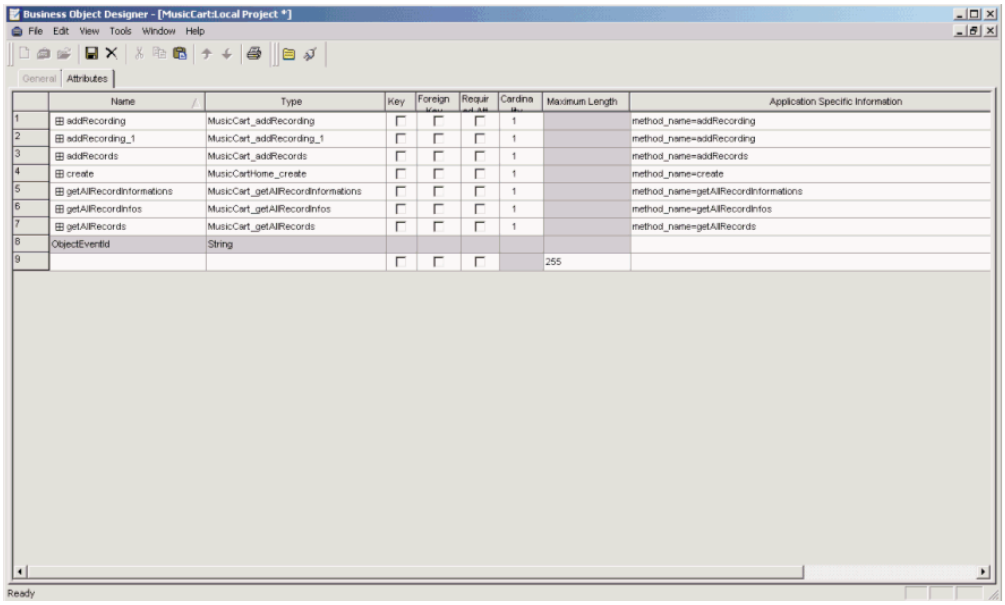


Figure 19. Définition de l'ASI d'attribut

Dans cet écran, indiquez si un objet parent est une clé (exigée par l'ODA pour enregistrer les objets métier dans un fichier séparé). Vous pouvez également utiliser cet écran pour définir des clés d'objets enfants si nécessaire et préciser les informations suivantes :

- L'attribut est-il obligatoire pour que le connecteur traite l'objet métier ? Si c'est le cas, cochez la case **Required**.
- La longueur maximum de l'attribut est-elle différente de la valeur affichée dans la colonne **Maximum Length** ?
- L'attribut a-t-il une valeur par défaut ? Si c'est le cas, saisissez la valeur dans la colonne **Default**.

Remarque : Même si vous pouvez créer un objet métier dans l'ODA (exécuté dans Business Object Designer) et définir des clés parents, ne procédez pas de cette manière pour configurer des clés étrangères. Les clés étrangères sont des métadonnées non ASI et doivent par conséquent toujours être configurées sans l'ODA (dans Business Object Designer. Cliquez sur **File** > **New** pour créer un nouvel objet métier sans utiliser l'ODA).

Indication de l'ASI de l'objet métier

Vous pouvez afficher et modifier l'ASI de l'objet métier. Pour plus d'informations sur l'ASI de l'objet métier, voir «ASI de l'objet métier», à la page 29.

L'ASI de l'objet métier est indiqué dans l'onglet **General**. La valeur ASI indiquée dans la zone **Business Object Level Application-specific information** contient le nom de l'interface éloignée correspondant à cet objet métier. Le connecteur utilise ces informations pour mapper un objet éloigné sur un objet métier.

L'onglet **General** répertorie également toutes les instructions prises en charge par l'objet métier en indiquant leur ASI telle que définie dans «Spécifiez l'ASI de l'instruction», à la page 45. Si une instruction est vide, la séquence de méthodes associée ne sera pas exécutée.

Téléchargement des fichiers d'objet métier

Les fichiers de définition d'objets métier nouvellement créés doivent être téléchargés dans le courtier d'intégration. Le processus varie selon que vous exécutez WebSphere InterChange Server, WebSphere MQ Integrator Broker ou WebSphere Application Server.

- **WebSphere InterChange Server** : Si vous avez enregistré vos fichiers de définition d'objet métier sur une machine locale et devez les télécharger dans le référentiel du serveur, consultez la documentation relative à l'implémentation d'InterChange Server.
- **WebSphere MQ Integrator Broker** : Vous devez exporter les définitions d'objet métier depuis Business Object Designer dans le courtier d'intégration. Pour plus d'informations, voir la documentation relative à l'implémentation de WebSphere MQ Integrator Broker.
- **WebSphere Application Server** : Pour plus d'informations, voir la documentation relative à l'implémentation de WebSphere Application Server.

Chapitre 6. Identification et résolution des erreurs

Le présent chapitre décrit comment le connecteur pour EJB gère les erreurs. Le connecteur génère des messages de consignation et de traçage. Ce chapitre décrit ces messages et donne des conseils de résolution des incidents.

Gestion des erreurs

Tous les messages générés par le connecteur sont conservés dans un fichier nommé `BIA_EJBConnector.txt`. (Le nom du fichier est déterminé par la propriété standard de configuration du connecteur `LogFileName`.) Chaque message se compose d'un numéro de message et du message lui-même :

```
Numéro de message
Texte du message
```

Le connecteur gère les erreurs spécifiques de la façon décrite dans les sections qui suivent.

ClassNotFound for proxy

Une exception est levée lorsque le connecteur ne trouve pas une interface éloignée dans le fichier JAR EJB fourni par l'utilisateur pour échanger les données avec le connecteur. Le connecteur consigne l'erreur avec le nom de la classe introuvable et retourne un code FAIL.

InstantiationException in Loader

Lorsque le connecteur reçoit le nom de classe du bean d'entreprise et tente de créer un objet de cette classe, une exception est levée s'il ne parvient pas à créer l'instance de l'objet. Le connecteur consigne l'erreur, qui inclut le nom de la classe de l'objet qui ne peut être instancié, et retourne un code FAIL.

Illegal AccessException in Loader or Invoker

Le connecteur lève une exception en cas de code non valide ou d'accès incorrect (public ou privé) à une méthode.

Le connecteur consigne l'erreur et retourne un code FAIL.

NoSuchMethodException in Invoker

Le connecteur lève une exception si une méthode est précisée sur l'objet métier et qu'elle n'existe pas dans l'objet de bean d'entreprise correspondant. Les méthodes sont chargées de façon dynamique, aussi cette exception est levée lorsque la méthode est introuvable dans la classe.

Le connecteur consigne l'erreur et retourne un code FAIL.

InvocationTargetException in Invoker

Le connecteur lève une exception lorsque l'application EJB (avec laquelle le connecteur échange des objets métier) lève une exception.

Le connecteur consigne l'erreur et retourne un code FAIL.

Invalid argument (CXIgnore) in a method object in Invoker

Le connecteur lève une exception lorsqu'une méthode est incluse dans l'ASI d'instruction de l'objet métier, mais que les arguments de cette méthode n'ont pas été renseignés.

Le connecteur consigne l'erreur et retourne un code FAIL.

Cast failure or wrong attribute type

Le connecteur lève une exception si une méthode d'objet EJB prend ou retourne un type de données différent de ce qui a été spécifié dans l'objet métier.

Le connecteur consigne l'erreur et retourne un code FAIL.

Invalid verb ASI

Le connecteur lève une exception si l'ASI d'instruction de l'objet métier qui lui est transmis est mis en forme de façon incorrecte ou s'il utilise une syntaxe erronée. C'est par exemple le cas lorsqu'un ASI d'instruction ne contient pas une séquence de méthode correcte.

Le connecteur consigne l'erreur et retourne un code FAIL.

App response timeout

Le connecteur lève une exception s'il perd la connexion avec le serveur d'applications sur lequel les beans d'entreprise sont déployés, ou s'il ne peut appeler de méthode EJB en raison d'un échec de connexion.

Le connecteur consigne l'erreur et retourne un code FAIL.

Consignation

Toutes les erreurs décrites dans «Gestion des erreurs», à la page 55 doivent être lues à partir d'un fichier de message (`BIA_EJBConnector.txt`).

Traçage

Le traçage est une fonction de débogage facultative que vous pouvez activer pour suivre de près le comportement d'un connecteur. Par défaut, les messages de trace sont consignés dans `STDOUT`. Pour plus d'informations sur la configuration des messages de trace, voir les propriétés de configuration du connecteur dans «Configuration du connecteur», à la page 15. Pour plus d'informations sur le traçage, en particulier savoir comment l'activer et le paramétrer, voir *Connector Development Guide*.

Le tableau 10, à la page 57 dresse la liste du contenu recommandé pour les niveaux de message de traçage du connecteur.

Tableau 10. Contenu des messages de trace

Niveau	Description
Niveau 0	Utilisez ce niveau pour les messages de trace qui identifient la version du connecteur. Aucun autre traçage n'est réalisé à ce niveau.
Niveau 1	Utilisez ce niveau pour les messages de trace qui : • Fournissent des informations sur le statut. • Fournissent des informations clés sur chaque objet métier traité. • Enregistrent à chaque fois qu'une unité d'exécution d'interrogation détecte un nouveau message dans une file d'attente d'entrée.
Niveau 2	Utilisez ce niveau pour les messages de trace qui : • Identifient le gestionnaire de BO utilisé pour chaque objet traité par le connecteur. • Signalent chaque fois qu'un objet métier est transmis au courtier d'intégration. • Signalent chaque fois qu'un objet métier de requête est reçu.
Niveau 3	Utilisez ce niveau pour les messages de trace qui : • Identifient les clés étrangères traitées, le cas échéant. Ces messages s'affichent quand le connecteur a rencontré une clé étrangère dans un objet métier ou quand il en définit une dans un objet métier. • Concernent le traitement d'un objet métier. C'est par exemple le cas lors de la détection d'une correspondance entre objets métier, ou d'un objet métier dans un tableau d'objets métier enfants.
Niveau 4	Utilisez ce niveau pour les messages de trace qui : • Identifient les informations spécifiques à l'application. C'est le cas lorsque des valeurs sont retournées par les méthodes qui traitent les zones d'information spécifiques à l'application dans les objets métier. • Identifient lorsque le connecteur entre ou quitte une fonction. Ces messages aident à procéder au traçage du flot de processus du connecteur. • Enregistrent tout traitement spécifique à l'unité d'exécution. Par exemple, si le connecteur engendre plusieurs unités d'exécution, un message signale la création de chacune.
Niveau 5	Utilisez ce niveau pour les messages de trace qui : • Indiquent l'initialisation du connecteur. Ce type de message peut inclure par exemple la valeur de chaque propriété de configurateur de connecteur, extraite du courtier. • Détaillent l'état de chaque unité d'exécution que le connecteur engendre pendant son exécution. • Représentent des instructions exécutées dans l'application. Le fichier de consignation du connecteur contient toutes les instructions exécutées dans l'application cible ainsi que la valeur des variables qui sont remplacées. • Enregistrent les suppressions d'objets métier. Le connecteur sort une représentation textuelle d'un objet métier avant le début du traitement (indiquant l'objet que le connecteur reçoit de la collaboration), et après la fin du traitement de l'objet (indiquant l'objet retourné par le connecteur à la collaboration).

Annexe A. Propriétés de configuration standard pour les connecteurs

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

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

Si votre adaptateur prend en charge DB2 Information Integrator, utilisez les options et les propriétés standard DB2 II (voir la colonne Remarque du tableau 11, à la page 61.)

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

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

Nouvelles propriétés

La propriété standard suivante a été ajoutée à cette édition :

- BOTrace

Présentation des propriétés de connecteur standard

Les connecteurs ont deux types de propriétés de configuration :

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

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

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

Démarrage de Connector Configurator

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

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

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

Présentation des valeurs des propriétés de configuration

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

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

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

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

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

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

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

Pour déterminer la manière dont une propriété donnée est mise à jour, reportez-vous à la colonne **Update Method** dans la fenêtre Connector Configurator, ou à la colonne Update Method dans le tableau 11, à la page 61.

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

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

Référence rapide des propriétés standard

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

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

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

Tableau 11. Récapitulatif des propriétés de configuration standard

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

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

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

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

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

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

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

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

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

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

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

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

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

Propriétés standard

Cette section décrit les propriétés standard de configuration du connecteur.

AdapterHelpName

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

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

AdminInQueue

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

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

AdminOutQueue

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

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

AgentConnections

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

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

La valeur par défaut de cette propriété est 1.

AgentTraceLevel

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

La valeur par défaut est `0`.

ApplicationName

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

La valeur par défaut est le nom du connecteur.

BiDi.Application

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

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

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

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

BiDi.Broker

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

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

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

BiDi.Metadata

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

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

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

BiDi.Transformation

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

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

La valeur par défaut est `false`.

BOTrace

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

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

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

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

La valeur par défaut est `false`.

BrokerType

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

CharacterEncoding

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

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

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

CommonEventInfrastructure

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

La valeur par défaut est `false`.

CommonEventInfrastructureContextURL

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

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

La valeur par défaut est une zone vide.

ConcurrentEventTriggeredFlows

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

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

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

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

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

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

La valeur par défaut est 1.

ContainerManagedEvents

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

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

- PollQuantity = 1 à 500
- SourceQueue = /SOURCEQUEUE

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

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

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

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

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

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

Il n'y a pas de valeur par défaut.

ControllerEventSequencing

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

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

La valeur par défaut est true.

ControllerStoreAndForwardMode

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

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

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

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

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

La valeur par défaut est `true`.

ControllerTraceLevel

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

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

La valeur par défaut est `0`.

DeliveryQueue

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

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

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

DeliveryTransport

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

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

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

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

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

WebSphere MQ et IDL

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

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

JMS

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

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

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

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

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

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

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

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

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

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

DuplicateEventElimination

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

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

La valeur par défaut est `false`.

EnableOidForFlowMonitoring

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

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

La valeur par défaut est `false`.

FaultQueue

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

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

jms.FactoryClassName

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

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

jms.ListenerConcurrency

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

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

La valeur par défaut est 1.

jms.MessageBrokerName

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

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

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

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

jms.NumConcurrentRequests

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

La valeur par défaut est 10.

jms.Password

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

Il n'y a pas de valeur par défaut.

jms.TransportOptimized

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

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

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

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

La valeur par défaut est `false`.

jms.UserName

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

Il n'y a pas de valeur par défaut.

JvmMaxHeapSize

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

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

La valeur par défaut est 128 Mo.

JvmMaxNativeStackSize

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

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

La valeur par défaut est 128 Ko.

JvmMinHeapSize

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

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

La valeur par défaut est 1 Mo.

ListenerConcurrency

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

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

La valeur par défaut est 1.

Locale

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

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

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

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

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

La valeur par défaut est `en_US`.

LogAtInterchangeEnd

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

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

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

La valeur par défaut est `false`.

MaxEventCapacity

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

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

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

La valeur par défaut est 2147483647.

MessageFileName

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

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

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

La valeur par défaut est `InterchangeSystem.txt`.

MonitorQueue

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

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

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

OADAutoRestartAgent

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

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

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

La valeur par défaut est `false`.

OADMaxNumRetry

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

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

La valeur par défaut est `1000`.

OADRetryTimeInterval

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

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

La valeur par défaut est `10`.

PollEndTime

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

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

Si l'exécution de l'adaptateur détecte :

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

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

PollFrequency

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

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

La valeurs suivantes sont valides pour cette propriété :

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

La valeur par défaut est `10000`.

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

PollQuantity

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

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

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

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

PollStartTime

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

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

Si l'exécution de l'adaptateur détecte :

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

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

RepositoryDirectory

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

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

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

RequestQueue

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

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

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

ResponseQueue

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

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

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

RestartRetryCount

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

La valeur par défaut est `7`.

RestartRetryInterval

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

Les valeurs possibles vont de 1 à 2147483647.

La valeur par défaut est `1`.

ResultsSetEnabled

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

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

La valeur par défaut est `false`.

ResultsSetSize

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

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

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

RHF2MessageDomain

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

Voici un exemple d'en-tête :

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

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

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

SourceQueue

La propriété SourceQueue désigne la file d'attente source JMS de l'architecture du connecteur qui assure la transmission garantie d'événements pour les connecteur activés par JMS qui utilisent un magasin d'événements JMS. Pour plus d'informations, voir «ContainerManagedEvents», à la page 70.

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

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

SynchronousRequestQueue

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

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

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

SynchronousRequestTimeout

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

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

La valeur par défaut est `0`.

SynchronousResponseQueue

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

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

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

TivoliMonitorTransactionPerformance

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

La valeur par défaut est `false`.

WireFormat

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

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

WsifSynchronousRequestTimeout

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

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

La valeur par défaut est 0.

XMLNameSpaceFormat

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

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

La valeur par défaut est `short`.

Annexe B. Connector Configurator

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

Connector Configurator vous permet de :

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

Les sujets traités dans cette annexe sont les suivants :

Présentation de Connector Configurator

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

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

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

Connector Configurator vous permet de :

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

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

d'intégration que vous exécutez. Par exemple, si vous utilisez WMQI comme courtier, vous exécutez Connector Configurator directement, et non à partir de System Manager (voir «Démarrage de Connector Configurator en mode autonome», à la page 86).

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 88 pour configurer un nouveau modèle.

Utilisation des connecteurs sous UNIX

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

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

Démarrage de Connector Configurator

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

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

Démarrage de Connector Configurator en mode autonome

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

Pour ce faire, procédez comme suit :

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

Vous pouvez choisir d'exécuter Connector Configurator en mode autonome pour créer le fichier, puis de vous connecter à System Manager afin de l'enregistrer dans un projet System Manager (voir «Remplissage d'un fichier de configuration», à la page 93).

Exécution de Connector Configurator à partir de System Manager

Vous pouvez exécuter Connector Configurator à partir de System Manager.

Pour exécuter Connector Configurator, procédez comme suit :

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

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

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

Création d'un modèle de propriétés spécifiques au connecteur

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

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

- Pour créer un modèle, voir «Création d'un modèle», à la page 88.
- Pour utiliser un fichier existant, il vous suffit de modifier un modèle existant et de l'enregistrer sous le nouveau nom. Vous pouvez trouver des modèles existants dans le répertoire `\WebSphereAdapters\bin\Data\App`.

Création d'un modèle

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

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

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

Indication des caractéristiques générales

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

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

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

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

Indication de valeurs

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

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

Pour créer une valeur de propriété, procédez comme suit :

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

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

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

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

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

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

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

Définition des dépendances

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

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

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

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

Configuration des noms de chemins

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

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

Création d'un fichier de configuration

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

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

Pour démarrer Connector Configurator :

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Utilisation d'un fichier existant

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

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

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

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

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

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

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

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

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

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

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

Remplissage d'un fichier de configuration

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

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

1. Dans l'onglet **Standard Properties (Propriétés standard)**, sélectionnez la zone de valeur pour la propriété `BrokerType`. Dans le menu déroulant, sélectionnez la valeur `ICS`, `WMQI` ou `WAS`.
2. L'onglet Standard Properties affiche les propriétés de connecteur associées au courtier sélectionné. La table indique les **Property name**, **Value**, **Type**, **Subtype** (si le Type est String), **Description** et **Update Method**.
3. Vous pouvez enregistrer le fichier maintenant ou renseigner les autres zones relatives à la configuration, comme décrit dans «Indication des définitions d'objets métier pris en charge», à la page 96.
4. Une fois la configuration terminée, cliquez sur **File>Save>To Project** ou sur **File>Save>To File**.

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

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

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

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

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

Définition des propriétés d'un fichier de configuration

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

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

- Propriétés standard
- Propriétés spécifiques au connecteur

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

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

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

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

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

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

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

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

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

Définition des propriétés standard du connecteur

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

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

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

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

Remarque : Si le bouton n'apparaît pas, c'est qu'il n'existe pas d'aide étendue pour cette propriété.

Les fichiers Extended Help sont installés dans le répertoire <*ProductDir*>\bin\Data\Std\Help\<*RegionalSetting*>\.

Configuration des propriétés spécifiques au connecteur

Vous pouvez ajouter ou modifier des noms, définir des valeurs, supprimer une propriété spécifique ou la chiffrer. La longueur par défaut d'une propriété est de 255 caractères.

1. Cliquez avec le bouton droit dans la partie supérieure gauche de la grille. Une barre de menus contextuelle apparaît. Cliquez sur **Add** pour ajouter une propriété. Pour ajouter une propriété enfant, cliquez avec le bouton droit sur le numéro de la ligne parent et cliquez sur **Add child**.
2. Entrez une valeur pour la propriété ou la propriété enfant.

   Remarque : Si la propriété est de Type String, vous pouvez sélectionner un sous-type dans la liste déroulante. Ce sous-type sert pour la validation étendue de la propriété.
3. Pour chiffrer une propriété, cochez la case **Encrypt**.

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

   Remarque : Si le bouton n'apparaît pas, c'est qu'il n'existe pas d'aide étendue pour cette propriété.
5. Vous pouvez enregistrer ou ignorer les modifications, comme décrit pour «Définition des propriétés standard du connecteur», à la page 95.

Si les fichiers Extended Help sont installés et que la propriété AdapterHelpName n'est pas renseignée, Connector Configurator pointera sur les fichiers Extended Help spécifiques au connecteur, dans le répertoire <*ProductDir*>\bin\Data\App\Help\<*RegionalSetting*>\. Sinon, Connector Configurator pointera sur les fichiers Extended Help spécifiques au connecteur, dans le répertoire <*ProductDir*>\bin\Data\App\Help\<*AdapterHelpName*>\<*RegionalSetting*>\. Voir la propriété AdapterHelpName, décrite dans l'annexe Propriétés standard.

La zone Update Method affichée pour chaque propriété indique si le redémarrage d'un composant ou d'un agent est nécessaire à l'activation des valeurs modifiées.

Important : La modification du nom prédéfini d'une propriété de connecteur spécifique à l'application peut entraîner l'échec d'un connecteur. Le connecteur peut nécessiter certains noms de propriété pour se connecter à une application ou s'exécuter correctement.

Chiffrement des propriétés du connecteur

Pour chiffrer les propriétés spécifiques à l'application, cochez la case **Encrypt** dans la fenêtre des propriétés spécifiques au connecteur. Pour déchiffrer une valeur, décochez la case **Encrypt**, entrez la valeur appropriée dans la boîte de dialogue **Verification** et cliquez sur **OK**. Si la valeur entrée est correcte, elle est déchiffrée et s'affiche.

Le guide d'utilisateur de l'adaptateur pour chaque connecteur contient la liste et la description de chaque propriété ainsi que sa valeur par défaut.

Si une propriété a plusieurs valeurs, la case **Encrypt** apparaît pour la première valeur de la propriété. Lorsque vous sélectionnez **Encrypt**, toutes les valeurs de la propriété sont chiffrées. Pour déchiffrer plusieurs valeurs d'une propriété, décochez la case **Encrypt** pour la première valeur de la propriété, puis entrez la nouvelle valeur dans la boîte de dialogue **Verification**. Si la valeur entrée est une correspondance, toutes les valeurs multiples sont déchiffrées.

Méthode de mise à jour

Reportez-vous aux descriptions des méthodes de mise à jour, dans l'annexe Propriétés standard, sous «Présentation des valeurs des propriétés de configuration», à la page 60.

Indication des définitions d'objets métier pris en charge

Utilisez l'onglet **Supported Business Objects** dans Connector Configurator pour indiquer les objets métier que le connecteur utilisera. Vous devez indiquer les objets métier génériques et les objets métier spécifiques à l'application, et indiquer les associations pour les mappes entre les objets métier.

Remarque : Certains connecteurs nécessitent que des objets métier soient indiqués comme étant pris en charge pour pouvoir exécuter la notification des événements ou une configuration supplémentaire (à l'aide des méta-objets) avec leurs applications. Pour plus d'informations, voir *Connector Development Guide for C++* ou *Connector Development Guide for Java*.

Si ICS est votre courtier

Pour indiquer qu'une définition d'objet métier est prise en charge par le connecteur ou modifier les paramètres de prise en charge d'une définition d'objet métier existante, cliquez sur l'onglet **Supported Business Objects** et utilisez les zones suivants :

Nom de l'objet métier : Pour indiquer qu'une définition d'objet métier est prise en charge par le connecteur, avec System Manager en cours d'exécution, procédez comme suit :

1. Cliquez dans une zone vide de la liste **Business Object Name**. Une liste déroulante s'affiche, avec toutes les définitions d'objet métier qui existent dans le projet System Manager.
2. Cliquez sur un objet métier pour l'ajouter.
3. Définissez la zone **Agent Support** (décrite plus bas) pour l'objet métier.
4. Dans le menu File de la fenêtre Connector Configurator, cliquez sur **Save to Project**. La définition révisée du connecteur, qui contient la prise en charge sélectionnée pour la définition de l'objet métier ajouté, est enregistrée dans un projet ICL (Integration Component Library) de System Manager.

Pour supprimer un objet métier dans la liste des objets métier pris en charge :

1. Pour sélectionner la zone d'un objet métier, cliquez sur le numéro situé à gauche de l'objet métier.
2. Dans le menu **Edit** de la fenêtre Connector Configurator, cliquez sur **Delete Row**. L'objet métier est supprimé de la liste.
3. Dans le menu **File**, cliquez sur **Save to Project**.

La suppression d'un objet métier dans la liste des objets métier pris en charge modifie la définition du connecteur et rend l'objet métier supprimé inutilisable dans cette implémentation du connecteur. Elle n'affecte pas le code du connecteur et ne supprime pas la définition de l'objet métier dans System Manager.

Prise en charge de l'agent : Si un objet métier dispose de la prise en charge de l'agent, le système tente d'utiliser cet objet métier pour fournir des données à une application via l'agent du connecteur.

En général, les objets métier spécifiques à l'application pour un connecteur sont pris en charge par l'agent de ce connecteur, mais les objets métier génériques ne le sont pas.

Pour indiquer que l'objet métier est pris en charge par l'agent du connecteur, cochez la case **Agent Support**. La fenêtre Connector Configurator ne valide pas vos sélections pour Agent Support.

Niveau de transaction maximum : Le niveau de transaction maximum d'un connecteur correspond au niveau de transaction le plus élevé pris en charge par le connecteur.

Pour la plupart des connecteurs, `Best Effort` est la seule valeur possible.

Vous devez redémarrer le serveur pour que les modifications prennent effet.

Si votre courtier est un courtier de messages WebSphere

Si vous utilisez le mode autonome (non connecté à System Manager), vous devez entrer manuellement le nom de l'objet métier.

Si System Manager est en cours d'exécution, vous pouvez cocher la case située sous la colonne **Business Object Name** dans l'onglet **Supported Business Objects**. Une boîte de dialogue mixte affiche la liste des objets métier disponibles dans le projet Integration Component Library auquel le connecteur appartient. Dans cette liste, sélectionnez l'objet métier de votre choix.

La zone **Message Set ID** est facultative pour WebSphere Business Integration Message Broker 5.0, et sa valeur ne doit pas nécessairement être unique le cas échéant. Cependant, pour WebSphere MQ Integrator et Integrator Broker 2.1, vous devez indiquer un **ID** unique.

Si WAS est votre courtier

Lorsque vous sélectionnez WebSphere Application Server comme type de courtier, Connector Configurator ne nécessite pas les ID d'ensemble de messages. L'onglet **Supported Business Objects** contient la colonne **Business Object Name** pour les objets métier pris en charge uniquement.

Si vous utilisez le mode autonome (non connecté à System Manager), vous devez entrer manuellement le nom de l'objet métier.

Si System Manager est en cours d'exécution, vous pouvez cocher la case située sous la colonne Business Object Name dans l'onglet Supported Business Objects. Une boîte de dialogue mixte affiche la liste des objets métier disponibles dans le projet Integration Component Library auquel le connecteur appartient. Dans cette liste, sélectionnez l'objet métier de votre choix.

Mappes associées (ICS)

Chaque connecteur prend en charge la liste des définitions des objets métier et leurs mappes associées actives dans WebSphere InterChange Server. Cette liste apparaît lorsque vous sélectionnez l'onglet **Associated Maps**.

La liste des objets métier contient l'objet métier spécifique à l'application pris en charge par l'agent et l'objet générique correspondant que le contrôleur envoie à la collaboration de souscription. L'association d'une mappe détermine la mappe qui sera utilisée pour transformer l'objet métier spécifique à l'application en objet métier générique, ou inversement.

Si vous utilisez des mappes uniquement définies pour des objets métier source et cible spécifiques, les mappes sont déjà associées aux objets métier appropriés lorsque vous affichez l'écran, et vous n'avez pas besoin de (ou ne pouvez pas) les modifier.

Si plusieurs mappes sont disponibles pour un objet métier pris en charge, vous devez lier de manière explicite cet objet métier à la mappe qu'il doit utiliser.

L'onglet **Associated Maps** affiche les zones suivantes :

- **Business Object Name**

  Il s'agit des objets métier pris en charge par ce connecteur, comme indiqué dans l'onglet **Supported Business Objects**. Si vous indiquez des objets métier

supplémentaires dans l'onglet Supported Business Objects, ils sont reflétés dans cette liste une fois que vous avez enregistré les modifications en sélectionnant **Save to Project** dans le menu **File** de la fenêtre Connector Configurator.

- **Associated Maps**

  L'écran affiche toutes les mappes installées sur le système à utiliser avec les objets métier pris en charge du connecteur. L'objet métier source pour chaque mappe s'affiche à gauche du nom de la mappe, dans l'écran **Business Object Name**.

- **Liaison explicite**

  Dans certains cas, vous devrez peut-être lier de manière explicite une mappe associée.

  Une liaison explicite est requise uniquement lorsque plusieurs mappes existent pour un objet métier pris en charge spécifique. Lorsque ICS s'amorce, il tente de lier automatiquement une mappe à chaque objet métier pris en charge pour chacun des connecteurs. Si plusieurs mappes prennent le même objet métier comme entrée, le serveur tente de localiser et de lier une mappe qui correspond au sur-ensemble des autres.

  Si aucune mappe n'est le sur-ensemble des autres, le serveur ne peut pas lier l'objet métier à une seule mappe et vous devrez définir la liaison de manière explicite.

  Pour lier une mappe de manière explicite, procédez comme suit :

  1. Dans la colonne **Explicit**, cochez la case correspondant à la mappe à lier.
  2. Sélectionnez la mappe que vous souhaitez associer à l'objet métier.
  3. Dans le menu **File** de la fenêtre Connector Configurator, cliquez sur **Save to Project**.
  4. Déployez le projet jusqu'à ICS.
  5. Réamorcez le serveur pour que les modifications prennent effet.

Ressources (ICS)

L'onglet **Resource** vous permet de définir une valeur qui détermine si l'agent du connecteur gérera plusieurs processus simultanément, et dans quelle mesure, à l'aide du parallélisme de l'agent du connecteur.

Tous les connecteurs ne prennent pas en charge cette fonction. Si vous exécutez un agent de connecteur conçu dans Java pour être multithread, nous vous recommandons de ne pas utiliser cette fonction dans la mesure où il est généralement plus efficace d'utiliser plusieurs unités d'exécution plutôt que plusieurs processus.

Messagerie (ICS)

l'onglet **Messaging** vous permet de configurer les propriétés de messagerie. Les propriétés de messagerie sont disponibles uniquement si vous avez défini MQ comme la valeur de la propriété standard `DeliveryTransport` et ICS comme le type de courtier. Ces propriétés affectent la manière dont le connecteur utilisera les files d'attente.

Validation des files d'attente de messages

Avant de valider une file d'attente de messages, vous devez :

- Vous assurer que WebSphere MQ Series est installé.
- Créer sur la machine hôte une file d'attente de messages avec le canal et le port.
- Configurer une connexion sur la machine hôte.

Pour valider la file d'attente, vous utiliserez le bouton Validate à droite des zones Messaging Type et Host Name, dans l'onglet Messaging.

Sécurité (ICS)

l'onglet **Security** du Connector Configurator permet de définir le niveau de confidentialité d'un message. Cette propriété n'est utilisable que si la propriété DeliveryTransport est définie sur JMS.

Par défaut, Privacy est désactivé. Pour l'activer, cochez la case **Privacy**.

Le **Keystore Target System Absolute Pathname** (Chemin absolu du magasin de clés du système cible) est :

- Pour Windows :
  `<ProductDir>\connectors\security\<connectorname>.jks`
- pour UNIX :
  `opt/IBM/WebSphereAdapters/connectors/security/<connectorname>.jks`

Ce chemin et le fichier qu'il indique doivent être sur le système qui vous servira à démarrer le connecteur, c'est-à-dire le système cible.

Vous pouvez utiliser le bouton Browse à droite, à condition que le système cible soit le seul en cours de fonctionnement. Ce bouton est désactivé jusqu'à ce que **Privacy** soit activé et que le **Target System** de la barre de menus soit défini sur Windows.

Le **Message Privacy Level** peut être défini comme suit pour les trois catégories de messages (All Messages, All Administrative Messages, et All Business Object Messages) :

- La valeur par défaut "" indique qu'aucun niveau de confidentialité n'a été défini pour une catégorie de messages.
- none
  Cette valeur n'a pas le même sens que la valeur par défaut : elle indique le choix délibéré d'attribuer ce niveau de confidentialité à une catégorie de messages.
- integrity
- privacy
- integrity_plus_privacy

La fonction **Key Maintenance** permet de générer, importer et exporter des clés publiques pour le serveur et l'adaptateur.

- La sélection de **Generate Keys** ouvre la boîte de dialogue correspondante, avec les valeurs par défaut pour l'outil qui générera les clés.
- Le magasin de clés est par défaut celui que vous avez indiqué pour **Keystore Target System Absolute Pathname** dans l'onglet Security.
- Lorsque vous cliquez sur OK, les entrées sont validées, le certificat est généré et la sortie est envoyée à la fenêtre de connexion du Connector Configurator.

Avant d'importer un certificat dans le magasin de clés de l'adaptateur, vous devez l'exporter depuis le magasin de clés du serveur. Pour cela, sélectionnez **Export Adapter Public Key**, ce qui ouvre la fenêtre correspondante.

- Par défaut, le certificat exporté prend les valeurs du magasin de clés, à l'exception de son extension qui est <nomdefichier>.cer.

Lorsque vous sélectionnez **Import Server Public Key**, la boîte de dialogue correspondante s'ouvre.

- Par défaut, le certificat importé prend la valeur <*ProductDir*>\bin\ics.cer (si le fichier existe sur le système).
- Le nom de serveur doit être la Certificate Association d'importation. Si un serveur est enregistré, vous pouvez le sélectionner dans la liste déroulante.

La fonction **Adapter Access Control** n'est activée que si DeliveryTransport est définie sur IDL. Par défaut l'adaptateur se connecte avec l'identité d'invité (guest). Si la case **Use guest identity** n'est pas cochée, les zones **Adapter Identity** et **Adapter Password** sont accessibles.

Définition des valeurs du fichier de trace ou du fichier journal

Lorsque vous ouvrez le fichier de configuration ou le fichier de définitions d'un connecteur, Connector Configurator utilise les valeurs de journalisation et de trace de ce fichier comme valeurs par défaut. Vous pouvez modifier ces valeurs dans Connector Configurator.

Pour modifier les valeurs de journalisation et de trace, procédez comme suit :

1. Cliquez sur l'onglet **Trace/Log Files**.
2. Pour la journalisation ou la fonction de trace, vous pouvez écrire des messages à l'un des composants suivants :
   - A la console (STDOUT) :
     Ecrit des messages de journalisation ou de trace à l'écran STDOUT.

     Remarque : Vous pouvez utiliser l'option STDOUT de l'onglet **Trace/Log Files** pour les connecteurs s'exécutant sur la plate-forme Windows.
   - A un fichier :
     Ecrit des messages de journalisation ou de trace vers un fichier indiqué. Pour indiquer le fichier, cliquez sur le bouton du répertoire (ellipse), accédez à l'emplacement de votre choix, indiquez un nom de fichier et cliquez sur **Save**. Les messages de journalisation ou de trace sont écrits vers le fichier et l'emplacement indiqués.

     Remarque : Les fichiers de journalisation et de trace sont de simples fichiers texte. Vous pouvez utiliser l'extension de votre choix lorsque vous définissez les noms de fichier. Cependant, pour les fichiers de trace, nous vous recommandons d'utiliser l'extension `.trace` plutôt que l'extension `.trc`, afin d'éviter toute confusion avec les autres fichiers pouvant résider sur le système. Pour les fichiers de journalisation, les extensions classiques sont `.log` et `.txt`.

Gestionnaires de données

La section des gestionnaires de données est disponible pour la configuration uniquement si vous avez indiqué une valeur JMS pour ContainerManagedEvents. Tous les adaptateurs n'utilisent pas les gestionnaires de données.

Pour connaître les valeurs à utiliser pour ces propriétés, reportez-vous aux descriptions sous `ContainerManagedEvents` dans l'Annexe A, Propriétés standard. Pour plus d'informations, voir *Connector Development Guide for C++* ou *Connector Development Guide for Java*.

Enregistrement de votre fichier de configuration

Une fois que vous avez configuré votre connecteur, enregistrez son fichier de configuration. Connector Configurator enregistre le fichier dans le mode courtier que vous avez sélectionné pendant la configuration. La barre de titre de Connector Configurator affiche toujours le mode courtier (ICS, WMQI ou WAS) en cours d'utilisation.

Le fichier est enregistré en tant que document XML. Pour enregistrer le document XML, vous avez trois possibilités :

- dans System Manager, en tant que fichier avec l'extension `*.con` dans le projet ICL ;
- dans un répertoire que vous avez indiqué ;
- en mode autonome, en tant que fichier avec l'extension `*.cfg` dans un répertoire (par défaut, le fichier est enregistré dans `\WebSphereAdapters\bin\Data\App`) ;
- dans un projet WebSphere Application Server, le cas échéant.

Pour plus d'informations sur l'utilisation des projets dans System Manager et sur le déploiement, voir les guides d'implémentation suivants :

- Pour ICS : *Implementation Guide for WebSphere InterChange Server*
- Pour les courtiers de messages WebSphere : *Implementing Adapters with WebSphere Message Brokers*
- Pour WAS : *Implementing Adapters with WebSphere Application Server*

Modification d'un fichier de configuration

Vous pouvez modifier les paramètres du courtier d'intégration pour un fichier de configuration existant. Cela vous permet d'utiliser le fichier comme modèle pour la création d'un nouveau fichier de configuration que vous pouvez utiliser avec un autre courtier.

Remarque : Vous devrez modifier d'autres propriétés de configuration ainsi que la propriété du mode courtier si vous changez de courtiers d'intégration.

Pour modifier votre sélection de courtier dans un fichier de configuration existant (facultatif) :

- Ouvrez le fichier de configuration existant dans Connector Configurator.
- Sélectionnez l'onglet **Standard Properties**.
- Dans la zone **BrokerType** de l'onglet Standard Properties, sélectionnez la valeur appropriée pour votre courtier.
  Lorsque vous modifiez la valeur courante, les onglets disponibles et les sélections de zones de la fenêtre des propriétés changent immédiatement, pour ne montrer que les onglets et zones qui correspondent au courtier que vous avez sélectionné.

Exécution de la configuration

Une fois que vous avez créé un fichier de configuration pour un connecteur et que vous l'avez modifié, assurez-vous que le connecteur peut localiser le fichier de configuration lorsqu'il démarre.

Pour ce faire, ouvrez le fichier de démarrage utilisé pour le connecteur et vérifiez que le nom de fichier et l'emplacement utilisés pour le fichier de configuration du connecteur correspondent exactement au nom attribué au fichier et au répertoire ou au chemin d'accès dans lequel vous l'avez placé.

Utilisation de Connector Configurator dans un environnement globalisé

Connector Configurator est globalisé et peut gérer la conversion des caractères entre le fichier de configuration et le courtier d'intégration. Connector Configurator utilise le codage natif. Lorsqu'il écrit dans le fichier de configuration, il utilise le codage UTF-8.

Connector Configurator prend en charge les caractères qui n'existent pas en anglais dans :

- toutes les zones de valeur ;
- le chemin d'accès au fichier journal et au fichier de trace (indiqué dans l'onglet **Trace/Log files**).

La liste déroulante pour les propriétés de configuration standard `CharacterEncoding` et `Locale` affiche uniquement un sous-ensemble des valeurs prises en charge. Pour ajouter d'autres valeurs à cette liste, vous devez modifier manuellement le fichier `\Data\Std\stdConnProps.xml` dans le répertoire produit.

Par exemple, pour ajouter l'environnement local `en_GB` à la liste des valeurs pour la propriété `Locale`, ouvrez le fichier `stdConnProps.xml` et ajoutez la ligne en caractère gras comme indiqué ci-dessous :

```
<Property name="Locale"
isRequired="true"
updateMethod="component restart">
                <ValidType>String</ValidType>
            <ValidValues>
                                <Value>ja_JP</Value>
                                <Value>ko_KR</Value>
                                <Value>zh_CN</Value>
                                <Value>zh_TW</Value>
                                <Value>fr_FR</Value>
                                <Value>de_DE</Value>
                                <Value>it_IT</Value>
                                <Value>es_ES</Value>
                                <Value>pt_BR</Value>
                                <Value>en_US</Value>
                                <Value>en_GB</Value>
                    <DefaultValue>en_US</DefaultValue>
            </ValidValues>
    </Property>
```

Informations légales

Le présent document peut contenir des informations ou des références concernant certains produits, logiciels ou services IBM non annoncés dans ce pays.

Pour plus de détails, référez-vous aux documents d'annonce disponibles dans votre pays, ou adressez-vous à votre partenaire d'échange IBM. Toute référence à un produit, logiciel ou service IBM n'implique pas que seul ce produit, logiciel ou service puisse être utilisé. Tout autre élément fonctionnellement équivalent peut être utilisé, s'il n'enfreint aucun droit d'IBM. Il est de la responsabilité de l'utilisateur d'évaluer et de vérifier lui-même les installations et applications réalisées avec des produits, logiciels ou services non expressément référencés par IBM.

IBM peut détenir des brevets ou des demandes de brevet couvrant les produits mentionnés dans le présent document. La remise de ce document ne vous donne aucun droit de licence sur ces brevets ou demandes de brevet. Si vous désirez recevoir des informations concernant l'acquisition de licences, veuillez en faire la demande par écrit à l'adresse suivante :

IBM EMEA Director of Licensing
IBM Europe Middle-East Africa
Tour Descartes
La Défense 5
2, avenue Gambetta
92066 - Paris-La Défense CEDEX
France

Pour le Canada, veuillez adresser votre courrier à :

IBM Director of Commercial Relations
IBM Canada Ltd.
3600 Steeles Avenue East
Markham, Ontario
L3R 9Z7
Canada

Les informations sur les licences concernant les produits utilisant un jeu de caractères double octet peuvent être obtenues par écrit à l'adresse suivante :

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

Le paragraphe suivant ne s'applique ni au Royaume-Uni, ni dans aucun pays dans lequel il serait contraire aux lois locales.

LE PRESENT DOCUMENT EST LIVRE EN L'ETAT, EN L'ETAT SANS AUCUNE GARANTIE EXPLICITE OU IMPLICITE. IBM DECLINE NOTAMMENT TOUTE RESPONSABILITE RELATIVE A CES INFORMATIONS EN CAS DE CONTREFAÇON AINSI QU'EN CAS DE DEFAUT D'APTITUDE A L'EXECUTION D'UN TRAVAIL DONNE. Certaines juridictions n'autorisent pas l'exclusion des garanties implicites, auquel cas l'exclusion ci-dessus ne vous sera pas applicable.

Le présent document peut contenir des inexactitudes ou des coquilles. Il est mis à jour périodiquement. Chaque nouvelle édition inclut les mises à jour. IBM peut, à tout moment et sans préavis, modifier les produits et logiciels décrits dans ce document.

Les références à des sites Web non IBM sont fournies à titre d'information uniquement et n'impliquent en aucun cas une adhésion aux données qu'ils contiennent. Les éléments figurant sur ces sites Web ne font pas partie des éléments du présent produit IBM et l'utilisation de ces sites relève de votre seule responsabilité.

IBM pourra utiliser ou diffuser, de toute manière qu'elle jugera appropriée et sans aucune obligation de sa part, tout ou partie des informations qui lui seront fournies.

Les licenciés souhaitant obtenir des informations permettant : (i) l'échange des données entre des logiciels créés de façon indépendante et d'autres logiciels (dont celui-ci), et (ii) l'utilisation mutuelle des données ainsi échangées, doivent adresser leur demande à :

IBM Corporation
577 Airport Blvd., Suite 800
Burlingame, CA 94010
U.S.A.

Ces informations peuvent être soumises à des conditions particulières, prévoyant notamment le paiement d'une redevance.

Le logiciel sous licence décrit dans ce document et tous les éléments sous licence disponibles s'y rapportant sont fournis par IBM conformément aux dispositions de l'ICA, des Conditions internationales d'utilisation des logiciels IBM ou de tout autre accord équivalent.

Les données de performance indiquées dans ce document ont été déterminées dans un environnement contrôlé. Par conséquent, les résultats peuvent varier de manière significative selon l'environnement d'exploitation utilisé. Certaines mesures évaluées sur des systèmes en cours de développement ne sont pas garanties sur tous les systèmes disponibles. En outre, elles peuvent résulter d'extrapolations. Les résultats peuvent donc varier. Il incombe aux utilisateurs de ce document de vérifier si ces données sont applicables à leur environnement d'exploitation.

Les informations concernant des produits non IBM ont été obtenues auprès des fournisseurs de ces produits, par l'intermédiaire d'annonces publiques ou via d'autres sources disponibles. IBM n'a pas testé ces produits et ne peut confirmer l'exactitude de leurs performances ni leur compatibilité. Elle ne peut recevoir aucune réclamation concernant des produits non IBM. Toute question concernant les performances de produits non IBM doit être adressée aux fournisseurs de ces produits.

Toute instruction relative aux intentions d'IBM pour ses opérations à venir est susceptible d'être modifiée ou annulée sans préavis, et doit être considérée uniquement comme un objectif.

Le présent document peut contenir des exemples de données et de rapports utilisés couramment dans l'environnement professionnel. Ces exemples mentionnent des noms fictifs de personnes, de sociétés, de marques ou de produits à des fins

illustratives ou explicatives uniquement. Toute ressemblance avec des noms de personnes, de sociétés ou des données réelles serait purement fortuite.

LICENCE DE COPYRIGHT :

Le présent logiciel contient des exemples de programmes en langage source, destinés à illustrer les techniques de programmation sur différentes platesformes d'exploitation. Vous avez le droit de copier, de modifier et de distribuer ces exemples de programmes sous quelque forme que ce soit et sans paiement d'aucune redevance à IBM, à des fins de développement, d'utilisation, de vente ou de distribution de programmes d'application conformes aux interfaces de programmation des plateformes pour lesquels ils ont été écrits ou aux interfaces de programmation IBM. Ces exemples de programmes n'ont pas été rigoureusement testés dans toutes les conditions. Par conséquent, IBM ne peut garantir expressément ou implicitement la fiabilité, la maintenabilité ou le fonctionnement de ces programmes.

Si vous visualisez ces informations en ligne, il se peut que les photographies et illustrations en couleur n'apparaissent pas à l'écran.

Informations sur les interfaces de programmation

Les informations sur les interfaces de programmation ont pour objectif de vous aider à créer des logiciels d'application à l'aide de ce programme.

Les interfaces de programmation génériques vous permettent de créer des logiciels d'application qui obtiennent les services des outils de ce programme.

Cependant, ces informations peuvent également contenir des informations sur le diagnostic, la modification et le réglage. Ces informations vous permettent d'exécuter le débogage de votre logiciel d'application.

Avertissement : N'utilisez pas ces informations sur le diagnostic, la modification et le réglage comme interface de programmation car elles sont susceptibles de changer.

Marques et marques de service

Les termes qui suivent sont des marques d'International Business Machines Corporation aux Etats-Unis et/ou dans certains autres pays :

i5/OS
IBM
le logo IBM
AIX
AIX 5L
CICS
CrossWorlds
DB2
DB2 Universal Database
Domino
HelpNow
IMS
Informix
iSeries
Lotus

Lotus Notes
MQIntegrator
MQSeries
MVS
Notes
OS/400
Passport Advantage
pSeries
Redbooks
SupportPac
WebSphere
z/OS

Java et toutes les marques incluant Java sont des marques de Sun Microsystems, Inc. aux Etats-Unis et/ou dans certains autres pays.

Microsoft, Windows, Windows NT et le logo Windows sont des marques de Microsoft Corporation aux Etats-Unis et/ou dans certains autres pays.

Intel, le logo Intel, Intel Inside, le logo Intel Inside, Intel Centrino, le logo Intel Centrino, Celeron, Intel Xeon, Intel SpeedStep, Itanium et Pentium sont des marques ou marques déposées de Intel Corporation ou de ses filiales, aux Etats-Unis et dans d'autres pays.

UNIX est une marque enregistrée de The Open Goup aux Etats-Unis et/ou dans certains autres pays.

Linux est une marque de Linus Torvalds aux Etats-Unis et/ou dans certains autres pays.

Les autres noms de sociétés, de produits et de services peuvent appartenir à des tiers.

Ce produit inclut un logiciel développé par Eclipse Project (http://www.eclipse.org/).

WebSphere Business Integration Adapters, Version 6.0

WebSphere Business Integration Adapter Framework V2.4.0