IBM WebSphere Business Integration Adapters

Adapter for HTTP - Guide de l'utilisateur

Version 1.3.0

IBM WebSphere Business Integration Adapters

Adapter for HTTP - Guide de l'utilisateur

Version 1.3.0

Consigne

Avant d'utiliser le présent document et le produit associé, prenez connaissance des informations générales figurant dans la section «Informations légales», à la page 157.

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 SANS AUCUNE GARANTIE EXPLICITE OU IMPLICITE. IBM DECLINE NOTAMMENT TOUTE RESPONSABILITE RELATIVE A CES INFORMATIONS EN CAS DE CONTREFACON AINSI QU'EN CAS DE DEFAUT D'APTITUDE A L'EXECUTION D'UN TRAVAIL DONNE.

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

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

Vous pouvez également consulter les serveurs Internet suivants :

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

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

Table des matières

Avis aux lecteurs canadiens

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

Illustrations

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

Terminologie

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

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

Claviers

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

OS/2 et Windows - Paramètres canadiens

Au Canada, on utilise :

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

Nomenclature

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

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

Brevets

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

Assistance téléphonique

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

A propos de ce document

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

Contenu du document

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

Limites du document

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

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

Public visé

Ce document s'adresse aux clients, consultants et développeurs IBM WebSphere ainsi qu'à tous ceux qui implémentent WebSphere Business Integration Adapter for HTTP.

Prérequis pour ce document

Ce document comprend des prérequis variés. La plupart d'entre eux sont des références à des sites Web contenant des informations ou des ressources relatives au protocole http. Vous devez également être familiarisé avec l'implémentation du système d'intégration métier WebSphere. Pour commencer, vous pouvez lire le document *Technical Introduction to IBM WebSphere InterChange Server*, qui propose des références croisées à des informations plus détaillées.

Documents connexes

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

Vous pouvez télécharger la documentation associée sur les sites suivants :

- Pour obtenir des informations générales sur les adaptateurs, pour apprendre à les utiliser avec des courtiers de messages WebSphere (WebSphere MQ Integrator, WebSphere MQ Integrator Broker, WebSphere Business Integration Message Broker) et avec WebSphere Application Server, voir le Centre de documentation IBM WebSphere Business Integration Adapters à l'adresse : http://www.ibm.com/websphere/integration/wbiadapters/infocenter
- Pour utiliser les adaptateurs avec WebSphere InterChange Server, voir les Centres de documentation IBM WebSphere InterChange Server : http://www.ibm.com/websphere/integration/wicserver/infocenter http://www.ibm.com/websphere/integration/wbicollaborations/infocenter
- Pour plus d'informations sur les courtiers de messages WebSphere : http://www.ibm.com/software/integration/mqfamily/library/manualsa/
- Pour plus d'informations sur WebSphere Application Server : http://www.ibm.com/software/webservers/appserv/library.html

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

Conventions typographiques

Ce document utilise les conventions suivantes :

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

Nouveautés de la présente édition

Nouveautés de l'édition 1.3

Cette édition comporte les améliorations suivantes :

- Prise en charge étendue de méthodes HTTP, comprenant GET et POST, et traitement des entêtes de demandes hôte.
- Ajout de la prise en charge des interrogations ″ping″ sur les programmes d'écoute. Voir «Pingability», à la page 51.
- Ajout de la prise en charge du traitement des demandes vides et des réponses vides. Voir Chapitre 4, «Connecteur HTTP», à la page 47.
- Attributs de méta-objets supplémentaires. Pour plus d'informations, voir «HTTP Protocol Config MO pour le traitement des événements», à la page 25 et «HTTP Protocol Config MO pour le traitement des demandes», à la page 36.
- Ajout de la prise en charge d'IBM WebSphere Business Integration Adapter Framework V2.6.0. Pour plus d'informations, reportez-vous au guide Migrating Adapters to Adapter Framework, Version 2.6.
- Ajout de la prise en charge des plateformes suivantes :
  - AIX 5.3
  - Red Hat 3.0
  - Solaris 9
  - SUSE 8.1
  - Windows 2003
- Ajout de la prise en charge du JRE/JDK 1.4.2.
- Ajout de la prise en charge de la gestion de l'adaptateur HTTP par IBM Tivoli License Manager (ITLM).

Nouveautés de l'édition 1.1

Cette édition comporte les améliorations suivantes :

- Si vous n'avez pas indiqué de valeur pour java.protocol.handler.pkgs, le connecteur utilise la valeur par défaut lors de l'initialisation. Pour plus d'informations, voir «JSSE», à la page 64.
- Le programme d'écoute de protocole HTTP prend en charge les demandes comprenant n'importe quelle valeur d'en-tête Accept ; si nécessaire, la validation de l'en-tête peut-être déléguée à la collaboration.
- La valeur minimale a changé pour la propriété WorkerThreadCount spécifique au connecteur. Pour plus d'informations, voir «WorkerThreadCount», à la page 70.
- Quand une réponse n'est pas complétée par une collaboration en cas de traitement des événements synchrones par des programmes d'écoute HTTP(S), la partie ContentType de l'entête HTTP Content-Type de la réponse sera définie sur le ContentType de la demande.

A partir de la version 1.1.x, l'adaptateur n'étant pas pris en charge sous Solaris 7, les références à cette plateforme ont été supprimées de ce guide.

Chapitre 1. Présentation de l'adaptateur

Le connecteur est un composant d'exécution de WebSphere Business Integration Adapter for HTTP. Le connecteur permet aux entreprises d'agréger, publier et consommer des messages HTTP(S) pour elles-même ou pour leurs partenaires d'échange. Le connecteur et les autres composants décrits dans ce document fournissent les fonctionnalités requises pour l'échanges d'informations d'objet métier dans le corps de messages transmis par les protocoles HTTP et HTTPS.

Ce chapitre décrit la portée, les composants, les outils de conception et l'architecture employés pour l'implémentation de WebSphere Business Integration Adapter for HTTP. Il propose également une présentation des tâches nécessaires à l'installation et à la configuration des composants HTTP décrits dans ce document. Pour plus d'informations sur l'installation et la configuration des composants, voir «Liste de contrôle d'installation, de configuration et de conception», à la page 8.

Remarque : L'adaptateur pour HTTP implémente l'API Adapter Framework standard. C'est pourquoi l'adaptateur peut fonctionner avec n'importe quel courtier d'intégration pris en charge par la structure. Cependant, les fonctionnalités fournies par l'adaptateur ont été spécialement conçues pour prendre en charge le courtier d'intégration IBM WebSphere InterChange Server (ICS).

Environnement d'Adapter for HTTP

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

Exigences matérielles et logicielles

Pour connaître les exigences matérielles et logicielles de cet adaptateur, voir IBM WebSphere Adapters and IBM WebSphere Business Integration Adapters: Hardware and Software Requirements. Sélectionnez l'adaptateur dans la liste d'adaptateurs WebSphere.

Normes et API

Un certain nombre de normes et de technologies donne accès à leurs fonctionnalités sur réseau.

Les normes utilisées par l'adaptateur sont les suivantes :

- HTTP 1.0

Les API utilisées par cet adaptateur sont les suivantes :

- IBM JSSE 1.0.3

Selon votre configuration, il peut être nécessaire installer des logiciels supplémentaires. Ces points sont traités ci-après.

SSL

Si vous envisagez d'utiliser SSL, vous devez recourir à un logiciel tiers de gestion des magasins de clés, des certificats et de la génération de clés. Aucun outil n'est fourni pour configurer magasins de clés, certificats et génération de clés. Cependant, vous pouvez choisir d'utiliser keytool (fourni avec IBM JRE) afin de créer des certificats auto-signés et de créer des magasins de clés. Pour plus d'informations, voir «SSL», à la page 64.

Données dépendant des paramètres nationaux

Le connecteur a été internationalisé de sorte qu'il puisse prendre en charge les jeux de caractères à deux octets. 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 WebSphere Business Integration sont écrits en Java. Par conséquent, lorsque des données sont transférées entre la plupart des composants d'intégration, la conversion des caractères est inutile.

Remarque : Le connecteur n'a pas été internationalisé. Ceci signifie que les messages de trace et d'historique ne sont pas traduits.

Connecteur HTTP

Cette section aborde l'internationalisation et le connecteur.

Notification d'événements : Le connecteur utilise des programmes d'écoute de protocole connectables pour la notification des événements. Les programmes d'écoute de protocole extraient le message du transport et appellent le gestionnaire de données indiqué dans les méta-données de message. Pour plus d'informations sur le traitement des programmes d'écoute, voir «HTTP et HTTPS traitement du programme d'écoute de protocole», à la page 51.

Traitement des demandes : Le connecteur utilise une structure de gestionnaire de protocole HTTP-HTTPS connectable pour le traitement des demandes. Les gestionnaires de protocole appellent le gestionnaire de données. Pour plus d'informations, voir «HTTP-HTTPS traitement du gestionnaire de protocole», à la page 60.

Gestionnaire de données

Vous pouvez configurer l'adaptateur HTTP pour utiliser n'importe quel gestionnaire de données. Pour plus d'informations sur la configuration des gestionnaires de données, voir «Configuration du gestionnaire de données», à la page 9.

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 au niveau de l'objet métier d'une définition d'objet métier.
- **BO (objet métier)** Ensemble d'attributs représentant une entité de l'entreprise (telle que les clients) et une action sur les données (telle que l'opération de création ou de mise à jour). Les composants du système IBM WebSphere utilisent des objets métier pour échanger des informations et déclencher des actions.
- **Content-Type** En-tête de protocole HTTP comprenant la combinaison *type/sous-type* et des paramètres facultatifs. Par exemple, dans la valeur Content-Type `text/xml;charset=ISO-8859-1`, `text/xml` est la paire type/sous-type et `charset=ISO-8859-1`, le paramètre Charset facultatif.
- **ContentType** fait référence à la partie *type/sous-type* de la valeur d'en-tête Content-Type uniquement. Par exemple, dans la valeur Content-Type `text/xml;charset=ISO-8859-1`, `text/xml` est désigné sous le nom de ContentType dans ce document.
- **MO_DataHandler_Default** Méta-objet de gestionnaire de données utilisé par l'agent connecteur pour déterminer quel gestionnaire de données instancier. Ceci est indiqué dans la propriété de configuration DataHandlerMetaObjectName du connecteur.
- **Protocol Config MO** Pendant le traitement des demandes, les gestionnaires de protocole HTTP-HTTPS utilisent un Protocol Config MO pour déterminer la destination. Si lors du traitement des événements vous exposez des collaborations, le connecteur utilise le Protocol Config MO pour transporter des informations d'entête de message depuis le programme d'écoute de protocole HTTP ou HTTPS vers la collaboration.
- **Objet métier de niveau supérieur (TLO)** Un objet métier de niveau supérieur contient un objet métier Request, un objet métier Response (facultatif) et un ou plusieurs objets métier Fault (facultatif). Un TLO est utilisé par le connecteur pour le traitement des événements et le traitement des demandes.

Composants de connecteur pour HTTP

La figure 1 illustre le connecteur pour HTTP, comprenant ses structures de gestionnaires et de programmes d'écoute de protocole.

Remarque : Adapter for HTTP offre une licence d'utilisation limitée du gestionnaire de données XML. Cependant, l'adaptateur ne nécessite pas le gestionnaire de données XML pour fonctionner.

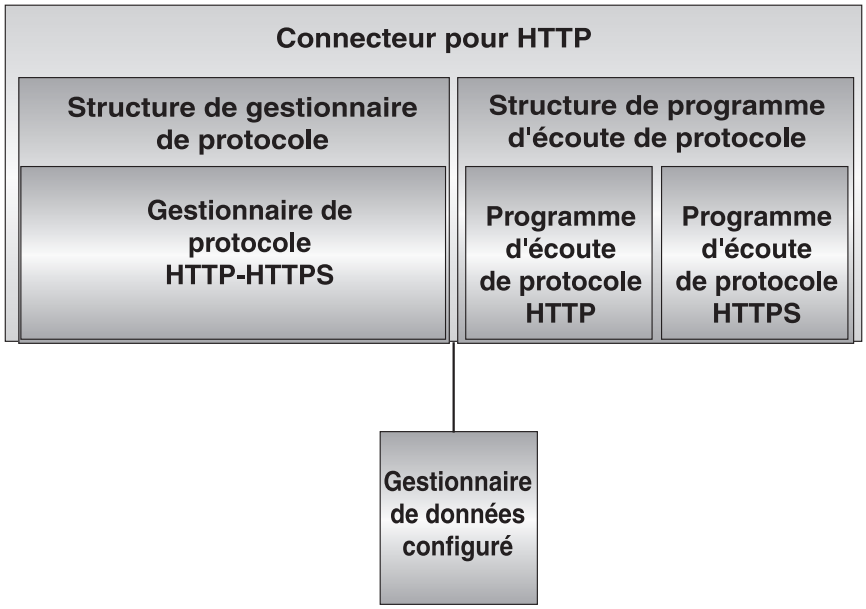


Figure 1. Connecteur pour HTTP

Les composants suivants interagissent pour activer les échanges de données sur Internet :

- Connecteur HTTP, comprenant le gestionnaire de données et les programmes d'écoute et gestionnaires de protocole configurés
- Collaborations HTTP
- Objets métier et messages HTTP(S)
- WebSphere Business Integration InterChange Server

Connecteur pour HTTP

Lors du traitement des demandes, le connecteur répond aux appels de service de collaboration en convertissant des objets métier en messages de demande et en les acheminant aux destinations indiquées. Le cas échéant (lors du traitement de demandes synchrones), le connecteur convertit les messages de réponse en objets métier de réponse et les renvoie à la collaboration.

Lors du traitement des événements, le connecteur traite les messages de demande émanant de clients en les convertissant en objets métier de demande et en les transmettant aux collaborations pour traitement. Le cas échéant, le connecteur reçoit des objets métier de réponse de la collaboration, qui sont convertis en messages de réponse et retournés aux clients.

Pour plus d'informations, voir Chapitre 4, «Connecteur HTTP», à la page 47

Remarque : Dans ce document, toute mention d'un connecteur est une référence implicite au connecteur HTTP sauf indication explicite.

Programmes d'écoute et gestionnaires de protocole

Le connecteur inclut les programme d'écoute et le gestionnaire de protocole suivants :

- programme d'écoute de protocole HTTP
- programme d'écoute de protocole HTTPS
- gestionnaire de protocole HTTP-HTTPS

Les programmes d'écoute de protocole détectent des événements de clients internes ou externes dans le format HTTP ou HTTPS. Ils notifient le connecteur d'événements nécessitant le traitement par une collaboration. Les programmes d'écoute de protocole lisent alors l'ASI de niveau objet métier et de niveau attribut, des propriétés de connecteur et des règles de transformation intégrées dans les objets de configuration de protocole pour déterminer la collaboration, le gestionnaire de données, le mode de traitement (synchrone/asynchrone) et les aspects spécifiques au transport de la transaction. «Programmes d'écoute de protocole», à la page 50 décrit en détail le traitement des programmes d'écoute de protocole.

Les **gestionnaires de protocole** appellent les services HTTP aux formats HTTP ou HTTPS au nom d'une collaboration. Le gestionnaire de protocole HTTP(S) lit l'ASI TLO et les règles de transformation imbriquées dans les objets de configuration de protocole pour déterminer la façon de traiter la demande (synchrone ou asynchrone), quel gestionnaire de données à utiliser pour convertir les messages en objets métier et inversement, et pour déterminer la destination (de l'attribut Destination du Protocol Config MO de l'objet métier de la demande). Pour les transactions synchrones, le gestionnaire de protocole traite les messages de réponse, les convertissant en objets métier de réponse et en les renvoyant à la collaboration.

Pour plus d'informations sur les gestionnaires de protocole, voir «Gestion de protocole», à la page 59.

Gestionnaire de données

Vous pouvez configurer l'adaptateur HTTP pour utiliser n'importe quel gestionnaire de données. A des fins d'illustration, ce document fait souvent référence à un type mime `text/xml` et à un gestionnaire de données XML.

Le gestionnaire de données configuré convertit les objets métier en messages et inversement. Pour plus d'informations, voir la documentation du gestionnaire de données que vous utilisez avec l'adaptateur HTTP.

Agents ODA (Object discovery agent)

Si vous utilisez un gestionnaire de données pour lequel il existe un agent ODA, vous pouvez utiliser cet ODA afin de générer des objets métier. Par exemple, si vous devez utiliser l'encodage XML et configurez l'adaptateur avec le gestionnaire de données XML, vous pouvez utiliser l'ODA XML afin de créer et de modifier des objets métier.

Déploiement du connecteur

Il existe deux moyens de déployer le connecteur HTTP :

- Derrière le pare-feu en tant que solution intranet (voir la figure 2) dans une entreprise dont les processus métier communiquent par HTTP ou HTTPS.

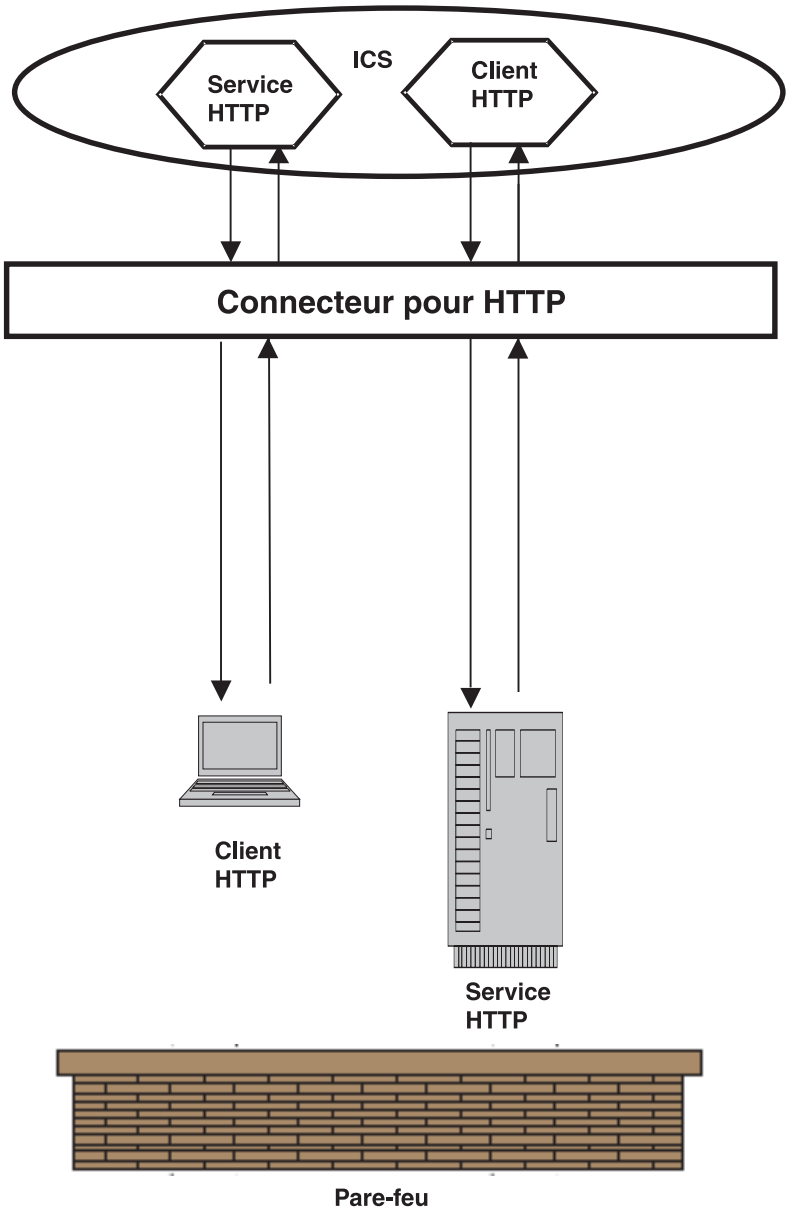


Figure 2. Adaptateur HTTP en tant que solution intranet

- Derrière le pare-feu avec un serveur frontal ou passerelle pour traiter, filtrer et gérer les communications externes à l'entreprise.

Remarque : Le connecteur HTTP n'inclut pas de passerelle ou de serveur frontal pour la gestion des messages entrants et sortants de clients externes. Vous devez configurer et déployer votre propre passerelle. ***Le connecteur doit être déployé dans l'entreprise uniquement et non pas dans la zone démilitarisée ou à l'extérieur du pare-feu.***

Architecture du connecteur pour HTTP

Pour illustrer l'architecture des composants à un niveau élevé, cette section décrit deux flux de données. La figure 3 illustre les deux scénarios. Ces deux scénarios sont décrits ci-après.

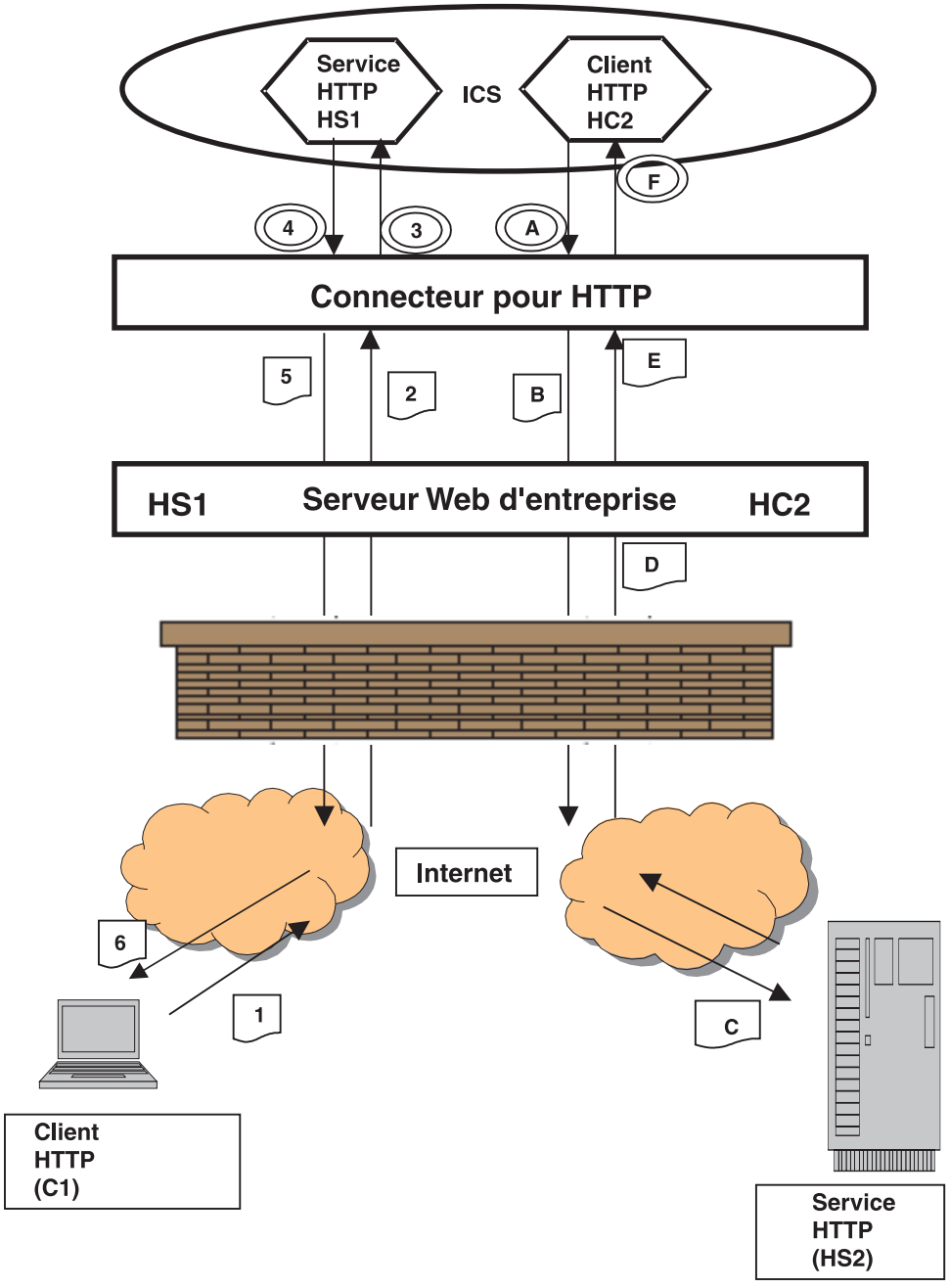


Figure 3. Déroulement d'un message HTTP

Le **traitement des demandes** illustre la séquence d'événements qui se produit lorsqu'une collaboration soumet une demande d'appel de service au connecteur. Dans ce scénario, la collaboration joue le rôle de client, envoyant une demande à un serveur.

A La collaboration envoie une demande d'appel de service au connecteur, qui appelle un gestionnaire de données pour convertir l'objet métier en message de demande.

B Le connecteur appelle l'URL du serveur Web d'entreprise en envoyant le message de demande.

C Le serveur Web d'entreprise appelle l'URL du serveur HTTP (HS2).

D Le serveur HTTP HS2 traite la demande et renvoie la réponse. La réponse est retournée lors de la même connexion.

E Le serveur Web d'entreprise retourne le message de réponse à l'adaptateur.

F Le connecteur reçoit le message de réponse (ou d'erreur), appelle le gestionnaire de données pour convertir le message en objet métier et le retourne à la collaboration.

Le **traitement des événements** illustre la séquence d'événements qui se produit lorsqu'une collaboration est appelée par un client HTTP. Dans ce scénario, la collaboration joue le rôle de serveur, acceptant une demande émanant d'un client (externe ou interne) et répondant en conséquence.

1 Le client HTTP (C1) envoie un message de demande à la destination — c'est-à-dire la collaboration.

2 Si le client HTTP est externe, la passerelle reçoit et achemine le message au connecteur.

3 Le connecteur envoie le message au gestionnaire de données pour conversion en objet métier. Le connecteur appelle la collaboration.

4 La collaboration retourne un objet métier de réponse (ou d'erreur).

5 Le connecteur appelle le gestionnaire de données pour convertir l'objet métier de réponse (ou d'erreur) en message de réponse. Le connecteur retourne la réponse à la passerelle.

6 Si le client est externe, la passerelle achemine le message de réponse au client HTTP (C1).

Liste de contrôle d'installation, de configuration et de conception

Cette section récapitule les tâches à effectuer pour installer, configurer et concevoir votre solution HTTP. Chaque section décrit brièvement les tâches et fournit des liens vers des sections de ce document (et des références à des documents connexes) décrivant la réalisation de la tâche ou fournissant des informations de fond.

Installation de l'adaptateur

Voir Chapitre 2, «Installation et démarrage», à la page 11 pour une description des éléments et des emplacements d'installation.

Configuration des propriétés de connecteur

Les connecteurs ont deux types de propriétés de configuration : les propriétés standard et les propriétés spécifiques. Certaines de ces propriétés possèdent des valeurs par défaut que vous n'avez pas à modifier. Vous pouvez définir les valeurs de certaines de ces propriétés avant d'exécuter le connecteur. Pour plus d'informations, voir Chapitre 4, «Connecteur HTTP», à la page 47.

Configuration des gestionnaires et des programmes d'écoute de protocole

La configuration de gestionnaires et de programmes d'écoute de protocole a lieu quand vous affectez des valeurs à des propriétés de configuration de connecteur régissant le comportement de ces composants. Pour plus d'informations, voir Chapitre 4, «Connecteur HTTP», à la page 47.

Création ou modification d'objets métier

Selon le(s) gestionnaire(s) de données que vous utilisez avec le connecteur HTTP, il se peut qu'un ODA soit disponible. L'agent ODA automatise le processus de création et de modification d'objets métier. Autrement, vous pouvez créer ou modifier manuellement des objets métier à l'aide de Business Object Designer. Pour plus d'informations, voir la documentation du gestionnaire de données que vous utilisez et le document *Business Object Development Guide*.

Configuration du gestionnaire de données

La configuration du (ou des) méta-objet(s) de gestionnaire de données a lieu après l'installation des fichiers de produit mais avant le démarrage. Commencez par configurer le gestionnaire de données en indiquant une propriété de configuration spécifique de connecteur : DataHandlerMetaObjectName. Indiquez le nom du méta-objet de niveau supérieur (MO_DataHandler_Default) que le gestionnaire de données utilise pour récupérer des propriétés de configuration. Suivez alors les étapes de configuration supplémentaires requises par le gestionnaire de données utilisé.Le cas échéant, vous pouvez indiquer un gestionnaire de données à l'aide de l'attribut TLO MimeType. Pour plus d'informations, voir tableau 5, à la page 22.

Pour plus d'informations sur la configuration du gestionnaire de données, voir «Propriétés de configuration spécifiques au connecteur», à la page 67

Chapitre 2. Installation et démarrage

Ce chapitre décrit l'installation de composants pour l'implémentation du connecteur pour HTTP. Pour plus d'informations concernant l'installation d'un système ICS en général, consultez le document *System Installation Guide* approprié à votre plateforme.

Présentation des tâches d'installation

Pour plus d'informations sur la compatibilité des courtiers, de la structure d'adaptateurs, les exigences logicielles, les dépendances, les normes et les API, voir «Environnement d'Adapter for HTTP», à la page 1.

Pour installer le connecteur pour HTTP, effectuez les tâches suivantes :

Installer ICS

Cette tâche, comprenant l'installation du système et le démarrage d'ICS, est décrit dans le guide d'installation système. Vous devez installer ICS version 4.2.2 ou ultérieure.

Pour charger des fichiers dans le référentiel, consultez le document *Implementation Guide for WebSphere InterChange Server*.

Installation du connecteur et des fichiers associés

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

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

Structure de fichiers installée

Les tableaux de cette section représentent la structure de fichiers installée.

Structure de fichiers du connecteur Windows

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

L'utilitaire installe le connecteur et ajoute dans le menu Démarrer un raccourci pour l'agent du connecteur.

Le tableau 1 décrit la structure de fichiers utilisée sous Windows par le connecteur, et indique les fichiers qui sont automatiquement installés lorsque vous utilisez le programme d'installation.

Tableau 1. Structure de fichiers installée sous Windows pour l'adaptateur

Sous-répertoire de *ProductDir*	Description
connectors\HTTP\BIA_HTTP.jar	Fichier jar WebSphere Business Integration Adapter
connectors\HTTP\start_HTTP.bat	Fichier de démarrage du connecteur
bin\Data\App\HTTPConnectorTemplate	Modèle de connecteur HTTP
connectors\HTTP\dependencies\mail.jar	API JavaMail d'IBM
connectors\HTTP\dependencies\IBMReadme.txt	Licence
connectors\HTTP\samples\WebSphereICS\HTTPSample.jar	Fichier du référentiel pour les exemples
connectors\HTTP\samples\WebSphereICS\CLIENT_SYNCH_TLO_OrderStatus.bo	Exemple d'objet métier pour connecteur de test (synchrone)
connectors\HTTP\samples\WebSphereICS\CLIENT_ASYNCH_TLO_Order.bo	Exemple d'objet métier pour connecteur de test (asynchrone)
connectors\HTTP\samples\WebSphereICS\SERVICE_SYNCH_TLO_OrderStatus.bo	Exemple d'objet métier (synchrone)
connectors\messages\HTTPConnector.txt	Fichier de messages du connecteur

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

Structure de fichiers de connecteur UNIX

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

Le tableau 2 décrit la structure de fichiers utilisée sous UNIX par le connecteur, et indique les fichiers qui sont automatiquement installés lorsque vous utilisez le programme d'installation.

Tableau 2. Structure de fichiers installée sous UNIX pour l'adaptateur

Sous-répertoire de *ProductDir*	Description
connectors/HTTP/BIA_HTTP.jar	Fichier jar WebSphere Business Integration Adapter
connectors/HTTP/start_HTTP.sh	Fichier de démarrage du connecteur
bin/Data/App/HTTPConnectorTemplate	Modèle de connecteur HTTP
connectors/HTTP/dependencies/mail.jar	API JavaMail d'IBM
connectors/HTTP/dependencies/IBMReadme.txt	Licence
connectors/HTTP/samples/WebSphereICS/HTTPSample.jar	Fichier du référentiel pour les exemples
connectors/HTTP/samples/WebSphereICS/CLIENT_SYNCH_TLO_OrderStatus.bo	Exemple d'objet métier pour connecteur de test (synchrone)
connectors/HTTP/samples/WebSphereICS/CLIENT_ASYNCH_TLO_Order.bo	Exemple d'objet métier pour connecteur de test (asynchrone)
connectors/HTTP/samples/WebSphereICS/SERVICE_SYNCH_TLO_OrderStatus.bo	Exemple d'objet métier (synchrone)
connectors/messages/HTTPConnector.txt	Fichier de messages du connecteur

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

Présentation des tâches de configuration

Après l'installation et avant le démarrage, vous devez configurer des composants comme suit :

Configurez le connecteur

Cette tâche comprend l'installation et la configuration du connecteur. Voir «Configuration du connecteur», à la page 66.

Configuration d'objets métier

La procédure de configuration des objets métier dépend de la façon dont vous choisissez d'implémenter la suite logicielle :

- **Traitement des demandes** Vous devez créer les objets métier correspondant aux éléments suivants :
  - Messages de demande sortants
  - Toutes les réponses possibles, y compris les erreurs

  Pour plus d'informations, voir Chapitre 3, «Exigences des objets métier», à la page 19.
- **Traitement des événements** Vous utilisez les objets métier TLO. Pour plus d'informations, voir Chapitre 3, «Exigences des objets métier», à la page 19.

Vous pouvez créer des objets métier manuellement à l'aide de Business Object Designer ou, selon votre gestionnaire de données, un ODA qui automatise le processus de génération d'objets métier. Pour plus d'informations, voir la documentation du gestionnaire de données.

Configuration du gestionnaire de données

Configurez le gestionnaire de données en indiquant une propriété de configuration spécifique de connecteur : DataHandlerMetaObjectName. Indiquez le nom du méta-objet de niveau supérieur (MO_DataHandler_Default) que le gestionnaire de données utilise pour récupérer des propriétés de configuration. Suivez alors les étapes de configuration supplémentaires requises par le gestionnaire de données utilisé.

Le cas échéant, vous pouvez indiquer un gestionnaire de données à l'aide de l'attribut TLO MimeType. Pour plus d'informations, voir le tableau 5, à la page 22.

Pour plus d'informations sur la configuration du gestionnaire de données, voir «Propriétés de configuration spécifiques au connecteur», à la page 67

Configuration de collaborations

Pour activer des collaborations en vue du traitement des demandes ou des événements, consultez la documentation suivante :

- *IBM WebSphere InterChange Server Collaboration Development Guide*
- *IBM WebSphere InterChange Server Map Development Guide*

Exécution d'instances multiples de l'adaptateur

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».
3. Créez un raccourci pour le script de démarrage (Windows uniquement).
4. Copiez le texte du raccourci du connecteur et modifiez le nom du connecteur initial (dans la ligne de commande) de sorte qu'il corresponde au nom de la nouvelle instance du connecteur.

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

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

Démarrage du connecteur

Important : Comme indiqué précédemment dans ce chapitre, le connecteur, les objets métier, les méta-objets du gestionnaire de données et les collaborations doivent être configurés après l'installation et après le démarrage du connecteur pour pouvoir fonctionner correctement. La section «Présentation des tâches de configuration», à la page 12 fournit un récapitulatif de ces tâches. En outre, la collecte de connecteurs ne doit pas être désactivée (cette fonction est activée par défaut).

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

ProductDir\connectors*connName*

où *connName* identifie le connecteur.

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

Le nom du script de démarrage dépend de la plateforme du système d'exploitation, comme le montre le tableau 3.

Tableau 3. Script de démarrage pour un connecteur

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

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

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

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

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

  Sélectionnez **Programmes>IBM WebSphere Business Integration Adapters>Adapters>Connectors**. Par défaut, le nom du programme est "IBM WebSphere Business Integration Adapters". Cependant, vous pouvez le personnaliser. Vous pouvez également créer sur le bureau un raccourci vers le connecteur.

- A partir de la ligne de commande :
  - Sur les systèmes Windows :

    ```
    start_connName connName brokerName [-cconfigFile]
    ```

  - Sur les systèmes UNIX :

    ```
    connector_manager -start connName
    brokerName [-cconfigFile ]
    ```

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

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

  Vous pouvez charger, activer, désactiver, interrompre, arrêter ou supprimer un connecteur à l'aide de cet outil.
- A partir de System Manager (pour tous les courtiers) :

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

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

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

Arrêt du connecteur

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

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

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

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

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

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

Chapitre 3. Exigences des objets métier

Ce chapitre décrit la structure, les exigences et les attributs des objets métier de connecteur.

Méta-données d'objet métier

Le connecteur pour HTTP est contrôlé par des méta-données. Dans les objets métier, les méta-données sont des données sur l'application stockées dans une définition d'objet métier et permettant au connecteur d'interagir avec cette application. Un connecteur contrôlé par des méta-données traite chaque objet métier qu'il prend en charge d'après les méta-données codées dans la définition de l'objet métier plutôt que d'après les instructions codées en dur dans le connecteur.

Les méta-données de l'objet métier incluent la structure d'un objet métier, les paramètres de ses propriétés d'attribut et le contenu de ses informations spécifiques à l'application. Dans la mesure où le connecteur est contrôlé par les méta-données, il peut traiter de nouveaux objets métier ou des objets modifiés sans avoir à modifier son code.Le connecteur établit des suppositions concernant la structure de ses objets métier, la cardinalité des objets, le format du texte spécifique à l'application et la représentation de l'objet métier dans la base de données. Par conséquent, lorsque vous créez ou modifiez un objet métier pour HTTP, vos modifications doivent être conformes aux règles que le connecteur est supposé suivre, sinon il ne peut pas traiter correctement les objets métier nouveaux ou modifiés.

Structure des objets métier du connecteur

Le connecteur traite les objets métier de niveau supérieur (Top-level objects, TLO), qui sont employés pour le traitement des demandes et le traitement des événements. Les TLO contiennent un objet métier de demande et, le cas, des objets métier de réponse et d'erreur. Ces objets enfants possèdent des données de contenu et, le cas échéant, des MO Protocol Config. Il s'agit également d'objets spécifiques gestionnaires de données ; par exemple, si vous utilisez le gestionnaire de données XML, l'enfant de demande serait un objet métier intelligible pour le gestionnaire de données XML. Les TLO, les objets de demande, de réponse et d'erreur ainsi que les informations, les attributs et les exigences spécifiques à l'application et relatifs au traitement des demandes (par opposition au traitement des événements) sont décrits et illustrés dans les sections ci-après.

TLO de traitement des événements synchrones

Concernant le traitement des événements, les TLO autorisés par le connecteur sont de deux types : les TLO synchrones et les TLO asynchrones. Cette section aborde les TLO de traitement des événements synchrones.

La figure 4 représente la hiérarchie des objets métier pour le traitement des événements synchrones. Les objets de demande et de réponse sont obligatoires, à la différence des objets d'erreur qui sont facultatifs.

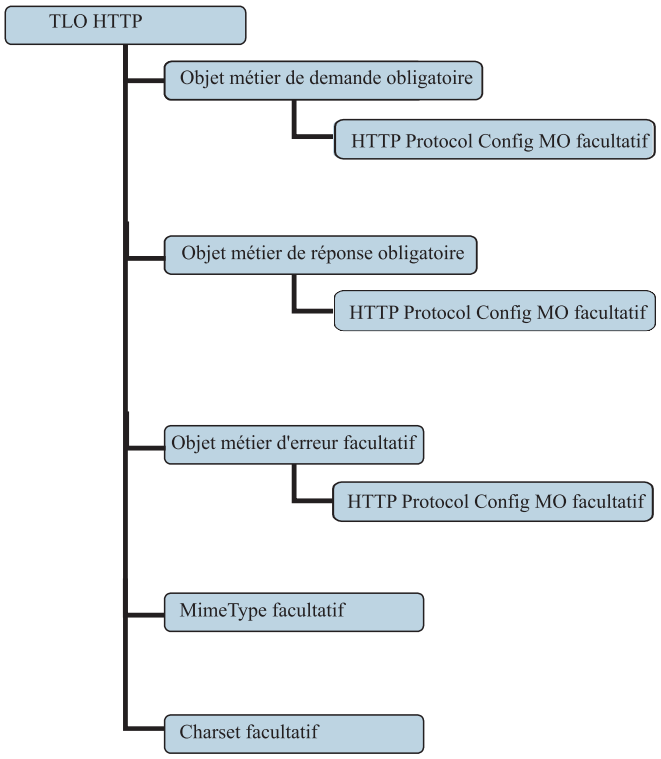


Figure 4. Hiérarchie des objets métier pour le traitement des événements synchrones.

Le TLO contient des informations ASI de niveau objet ainsi que des attributs avec des informations ASI de niveau attribut. Les deux types d'ASI sont décrits ci-après.

ASI de niveau objet pour les TLO de traitement des événements synchrones

L'ASI de niveau objet apporte des informations fondamentales sur la nature d'un TLO et sur les objets qu'il contient. La figure 5 représente des informations ASI de niveau objet pour SERVICE_SYNCH_OrderStatus, TLO exemple pour le traitement des événements synchrones.

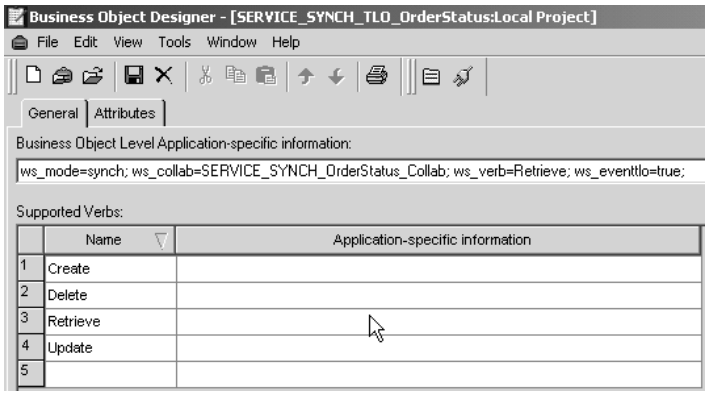


Figure 5. Objet métier de niveau supérieur pour le traitement des événements synchrones

Le tableau 4 ci-après décrit les informations ASI de niveau supérieur pour un TLO de traitement des événements synchrones.

Tableau 4. Informations ASI d'objet TLO de traitement des événements synchrones

ASI de niveau objet	Description
`ws_eventtlo=true`	Si cette propriété ASI est définie sur `true`, le connecteur traite cet objet comme un TLO activé pour le traitement des événements.
`ws_collab=`*`collabname`*	Cet ASI indique au connecteur quelle collaboration appeler. Sa valeur est le nom de la collaboration. Dans l'exemple représenté dans la figure 5, le nom de la collaboration est SERVICE_SYNCH_OrderStatus_Collab)
`ws_verb=`*`instruction`*	Avant de livrer le TLO à la collaboration, le connecteur utilise cet ASI pour définir l'instruction sur le TLO. Dans l'exemple représenté dans la figure 5, l'instruction est Retrieve.
`ws_mode=synch`	Lors de la notification d'événement, le connecteur emploie cette propriété ASI pour déterminer le mode d'appel de la collaboration : synchrone (`synch`) ou asynchrone (`asynch`). Pour le traitement synchrone, cet ASI doit être réglé sur `synch`. La valeur par défaut est `asynch`.

ASI de niveau attribut pour les TLO de traitement des événements synchrones

Chaque TLO de traitement des événements synchrones possède des attributs et des informations ASI de niveau attribut. La figure 6 représente les attributs d'un TLO exemple dénommé SERVICE_SYNCH_OrderStatus. Les informations ASI de niveau attribut sont représentées dans la colonne App Spec Info.

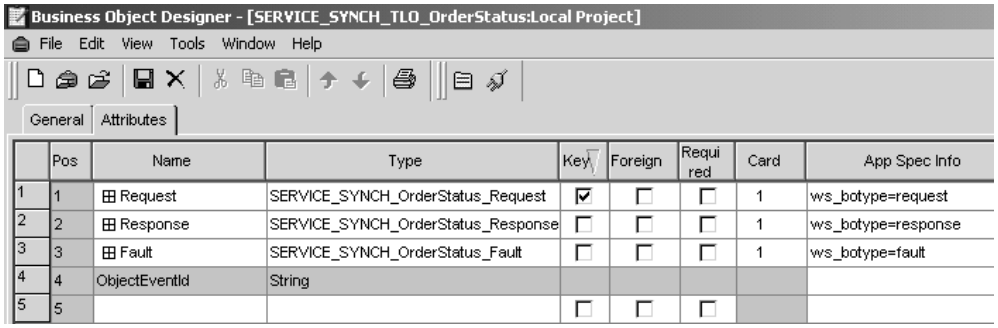


Figure 6. Attributs TLO pour le traitement des événements synchrones.

Le tableau 5 récapitule les informations ASI de niveau attribut pour les attributs de demande, de réponse, d'erreur, MimeType et Charset d'un TLO de traitement des événements asynchrones.

Tableau 5. Informations ASI d'attribut TLO de traitement des événements synchrones

Attribut TLO	ASI de l'attribut	Description
MimeType		Attribut facultatif ; sa valeur est le type mime du gestionnaire de données pour appeler la réponse synchrone.
Charset		Ce paramètre facultatif de type String indique le charset à associer au gestionnaire de données lors de la transformation d'un objet métier sortant vers le message. REMARQUE : la valeur charset indiquée dans cet attribut ne sera pas propagée dans l'entête de protocole Content-Type du message de réponse.
Demande	`ws_botype=request`	Cet attribut correspond à une demande de service HTTP. Le connecteur utilise ses informations ASI pour déterminer si cet attribut TLO est de type BO de demande. Le type d'attribut est déterminé par ces informations et non par le nom d'attribut. S'il existe plusieurs attributs de demande, le connecteur utilise les informations ASI du premier. Cet attribut est obligatoire pour les TLO de traitement des événements synchrones.
Réponse	`ws_botype=response`	Cet attribut correspond à la réponse retournée par un service HTTP. Le connecteur utilise ces informations ASI pour déterminer si cet attribut TLO est de type BO de réponse. Le type d'attribut est déterminé par ces informations et non par le nom d'attribut. S'il existe plusieurs attributs de réponse, le connecteur utilise les informations ASI du premier. Cet attribut est obligatoire pour les TLO de traitement des événements synchrones.

Tableau 5. Informations ASI d'attribut TLO de traitement des événements synchrones (suite)

Attribut TLO	ASI de l'attribut	Description
Fault	`ws_botype=fault` `ws_botype=defaultfault`	Cet attribut, facultatif pour le traitement des événements synchrones, correspond à un message d'erreur retourné par une collaboration quand il n'est pas possible de compléter une réponse avec succès. Le connecteur utilise ces informations ASI et non pas le nom d'attribut pour déterminer si l'attribut est de type BO d'erreur.

Objet métier de demande pour le traitement des événements synchrones

Un objet métier de demande est enfant d'un TLO et obligatoire pour le traitement des événements synchrones. Un objet métier de demande possède des informations ASI de niveau objet. Les informations ASI de niveau objet d'un objet métier de demande pour le traitement des événements synchrones est décrit dans le tableau 6. Vous pouvez indiquer une instruction par défaut pour l'objet métier de demande. Pour cela, saisissez :

```
DefaultVerb=true;
```

dans la zone ASI de l'instruction dans la liste des instructions prises en charge au niveau le plus élevé de l'objet métier de demande. Si les informations ASI de `DefaultVerb` ne sont pas indiquées et que le gestionnaire de données traite un objet métier sans instruction définie, l'objet métier est renvoyé sans instruction.

Tableau 6. Traitement des événements synchrones : ASI de niveau objet pour les objets métier de demande

ASI de niveau objet	Description
`cw_mo_http=`*HTTPCfgMO*	La valeur de cet ASI doit correspondre au nom de l'attribut associé au Protocol Config MO. L'ASI désigne le programme d'écoute de protocole HTTP ou HTTPS. L'ASI et le Protocol Config MO sont facultatifs. Pour plus d'informations, voir «Protocol Config MO», à la page 25. Remarque : Les gestionnaires de données que vous configurez pour les transformations d'objet métier doivent pouvoir lire tout ASI commençant par `cw_mo` en tant que méta-données et non pas élément de données métier à convertir. Le gestionnaire de données XML est capable de détecter des méta-données `cw_mo`, ignorant les attributs qui leur sont associés.

Tableau 6. Traitement des événements synchrones : ASI de niveau objet pour les objets métier de demande (suite)

ASI de niveau objet	Description
ws_tloname=*tloname*	Cet ASI indique le nom du TLO auquel appartient cet objet. Lors du traitement d'événements, le connecteur utilise cet ASI pour déterminer si l'objet métier de demande livré par le gestionnaire de données est enfant du TLO. Dans l'affirmative, le connecteur crée le TLO indiqué, définit l'objet métier de demande comme objet enfant et utilise l'ASI de niveau objet des TLO pour le livrer à la collaboration souscriptrice.

Objet métier de réponse pour le traitement des événements synchrones

Un objet métier de réponse est enfant d'un TLO et obligatoire pour le traitement des événements synchrones. Les informations ASI de niveau objet d'un objet métier de réponse pour le traitement des événements synchrones est décrit dans le tableau 7.

Tableau 7. Traitement des événements synchrones : ASI de niveau objet pour les objets métier de réponse

ASI de niveau objet	Description
cw_mo_http=*HTTPCfgMO*	La valeur de cet ASI doit correspondre au nom de l'attribut associé au Protocol Config MO. L'ASI désigne le programme d'écoute de protocole HTTP ou HTTPS. L'ASI et le Protocol Config MO sont facultatifs. Pour plus d'informations, voir «Protocol Config MO», à la page 25. Remarque : Les gestionnaires de données que vous configurez pour les transformations d'objet métier doivent pouvoir lire tout ASI commençant par cw_mo en tant que méta-données et non pas élément de données métier à convertir. Le gestionnaire de données XML est capable de détecter des méta-données cw_mo, ignorant les attributs qui leur sont associés.

Remarque : Le cas échéant, vous pouvez inclure des informations ASI de niveau objet pour le Protocol Config MO de l'objet métier de réponse.

Objet métier d'erreur pour le traitement des événements synchrones

Un objet métier d'erreur est enfant d'un TLO et obligatoire pour le traitement des événements synchrones. Les informations ASI de niveau objet d'un objet métier d'erreur pour le traitement des événements synchrones est décrit dans le tableau 8.

Tableau 8. Traitement des événements synchrones : ASI de niveau objet pour les objets métier d'erreur

ASI de niveau objet	Description
`cw_mo_http=`*HTTPCfgMO*	La valeur de cet ASI doit correspondre au nom de l'attribut associé au Protocol Config MO. L'ASI désigne le programme d'écoute de protocole HTTP ou HTTPS. L'ASI et le Protocol Config MO sont facultatifs. Pour plus d'informations, voir «Protocol Config MO». Remarque : Les gestionnaires de données que vous configurez pour les transformations d'objet métier doivent pouvoir lire tout ASI commençant par `cw_mo` en tant que méta-données et non pas élément de données métier à convertir. Le gestionnaire de données XML est capable de détecter des méta-données `cw_mo`, ignorant les attributs qui leur sont associés.

Remarque : Le cas échéant, vous pouvez inclure des informations ASI de niveau objet pour le Protocol Config MO de l'objet métier d'erreur.

Protocol Config MO

Le Protocol Config MO est inclus facultativement comme enfant des objets métier de demande, réponse ou erreur dans le cadre du traitement des événements. En général, vous le spécifiez quand vous devez lire (à partir de messages de demande) ou propager (vers des messages de réponse ou d'erreur) les propriétés d'entête de protocole et les propriétés personnalisées. Comme nous l'avons vu précédemment, l'objet métier de demande déclare facultativement le nom du Protocol Config MO comme informations ASI de niveau objet métier :

`cw_mo_http=`*`HTTPProtocolListenerConfigMOAttribute`*

HTTP Protocol Config MO pour le traitement des événements

Lors du traitement des événements, le connecteur utilise des programmes d'écoute de protocole (HTTP ou HTTPS) pour récupérer des événements auprès du transport. Ces événements sont des messages émanant de clients internes ou externes demandant un service auprès de collaborations. Chaque transport possède ses propres exigences d'en-tête. Le connecteur utilise le Protocol Config MO pour transporter des informations d'entête depuis le programme d'écoute de protocole vers la collaboration. Les attributs de Protocol Config MO correspondent à des entêtes dans le message entrant. Le connecteur définit la valeur de ces attributs dans l'objet métier à l'aide du contenu de message entrant.

Pour le protocole HTTP(S), les attributs de Protocol Config MO sont les suivants :

Tableau 9. Attributs de Protocol Config MO HTTP/HTTPS pour le traitement des événements

Attribut	Obligatoire	Type	Description
Destination	Non	String	Utilisé pour la notification d'événements afin de propage Request-URI à partir de la ligne de demande.

Tableau 9. Attributs de Protocol Config MO HTTP/HTTPS pour le traitement des événements (suite)

Attribut	Obligatoire	Type	Description
Content-Type	Non	String	La valeur de cet attribut définit l'entête Content-Type du message sortant (incluant le ContentType du message ainsi que des paramètres --le charset-- pour le message sortant ; le nombre de paramètres peut être nul ou non). La syntaxe est la même que pour l'entête Content-Type dans le protocole HTTP ; par exemple : `text/xml; charset=ISO-8859-4`. Si aucun attribut Content-Type n'est défini, le connecteur utilise le ContentType de la demande comme ContentType du message de réponse/d'erreur.
Method	Non	String	Sera complété avec la méthode HTTP de demande d'événement dans la notification d'événement.
UserDefinedProperties	Non	Objet métier	Cet attribut comprend l'objet métier des propriétés de protocole définies par l'utilisateur.
Un ou plusieurs entêtes HTTP	Non	String	Cet attribut permet au gestionnaire de transmettre ou de récupérer la valeur de l'entête HTTP indiqué.
Authorization_UserId	Non	`String`	Cet attribut correspond à l'ID utilisateur de l'authentification HTTP de base.
Authorization_Password	Non	`String`	Cet attribut correspond au mot de passe de l'authentification HTTP de base.

Ces attributs sont décrits dans :

- «Propriétés définies par l'utilisateur pour le traitement des événements»
- «Propagation des identifiants HTTP pour le traitement des événements», à la page 27

Pour plus d'informations sur les programmes d'écoute de protocole, voir «Programmes d'écoute de protocole», à la page 50. Pour plus d'informations décrivant le Protocol Config MO dans le cadre du traitement des demandes, voir «TLO de traitement des demandes synchrones», à la page 31.

Propriétés définies par l'utilisateur pour le traitement des événements : Le cas échéant, vous pouvez indiquer des propriétés dans le Protocol Config MO HTTP(S). Pour cela, incluez l'attribut UserDefinedProperties. Cet attribut correspond à un objet métier possédant un ou plusieurs attributs enfants avec des valeurs de propriété. Chaque attribut de cet objet enfant doit définir une propriété

unique, afin qu'elle soit lue (ou, pour les réponses synchrones, écrite) dans la partie variable de l'en-tête de message, de la façon suivante :

- Le type de l'attribut doit toujours être String, quel que soit le type de propriété JMS. Les informations spécifiques à l'application de l'attribut peuvent contenir deux paires nom-valeur, définissant le nom et le format de la propriété du message auquel l'attribut est mappé.

Le tableau 10 récapitule les informations spécifiques d'application pour ces attributs.

Tableau 10. Informations spécifiques d'application pour les attributs de propriétés de protocole définis par l'utilisateur : name=contenu paire valeurs

Nom	Valeur	Description
ws_prop_name (casse indifférenciée ; s'il n'est pas indiqué, le nom d'attribut sera employé comme nom de propriété	Tout nom de propriété de protocole valide	Il s'agit du nom de la propriété de protocole. Certains fournisseurs réservent certaines propriétés pour apporter une fonctionnalité étendue.

Si les informations ASI de la propriété personnalisée (ws_prop_name) ne sont pas correctes et s'il n'existe pas de façon logique de traiter cet entête, le connecteur consigne un avertissement et ne tient pas compte de cette propriété. Si la valeur de la valeur personnalisée ne peut être ni défini ni récupérée après la vérification nécessaire de ws_prop_name, le connecteur consigne une erreur et fait échouer l'événement.

Si l'attribut UserDefinedProperties est indiqué, le connecteur créera une instance d'un objet métier UserDefinedProperties. Le connecteur tente alors d'extraire des valeurs de propriété du message pour les stocker dans l'objet métier. Si au moins une propriété est récupérée avec succès, le connecteur définira un attribut UserDefinedProperties modifié dans le Protocol Config MO.

Si, dans le cadre du traitement des événements synchrones, un attribut UserDefinedProperties est indiqué et que son objet métier est instancié, le connecteur traitera chaque attribut de son objet métier enfant et définira la valeur de propriété de message en conséquence.

Propagation des identifiants HTTP pour le traitement des événements : Afin de propager les identifiants, le connecteur prend en charge les attributs Authorization_UserId et Authorization_Password de HTTP Protocol Config MO. Le support est limité à la propagation de ces identifiants dans le cadre du schéma d'authentification de base HTTP.

Un programme d'écoute de protocole HTTP ou HTTPS qui traite une demande de service HTTP incluant un entête d'autorisation analysera celui-ci pour déterminer s'il est conforme à l'authentification de base HTTP. Dans ce cas, le programme d'écoute extrait et décode (en Base64) le nom d'utilisateur et le mot de passe. Cette chaîne décodée se compose d'un nom d'utilisateur et d'un mot de passe séparés l'un de l'autre par le signe deux-points. Si le programme d'écoute de protocole trouve les attributs Authorization_UserId et Authorization_Password dans Protocol Config MO, il affecte à leurs valeurs celles qui ont été extraites de l'entête d'autorisation d'événement.

TLO de traitement des événements asynchrones

La figure 7 représente la hiérarchie des objets métier pour le traitement des événements asynchrones. Seul un objet de demande est obligatoire.

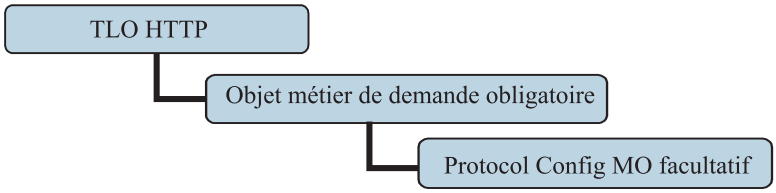


Figure 7. Hiérarchie des objets métier pour le traitement des événements asynchrones.

Le TLO contient des informations ASI de niveau objet ainsi que des attributs avec des informations ASI de niveau attribut. Les deux types d'ASI sont décrits ci-après.

ASI de niveau objet pour les TLO de traitement des événements asynchrones

L'ASI de niveau objet apporte des informations fondamentales sur la nature d'un TLO et sur les objets qu'il contient. La figure 8 représente des informations ASI de niveau objet pour SERVICE_ASYNCH_TLO_Order, TLO exemple pour le traitement des événements asynchrones.

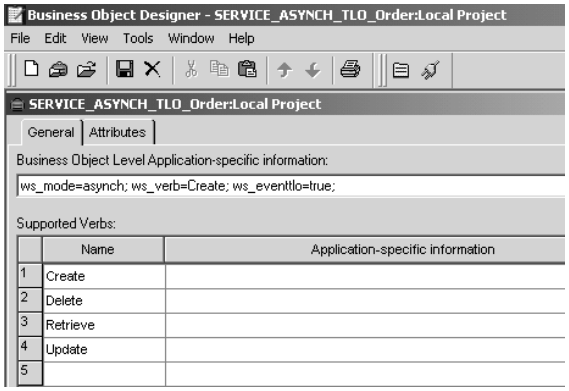


Figure 8. Objet métier de niveau supérieur pour le traitement des événements asynchrones

Le tableau 4 ci-après décrit les informations ASI de niveau supérieur pour un TLO de traitement des événements asynchrones.

Tableau 11. Informations ASI d'objet TLO de traitement des événements asynchrones

ASI de niveau objet	Description
`ws_eventtlo=true`	Si cette propriété ASI est définie sur `true`, le connecteur traite cet objet comme un TLO pour le traitement des événements.
`ws_verb=`*`instruction`*	Avant de livrer le TLO à la collaboration, le connecteur utilise cet ASI pour définir l'instruction sur le TLO. Dans l'exemple représenté dans la figure 8, l'instruction est `Retrieve`.

Tableau 11. Informations ASI d'objet TLO de traitement des événements asynchrones (suite)

ASI de niveau objet	Description
`ws_mode=asynch`	Lors de la notification d'événement, le connecteur emploie cette propriété ASI pour déterminer le mode d'appel de la collaboration : synchrone (`synch`) ou asynchrone (`asynch`). Pour le traitement asynchrone, cet ASI doit être réglé sur `asynch`. La valeur par défaut est `asynch`.

Remarque : A la différence du traitement des événements synchrones, le traitement des événements asynchrones ne requiert d'informations ASI de nom de collaboration au niveau TLO. En fait, le courtier d'intégration assure que les événements d'application atteignent toutes les collaborations souscrivant à de telles combinaisons objet métier-instruction.

ASI de niveau attribut pour les TLO de traitement des événements asynchrones

Chaque TLO de traitement des événements asynchrones possède un seul attribut correspondant à un objet métier de demande. La figure 9 représente l'attribut de demande SERVICE_ASYNCH_TLO_Order, un TLO exemple et l'ASI de l'attribut.

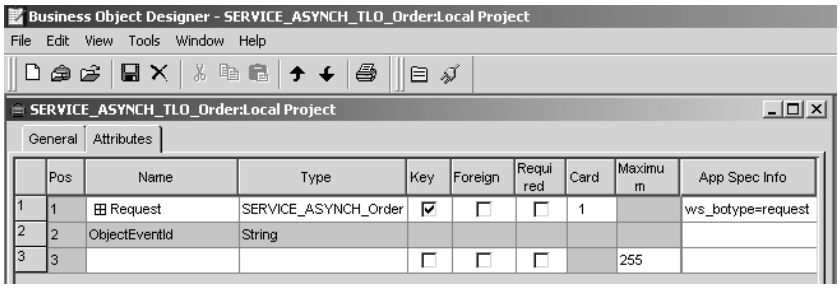


Figure 9. Attribut TLO pour le traitement des événements asynchrones.

Le tableau 12 récapitule les informations ASI de niveau attribut pour les attributs de demande d'un TLO de traitement d'événement asynchrone.

Tableau 12. Informations ASI d'attribut TLO de traitement des événements asynchrones

Attribut TLO	ASI de l'attribut	Description
Request	`ws_botype=request`	Cet attribut correspond à une demande. Le connecteur utilise ses informations ASI pour déterminer si cet attribut TLO est de type BO de demande. Le type d'attribut est déterminé par ces informations et non par le nom d'attribut. S'il existe plusieurs attributs de demande, le connecteur utilise les informations ASI du premier. Cet attribut est obligatoire pour les TLO de traitement des événements synchrones.

Objet métier de demande pour le traitement des événements asynchrones

Un objet métier de demande est enfant d'un TLO et obligatoire pour le traitement des événements asynchrones. Vous pouvez indiquer une instruction par défaut pour l'objet métier de demande. Pour cela, saisissez :

```
DefaultVerb=true;
```

dans la zone ASI de l'instruction dans la liste des instructions prises en charge au niveau le plus élevé de l'objet métier de demande. Si les informations ASI de `DefaultVerb` ne sont pas indiquées et que le gestionnaire de données traite un objet métier sans instruction définie, l'objet métier est renvoyé sans instruction. Les informations ASI de niveau objet d'un objet métier de demande pour le traitement des événements asynchrones est décrit dans le tableau 13.

Tableau 13. Traitement des événements asynchrones : ASI de niveau objet pour les objets métier de demande

ASI de niveau objet	Description
`cw_mo_http=`*HTTPCfgMO*	La valeur de cet ASI doit correspondre au nom de l'attribut associé au Protocol Config MO. L'ASI désigne le programme d'écoute de protocole HTTP ou HTTPS. L'ASI et le Protocol Config MO sont facultatifs. Pour plus d'informations, voir «Protocol Config MO», à la page 25. Remarque : Les gestionnaires de données que vous configurez pour les transformations d'objet métier doivent pouvoir lire tout ASI commençant par `cw_mo` en tant que méta-données et non pas élément de données métier à convertir. Le gestionnaire de données XML est capable de détecter des méta-données `cw_mo`, ignorant les attributs qui leur sont associés.

Tableau 13. Traitement des événements asynchrones : ASI de niveau objet pour les objets métier de demande (suite)

ASI de niveau objet	Description
`ws_tloname=tloname`	Cet ASI indique le nom du TLO auquel appartient cet objet. Lors du traitement d'événements, le connecteur utilise cet ASI pour déterminer si l'objet métier de demande livré par le gestionnaire de données est enfant du TLO. Dans l'affirmative, le connecteur crée le TLO indiqué, définit l'objet métier de demande comme objet enfant et utilise l'ASI de niveau objet des TLO pour le livrer à la collaboration souscriptrice.

TLO de traitement des demandes synchrones

Concernant le traitement des demandes, les TLO autorisés par le connecteur sont de deux types : les TLO synchrones et les TLO asynchrones. Cette section aborde les TLO de traitement des demandes synchrones.

La figure 10 représente la hiérarchie des objets métier TLO pour le traitement des demandes synchrones. Les objets de demande, de réponse et de gestionnaire sont obligatoires, à la différence des objets d'erreur qui sont facultatifs. A la différence du traitement des événements, un Protocol Config MO est obligatoire pour les objets de demande et facultatif pour les objets de réponse et d'erreur.

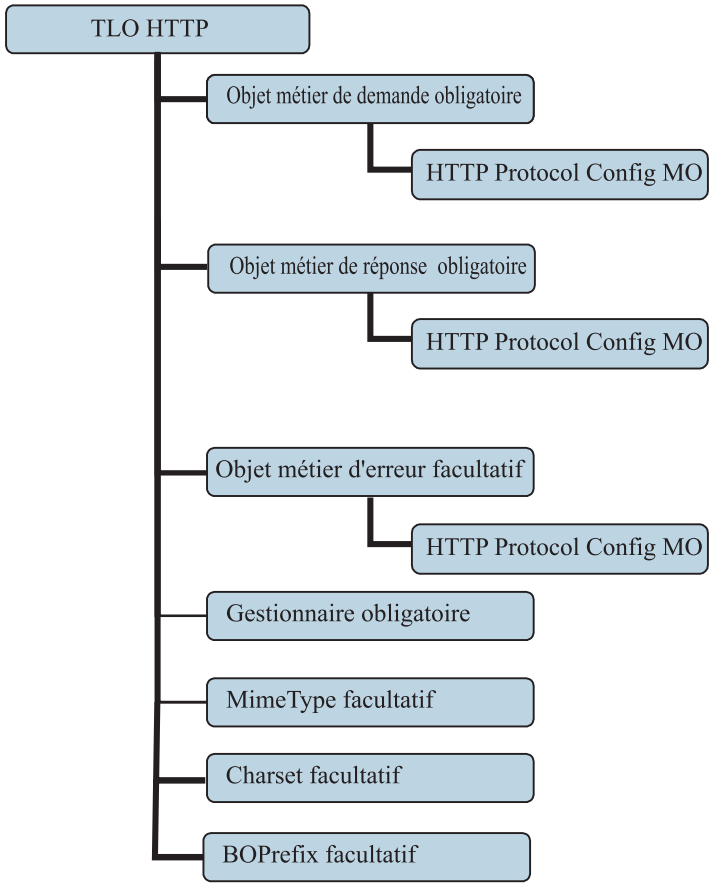


Figure 10. Hiérarchie des objets métier pour le traitement des demandes synchrones.

ASI de niveau objet pour les TLO de traitement des demandes synchrones

L'ASI de niveau objet apporte des informations importantes sur la nature d'un TLO et sur les objets qu'il contient. La figure 11 affiche CLIENT_SYNCH_TLO_OrderStatus, TLO exemple pour le traitement des demandes synchrones.

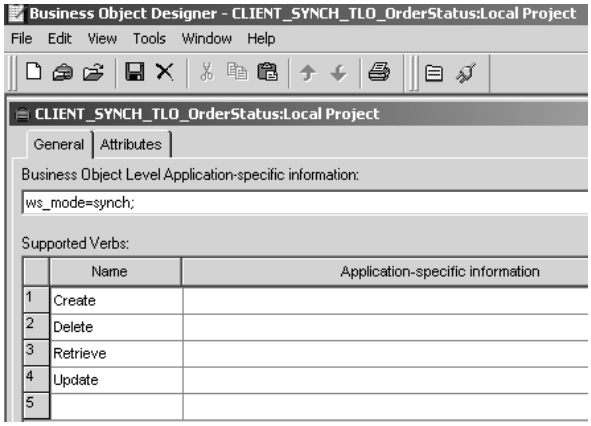


Figure 11. Objet métier de niveau supérieur pour le traitement des demandes synchrones

Le tableau 14 décrit les informations ASI de niveau supérieur pour un TLO de traitement des demandes synchrones. A la différence des informations ASI pour les TLO de traitement des événements synchrones, aucune information ASI `ws_collab`, `ws_verb` ou `ws_eventtlo` n'est requise à ce niveau pour le traitement des demandes.

Tableau 14. Informations ASI d'objet TLO de traitement des demandes synchrones

ASI de niveau objet	Description
`ws_mode=synch`	Lors du traitement des demandes, le connecteur emploie cette propriété ASI pour déterminer le mode d'appel du service HTTP : synchrone (`synch`) ou asynchrone (`asynch`). Si `synch` est indiqué, le connecteur attend une réponse et le TLO doit comprendre des objets métier de demande et de réponse, voire un ou plusieurs objets d'erreur. La valeur par défaut est `asynch`.

ASI de niveau attribut pour les TLO de traitement des demandes synchrones

Le tableau 15 décrit les attributs et les informations ASI pour les TLO de traitement des demandes synchrones.

Tableau 15. Attributs TLO de traitement de demandes

Attribut TLO	ASI de niveau attribut	Description
MimeType	Néant	Cet attribut indique le type mime du gestionnaire de données que le connecteur appelle pour transformer un objet métier de demande en message de demande. Il est possible d'utiliser cette valeur pour transformer des messages de réponse/d'erreur synchrones en objet métier, selon la configuration des règles de transformation des messages.
BOPrefix	Néant	Cet attribut de type `String` est transmis au gestionnaire de données.
Handler	Néant	Destiné au traitement des demandes uniquement, cet attribut indique le gestionnaire de protocole à utiliser pour traiter la demande indiquée. Il accepte la valeur `http`, qui désigne le gestionnaire de protocole HTTP-HTTPS. La valeur par défaut est `http`.
Charset		Ce paramètre facultatif de type String indique le charset à associer au gestionnaire de données lors de la transformation de l'objet métier de demande sortant vers un message. REMARQUE : la valeur charset indiquée dans cet attribut ne sera pas propagée dans l'entête de protocole Content-Type du message de demande.
Request	`ws_botype=request`	Cet attribut correspond à un objet métier de demande. Le connecteur utilise ces informations ASI d'attribut pour déterminer si cet attribut TLO est de type BO de demande. Le type d'attribut est déterminé par ces informations et non par le nom d'attribut. S'il existe plusieurs attributs de demande, le connecteur utilise les informations ASI du premier attribut complété.
Response	`ws_botype=response`	Cet attribut correspond à la réponse retourné par une collaboration et est obligatoire pour le traitement des demandes synchrones. Le connecteur utilise ces informations ASI d'attribut pour déterminer si cet attribut TLO est de type BO de réponse. Le type d'attribut est déterminé par ces informations et non par le nom d'attribut.

Tableau 15. Attributs TLO de traitement de demandes (suite)

Attribut TLO	ASI de niveau attribut	Description

Fault	`ws_botype=fault`
	ou
	`ws_botype=defaultfault`
		Cet attribut, facultatif pour le traitement des demandes synchrones, correspond à un message d'erreur retourné par un service HTTP quand il n'est pas possible de compléter une réponse avec succès.
		
		Le connecteur utilise ces informations ASI pour déterminer si cet attribut TLO est de type BO d'erreur. Le type d'attribut est déterminé par ces informations et non par le nom d'attribut. Un objet métier `defaultfault` est retourné si le message d'erreur est un élément de détail. `defaultfault` est employé dans la résolution d'objets métier par défaut.

Objet métier de demande pour le traitement des demandes synchrones

Un objet métier de demande est enfant d'un TLO et obligatoire pour le traitement des demandes synchrones. Un objet métier de demande possède des informations ASI de niveau objet.

Le tableau 16 décrit les informations ASI de niveau objet d'un objet métier de demandes dans le cadre du traitement des demandes synchrones.

Tableau 16. Traitement des requêtes synchrones : ASI de niveau objet pour les objets métier de demande

ASI de niveau objet	Description
`cw_mo_http=`*`HTTPCfgMO`*	La valeur de cet ASI facultatif doit correspondre au nom de l'attribut associé au Protocol Config MO. Ce Protocol Config MO indique la destination du gestionnaire de protocole HTTP-HTTPS. Ces informations ASI sont utilisées par le gestionnaire de protocole HTTP-HTTPS. Notez que l'attribut de demande TLO doit posséder un HTTP Protocol Config MO pour le traitement de la demande. Pour plus d'informations, voir «HTTP Protocol Config MO pour le traitement des demandes», à la page 36. Remarque : Les gestionnaires de données que vous configurez pour les transformations d'objet métier doivent pouvoir lire tout ASI commençant par `cw_mo` en tant que méta-données et non pas élément de données métier à convertir. Le gestionnaire de données XML est capable de détecter des méta-données `cw_mo`, ignorant les attributs qui leur sont associés.

Objet métier de réponse pour le traitement des demandes synchrones

Un objet métier de réponse est enfant d'un TLO et obligatoire pour le traitement des demandes synchrones. Les informations ASI de niveau objet d'un objet métier de réponse pour le traitement des demandes synchrones est décrit dans le tableau 17.

Tableau 17. Traitement des demandes synchrones : ASI de niveau objet pour les objets métier de réponse

ASI de niveau objet	Description
`cw_mo_http=HTTPCfgMO`	La valeur de cet ASI doit correspondre au nom de l'attribut associé au Protocol Config MO. Facultatif pour un objet métier de réponse, ce Protocol Config MO indique les entêtes dans le message de réponse pour le gestionnaire de protocole HTTP(s). Pour plus d'informations, voir «HTTP Protocol Config MO pour le traitement des demandes», à la page 36. Remarque : Les gestionnaires de données que vous configurez pour les transformations d'objet métier doivent pouvoir lire tout ASI commençant par `cw_mo` en tant que méta-données et non pas élément de données métier à convertir. Le gestionnaire de données XML est capable de détecter des méta-données `cw_mo`, ignorant les attributs qui leur sont associés.

Vous pouvez indiquer une instruction par défaut pour l'objet métier de réponse. Pour cela, saisissez :

```
DefaultVerb=true;
```

dans la zone ASI de l'instruction dans la liste des instructions prises en charge au niveau le plus élevé de l'objet métier de réponse. Si les informations ASI de `DefaultVerb` ne sont pas indiquées et que le gestionnaire de données traite un objet métier sans instruction définie, l'objet métier de réponse est renvoyé sans instruction.

Objet métier d'erreur pour le traitement des demandes synchrones

Un objet métier d'erreur est enfant d'un TLO et obligatoire pour le traitement des demandes synchrones. Les informations ASI de niveau objet d'un objet métier d'erreur pour le traitement des demandes synchrones est décrit dans le tableau 18.

Tableau 18. Traitement des requêtes synchrones : ASI de niveau objet pour les objets métier d'erreur

ASI de niveau objet	Description
`cw_mo_http=HTTPCfgMO`	La valeur de cet ASI doit correspondre au nom de l'attribut associé au Protocol Config MO. Facultatif pour un objet métier d'erreur, ce Protocol Config MO indique les entêtes dans le message de réponse pour le gestionnaire de protocole HTTP-HTTPS. Pour plus d'informations, voir «Protocol Config MO», à la page 25. Remarque : Les gestionnaires de données que vous configurez pour les transformations d'objet métier doivent pouvoir lire tout ASI commençant par `cw_mo` en tant que méta-données et non pas élément de données métier à convertir. Le gestionnaire de données XML est capable de détecter des méta-données `cw_mo`, ignorant les attributs qui leur sont associés.

HTTP Protocol Config MO pour le traitement des demandes

Pendant le traitement des demandes, les gestionnaires de protocole HTTP-HTTPS utilisent un Protocol Config MO pour déterminer la destination du service HTTP cible. Ce Protocol Config MO est nécessaire pour un objet métier de demande. Les gestionnaires de protocole HTTP-HTTPS prennent en charge les demandes HTTP 1.0 POST et GET. Comme le montre le tableau 19, le seul attribut obligatoire (Destination) est l'URL complète du service HTTP cible. Les attributs d'autorisation facultatives sont décrits dans les sections ci-après.

Tableau 19. Attributs HTTP Protocol Config MO pour le traitement des demandes

Attribut	Obligatoire	Type	Description
Destination	Obligatoire pour le traitement des demandes dans l'objet métier Request.	`String`	URL de destination du service HTTP cible. Le gestionnaire de protocole HTTP-HTTPS utilise cet attribut pour déterminer la destination du service HTTP.

Tableau 19. Attributs HTTP Protocol Config MO pour le traitement des demandes (suite)

Attribut	Obligatoire	Type	Description
Content-Type	Obligatoire pour le traitement des demandes dans l'objet métier Request quand la charge de demande doit être générée. La valeur de cet attribut est obligatoire en cas de réponse synchrone du programme d'écoute de protocole quand la charge de la demande est vide alors que la charge de la réponse ne l'est pas.	`String`	La valeur de cet attribut définit l'entête Content-Type du message sortant (incluant le ContentType du message et facultativement le charset du message sortant). La syntaxe est la même que pour l'entête Content-Type dans le protocole HTTP ; par exemple : `text/xml; charset=ISO-8859-4`.
Method	Non	`String`	Utilisé dans le traitement des demandes pour déterminer la méthode HTTP. Les valeurs possibles sont `POST` et `GET`. La valeur par défaut est `POST`.
Authorization_UserId	Non	`String`	Cet attribut correspond à l'ID utilisateur de l'authentification HTTP de base. Pour plus d'informations, voir «HTTP Propagation d'identifiants pour le traitement des demandes» , à la page 42
Authorization_Password	Non	`String`	Cet attribut correspond au mot de passe de l'authentification HTTP de base. Pour plus d'informations, voir «HTTP Propagation d'identifiants pour le traitement des demandes» , à la page 42
Un ou plusieurs entêtes HTTP	Non	String	Cet attribut permet au gestionnaire de transmettre ou de récupérer la valeur de l'entête HTTP indiqué.
UserDefinedProperties	Non	Objet métier	Cet attribut comprend l'objet métier des propriétés de protocole définies par l'utilisateur. Pour plus d'informations, voir «Propriétés définies par l'utilisateur pour le traitement des demandes», à la page 38.

Tableau 19. Attributs HTTP Protocol Config MO pour le traitement des demandes (suite)

Attribut	Obligatoire	Type	Description
MessageTransformationMap	Facultatif pour le traitement des demandes - réponse synchrone uniquement. Inutilisé dans le traitement des événements.	Objet métier de cardinalité simple	Il s'agit de l'attribut pointant vers un objet métier qui contient zéro, une ou plusieurs règles de transformation de message. Les règles comportent des informations relatives au type mime et au charset à appliquer au message entrant indiqué dans la règle. Pour plus d'informations, voir «Mappes de transformation de message», à la page 39.

Les attributs HTTP Protocol Config MO sont décrits dans :

- «Propriétés définies par l'utilisateur pour le traitement des demandes»
- «Mappes de transformation de message», à la page 39
- «HTTP Propagation d'identifiants pour le traitement des demandes», à la page 42

Propriétés définies par l'utilisateur pour le traitement des demandes : Le cas échéant, vous pouvez indiquer des propriétés dans le Protocol Config MO HTTP. Pour cela, incluez l'attribut UserDefinedProperties. Cet attribut correspond à un objet métier possédant un ou plusieurs attributs enfants avec des valeurs de propriété. Chaque attribut de cet objet enfant doit définir une propriété unique, afin qu'elle soit lue (ou, pour les réponses synchrones, écrite) dans la partie variable de l'en-tête de message, de la façon suivante :

- Le type de l'attribut doit toujours être `String`. Les informations spécifiques à l'application de l'attribut peuvent contenir la paire nom-valeur, définissant le nom de la propriété du message auquel l'attribut est mappé.

Le tableau 20 récapitule les informations spécifiques d'application pour ces attributs.

Tableau 20. Informations spécifiques d'application pour les attributs de propriétés de protocole définis par l'utilisateur : name=contenu paire valeurs

Nom	Valeur	Description
ws_prop_name (casse indifférenciée ; s'il n'est pas indiqué, le nom d'attribut sera employé comme nom de propriété	Tout nom de propriété de protocole valide	Il s'agit du nom de la propriété de protocole. Certains fournisseurs réservent certaines propriétés pour apporter une fonctionnalité étendue.

Si les informations ASI de la propriété personnalisée (ws_prop_name) ne sont pas correctes et s'il n'existe pas de façon logique de traiter cet entête, le connecteur consigne un avertissement et ne tient pas compte de cette propriété. Si la valeur de la valeur personnalisée ne peut être ni défini ni récupérée après la vérification nécessaire de ws_prop_name, le connecteur consigne une erreur et fait échouer l'événement.

Si un attribut UserDefinedProperties est indiqué et que son objet métier est instancié, le connecteur traite chaque attribut de son objet métier enfant et définit les valeurs des propriétés de message en conséquence.

Par exemple, dans le cadre du traitement des demandes synchrones, si l'attribut UserDefinedProperties a été indiqué, dès la réception d'un message de réponse, le connecteur crée une instance d'un objet UserDefinedProperties et tente d'extraire des valeurs de propriété dans le message pour les stocker dans le nouvel objet métier. Si au moins une propriété a été récupérée avec succès, le connecteur définira un objet métier UserDefinedProperties modifié dans le Protocol Config MO.

Mappes de transformation de message : La fonction MTM (Message Transformation Map) n'est prise en charge que pour les gestionnaires de protocole HTTP(S) de traitement des demandes. MessageTransformationMap est un attribut facultatif dans le Protocol Config MO pointant vers un objet métier. L'objet métier contient des règles de transformation comportant des types mime et des charsets. S'il rencontre l'attribut MessageTransformationMap (à casse différenciée) et que celui-ci est de type objet métier, le connecteur utilise les règles dans cet objet pour transformer un message.

L'attribut MessageTransformationMap possède deux attributs d'objet métier enfants dénommés TransformationRule et EmptyResponseRule. Lors de la tentative de localisation de l'attribut TransformationRule d'un message non vide, le gestionnaire de protocole HTTP-HTTPS essaie tout d'abord de trouver le message correspondant au ContentType indiqué dans tous les TransFormationRules. En cas d'échec, le connecteur tente de trouver la règle qui s'applique à plusieurs types de messages. Quand la réponse ne contient que des entêtes HTTP, le gestionnaire de protocole utilise l'attribut d'objet métier EmptyResponseRule de MessageTransformationMap. Pour plus d'informations sur le traitement de gestionnaire de protocole, voir «HTTP-HTTPS traitement du gestionnaire de protocole», à la page 60.

Les attributs d'objet métier TransformationRule sont répertoriés dans le tableau 21.

Les attributs d'objet métier EmptyResponseRule sont répertoriés dans le tableau 22, à la page 41.

Tableau 21. Attributs TransformationRule de MessageTransformationMaps dans HTTP Protocol Config MO

Nom de l'attribut	Obligatoire	Type	Valeur par défaut	Description
TransformationRule	Facultatif pour le traitement des demandes. Inutilisé dans le notification d'événements.	Objet métier, cardinalité N		Il s'agit de l'attribut contenant une règles pour la transformation de message. Il peut exister une ou plusieurs instances de cet attribut ou n'en exister aucune sous l'attribut MessageTranformationMap.

Tableau 21. Attributs TransformationRule de MessageTransformationMaps dans HTTP Protocol Config MO (suite)

Nom de l'attribut	Obligatoire	Type	Valeur par défaut	Description
+ContentType	Oui	String	*/*	La valeur de cette propriété indique le ContentType HTTP du message pour lequel cette règles de transformation s'applique. La valeur par défaut */* de cet attribut permet au connecteur d'appliquer cette règles à n'importe quel ContentType. Pour plus d'informations sur le traitement de gestionnaire de protocole, voir «HTTP-HTTPS traitement du gestionnaire de protocole», à la page 60. Notez que si le gestionnaire de protocole trouve plusieurs règles possédant le même ContentType comme autre règle, il consigne l'avertissement, ignore les règles en double, mais utilise des règles uniques.
+MimeType	Non	String		Type mime à utiliser lors de l'appel d'un gestionnaire de données pendant le traitement de messages de ContentType indiqué dans l'objet métier.
+Charset	Non	String		Charset à utiliser lors de la transformation d'une demande du ContentType indiqué dans l'objet métier.
+BOPrefix	Non	String		La valeur de cet attribut remplace le BOPrefix du TLO de la réponse synchrone de la charge en transformation d'objet métier.
+BOName	Non	String		La valeur de cette propriété est transmise au gestionnaire de données dans la table de hachage des entêtes avec BOName comme nom de l'élément. Si un entête de protocole porte le même nom, la valeur d'entête de protocole est prioritaire sur cette valeur de propriété.
+BOVerb	Non	String		La valeur de cette propriété est transmise au gestionnaire de données dans la table de hachage des entêtes avec BOVerb comme nom de l'élément. Si un entête de protocole porte le même nom, la valeur d'entête de protocole est prioritaire sur cette valeur de propriété.

Tableau 22. Attributs EmptyResponseRule de MessageTransformationMaps dans HTTP Protocol Config MO

Nom de l'attribut	Obligatoire	Type	Valeur par défaut	Description
EmptyResponseRule	Non	Objet métier de cardinalité simple		Encapsule la règle d'appel du gestionnaire de données de la réponse vide.
+Action	Non	`String`	Ignore	Transmet l'une des valeurs à casse indifférenciée suivantes : Ignore - consigne un message de trace indiquant la réception d'une réponse vide, ne tente pas de générer l'objet métier de réponse. Warning - consigne un message d'avertissement indiquant la réception d'une réponse vide, ne tente pas de générer l'objet métier de réponse. Error - consigne un message d'erreur indiquant la réception d'une réponse vide, ne tente pas de générer l'objet métier de réponse. Fail - consigne un message d'erreur indiquant la réception d'une réponse vide, ne tente pas de générer l'objet métier de réponse, fait échouer la demande. Process - appelle le gestionnaire de données transmettant des entêtes http comme Hashtable à la place du paramètre d'objet de configuration.
+MimeType	Non	`String`		Type mime du gestionnaire de données à appeler si la valeur de l'attribut Action est Process.
+Charset	Non	`String`		Charset à transmettre au gestionnaire de données. S'il n'est pas indiqué, aucun charset n'est transmis.
+BOPrefix	Non	`String`		La valeur de cet attribut remplace le BOPrefix du TLO de la réponse synchrone de la charge en transformation d'objet métier.
+BOName	Non	`String`		La valeur de cette propriété est transmise au gestionnaire de données dans la table de hachage des entêtes avec BOName comme nom de l'élément. Si un entête de protocole porte le même nom, la valeur d'entête de protocole est prioritaire sur cette valeur de propriété.

Tableau 22. Attributs EmptyResponseRule de MessageTransformationMaps dans HTTP Protocol Config MO (suite)

Nom de l'attribut	Obligatoire	Type	Valeur par défaut	Description
+BOVerb	Non	String		La valeur de cette propriété est transmise au gestionnaire de données dans la table de hachage des entêtes avec BOVerb comme nom de l'élément. Si un entête de protocole porte le même nom, la valeur d'entête de protocole est prioritaire sur cette valeur de propriété.

HTTP Propagation d'identifiants pour le traitement des demandes : Afin de propager les identifiants, le connecteur prend en charge les attributs Authorization_UserId et Authorization_Password de HTTP Protocol Config MO. Le support est limité à la propagation de ces identifiants dans le cadre du schéma d'authentification de base HTTP.

La collaboration définit les valeurs des attributs Authorization_UserId et Authorization_Password dans le Protocol Config MO. Si ces attributs ne sont ni nuls ni vides, le connecteur crée un entête d'autorisation sur la demande adressée au service HTTP cible. Le gestionnaire de protocole HTTP/HTTPS se base sur *HTTP Authentication: Basic and Digest Access Authentication (RFC 2617)* lors de la création de l'entête d'autorisation.

Remarque : Le schéma d'authentification condensé n'est pas pris en charge ; il en va de même du mécanisme défi-réponse facultatif dans le cadre de l'authentification HTTP définie dans le document Rfc2617. Si le gestionnaire de protocole HTTP(s) appelle un serveur nécessitant des données d'identification, le connecteur n'attend pas la réponse défi de la part du serveur. En fait, il envoie les données d'identification en continu.

TLO de traitement des demandes asynchrones

La figure 12 représente la hiérarchie des objets métier pour le traitement des demandes asynchrones. Un objet de demande et de gestionnaire sont requis. L'objet de demande contient un Protocol Config MO pour le gestionnaire de protocole HTTP-HTTPS. Ils sont décrits dans les sections ci-après.

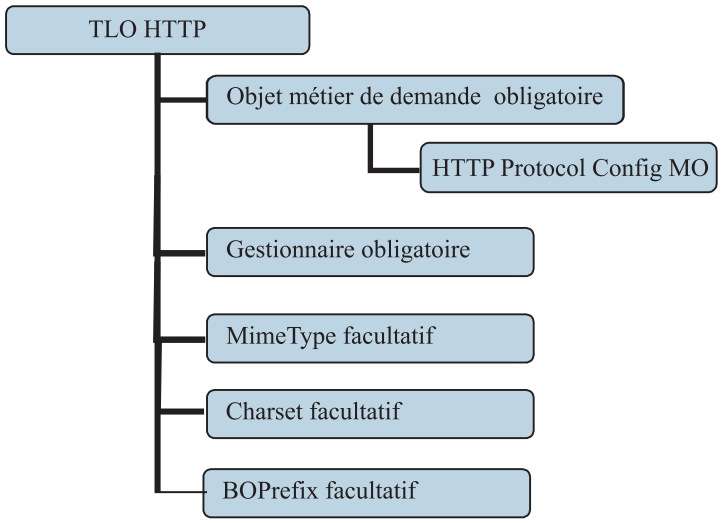


Figure 12. Hiérarchie des objets métier pour le traitement des demandes asynchrones.

Le TLO contient des informations ASI de niveau objet ainsi que des attributs avec des informations ASI de niveau attribut. Les deux types d'ASI sont décrits ci-après.

ASI de niveau objet pour les TLO de traitement des événements asynchrones

La figure 13 représente CLIENT_ASYNCH_Order_TLO, TLO exemple pour le traitement des demandes asynchrones.

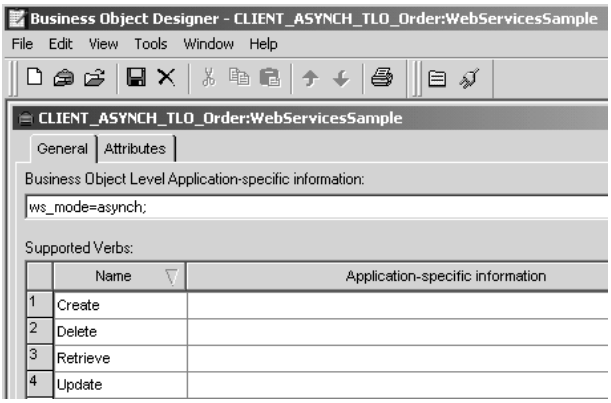


Figure 13. Objet métier de niveau supérieur pour le traitement des demandes asynchrones

Le tableau 23 ci-après décrit les informations ASI de niveau supérieur pour un TLO de traitement des demandes asynchrones.

Tableau 23. Informations ASI d'objet TLO de traitement des demandes asynchrones

ASI de niveau objet	Description
`ws_mode=asynch`	Lors du traitement des demandes, le connecteur emploie cette propriété ASI pour déterminer le mode d'appel de la collaboration : synchrone (`synch`) ou asynchrone (`asynch`). Pour le traitement des demandes asynchrones, cet ASI doit être réglé sur `asynch`. La valeur par défaut est `asynch`.

ASI de niveau attribut pour les TLO de traitement des demandes asynchrones

Le tableau 24 récapitule les informations ASI de niveau attribut pour les attributs de demande d'un TLO de traitement de demande asynchrone.

Tableau 24. Attributs TLO de traitement de demandes asynchrones

Attribut TLO	ASI de l'attribut	Description
MimeType	Néant	Cet attribut indique le type mime du gestionnaire de données que le connecteur appelle. Notez que cet attribut n'est utilisé que pour le traitement des demandes.
BOPrefix	Néant	La valeur de cet attribut est transmise au gestionnaire de données.
Handler	Néant	Destiné au traitement des demandes uniquement, cet attribut indique le gestionnaire de protocole à utiliser pour traiter la demande indiquée. Il accepte la valeur `http`, qui désigne le gestionnaire de protocole HTTP-HTTPS qui doit traiter la demande. La valeur par défaut est `http`.
Charset		Ce paramètre facultatif de type String indique le charset à associer au gestionnaire de données lors de la transformation de l'objet métier de demande sortant vers un message. REMARQUE : la valeur charset indiquée dans cet attribut ne sera pas propagée dans l'entête de protocole Content-Type du message de demande.
Request	`ws_botype=request`	Cet attribut correspond à un objet métier de demande de service HTTP. Le connecteur utilise ces informations ASI d'attribut pour déterminer si cet attribut TLO est de type BO de demande. Le type d'attribut est déterminé par ces informations et non par le nom d'attribut. S'il existe plusieurs attributs de demande, le connecteur utilise les informations ASI du premier.

Objet métier de demande pour le traitement des demandes asynchrones

Un objet métier de demande est enfant d'un TLO et obligatoire pour le traitement des demandes asynchrones. Les informations ASI de niveau objet d'un objet métier de demande pour le traitement des demandes asynchrones est décrit dans le tableau 25.

Tableau 25. Traitement des requêtes asynchrones : ASI de niveau objet pour les objets métier de demande

ASI de niveau objet	Description
`cw_mo_http=HTTPCfgMO`	La valeur de cet ASI doit correspondre au nom de l'attribut associé au Protocol Config MO. Ce Protocol Config MO indique la destination du gestionnaire de protocole HTTP-HTTPS. Ces informations ASI sont utilisées par le gestionnaire de protocole HTTP-HTTPS. Notez que l'attribut de demande TLO doit posséder un HTTP Protocol Config MO pour le traitement de la demande. Pour plus d'informations, voir «HTTP Protocol Config MO pour le traitement des demandes», à la page 36. Remarque : Les gestionnaires de données que vous configurez pour les transformations d'objet métier doivent pouvoir lire tout ASI commençant par `cw_mo` en tant que méta-données et non pas élément de données métier à convertir. Le gestionnaire de données XML est capable de détecter des méta-données `cw_mo`, ignorant les attributs qui leur sont associés.

Protocol Config MO pour le traitement des demandes asynchrones

Pendant le traitement des demandes, le gestionnaire de protocole HTTP-HTTPS utilise un Protocol Config MO pour déterminer la destination du service HTTP cible. Ce Protocol Config MO est nécessaire pour un objet métier de demande. Pour plus d'informations, voir «HTTP Protocol Config MO pour le traitement des demandes», à la page 36.

Développement d'objets métier

Vous utiliserez Business Object Designer pour créer des objets métier, et Connector Configurator pour configurer le connecteur nécessaire pour les prendre en charge. Pour plus d'informations sur l'outil Business Object Designer, voir le document *Business Object Development Guide* et l'Annexe B, «Connector Configurator», à la page 113.

Chapitre 4. Connecteur HTTP

Ce chapitre décrit le connecteur HTTP et la façon de le configurer.

Tous les connecteurs d'intégration WebSphere fonctionnent avec un courtier d'intégration. Le connecteur HTTP travaille avec le courtier IBM WebSphere InterChange Server, décrit dans le document *Technical Introduction to IBM WebSphere InterChange Server*.

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

- Reçoit et envoie des objets métier.
- Assure l'échange des messages de démarrage et d'administration.

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

Pour plus d'informations sur la relation du courtier d'intégration avec le connecteur, voir le document *System Administration Guide*.

Traitement du connecteur

Le connecteur comprend une structure d'écoute de protocole et une structure de gestionnaire de protocole pour le traitement des demandes. Cette fonction bidirectionnelle permet à la structure de connecteur de :

- Traiter les appels émanant de clients HTTP (traitement des événements)
- Traiter une demande à l'aide d'une collaboration qui appelle un service HTTP (traitement des demandes)

Présentation générale du traitement des événements

Le traitement des événements du connecteur (ou notification des événements) permet de gérer les demandes émanant de clients HTTP. Cette fonction de traitement des événements comprend une structure d'écoute de protocole, incluant les composants suivants traités en détails plus loin dans ce chapitre :

- programme d'écoute de protocole HTTP
- programme d'écoute de protocole HTTPS

Le connecteur utilise ces composants pour écouter les appels de clients à destination des collaborations via le transport de communication.

Quand il reçoit une demande émanant d'un client, le module d'écoute convertit le message en objet métier et appelle la collaboration. S'il s'agit d'une demande synchrone, le connecteur reçoit un objet métier de réponse du même type que l'objet métier de réponse. Le programme d'écoute convertit l'objet métier de réponse dans le message de réponse. Le programme d'écoute transporte alors le message de réponse jusqu'au client. Notez que le séquençage d'événement n'est pas obligatoire pour ce connecteur ; le connecteur peut livrer les événements dans n'importe quel ordre.

Le connecteur HTTP utilise le gestionnaire de données configuré pour convertir des messages de demande entrants en objets métier. Pour que le gestionnaire de données détermine quel objet métier résoudre pour le message de demande entrant, le connecteur fournit au gestionnaire de données des méta-informations concernant ses objets métier pris en charge. A partir de ses objets métier pris en charge, le connecteur établit tout d'abord la liste de tous les objets métier qui sont candidats potentiels pour la conversion. Cette liste ne comprend que les TLO pris en charge. Les objets métier TLO pris en charge sont ceux ayant pour paramètre `ws_eventtlo=true` des informations ASI de niveau objet.

Le programme d'écoute du protocole lit les informations ASI de niveau objet du TLO comme suit :

- `ws_collab=` Ceci détermine quelle collaboration appeler
- `ws_mode=` Ceci détermine le mode d'appel de la collaboration, c'est-à-dire de façon synchrone (`synch`) ou de façon asynchrone (`asynch`)

Le connecteur inspecte l'objet métier de la demande extrait par le gestionnaire de données. Il utilise les informations ASI `ws_tloname` de cet objet métier pour extraire le nom du TLO parent. Ce TLO sera instancié et l'objet métier de la demande sera défini dans le TLO. Enfin, ce TLO construit servira à appeler la collaboration.

Pour une collaboration synchrone, le connecteur utilise le gestionnaire de données pour créer un message de réponse ou d'erreur à renvoyer au client. Dans ce cas, le connecteur transmet simplement un objet métier (enfant de TLO) au gestionnaire de données. Le gestionnaire de données renvoie un message basé sur l'objet métier qui lui est transmis.

Présentation générale du traitement des demandes

Au nom d'une collaboration, le connecteur peut appeler des services HTTP sur HTTP(S). La fonction de traitement des demandes est prise en charge par une structure de gestionnaire de protocole. Cette structure est un module d'exécution configurable composée du gestionnaire de protocole HTTP-HTTPS, abordé en détail ultérieurement dans le présent chapitre.

Lors de la réception d'un objet métier de demande de collaboration, qui est toujours défini dans un TLO, la structure de gestionnaire de protocole charge le gestionnaire de protocole. Le gestionnaire de protocole gère les informations détaillées de niveau transport nécessaires à l'appel du service HTTP et (le cas échéant) à la sécurisation d'une réponse, en effectuant trois tâches principales : conversion d'un objet métier de demande de collaboration en message de demande, appel du service HTTP avec ce message, et, si la demande est soumise dans le mode demande/réponse (synchrone), conversion du message de réponse en objet métier et renvoi de cet objet à la collaboration.

Le connecteur HTTP est toujours appelé depuis une collaboration utilisant des TLO. Le connecteur détermine l'objet métier de la demande depuis le TLO, et appelle le gestionnaire de données avec cet objet métier. Le gestionnaire de données retourne un message de demande envoyé par le connecteur au service HTTP.

Pour une exécution synchrone, le connecteur utilise le gestionnaire de données pour convertir des messages de réponse et d'erreur en objets métier de réponse et d'erreur. Pour que le gestionnaire de données détermine quel objet métier pour ces réponses/erreurs en conversions d'objet métier, le connecteur fournit des méta-informations spécifiques au gestionnaire de données. En particulier, le connecteur établit la liste de tous les objets métier de réponse et d'erreur qui sont enfant du TLO appelant. A la différence de l'objet métier de réponse qui doit être unique, il peut exister plusieurs objets métier d'erreur. Il peut également exister un seul et unique objet métier defaultfault. Le connecteur transmet simplement au gestionnaire de données le nom de l'objet métier defaultfault. L'objet métier defaultfault doit être résolu par le gestionnaire de données en dernier lieu si aucun autre objet métier d'erreur n'est résolu pour cette transformation.

Appel de gestionnaire de données personnalisé

Quand un message HTTP est converti en objet métier dans le cadre de l'appel du gestionnaire de données, l'adaptateur HTTP construit une table de hachage des en-têtes entrants, en plus des paires nom-valeur spéciales telles que BOName, BOVerb et Destination (spécifiques au traitement des demandes d'événement). Le gestionnaire de données personnalisé reçoit cette table de hachage en lieu et place du paramètre `Object` *`config`* et l'utilise pour accéder au message et aux en-têtes définis de l'adaptateur.

Services HTTP(S)

Les services HTTP prennent en charge le protocole de transport HTTP. HTTP implémente un modèle client-serveur dans lequel un client HTTP ouvre une connexion et envoie un message de demande à un serveur HTTP. Le message de demande client permet d'appeler le service HTTP. Le serveur HTTP distribue le message contenant l'appel et ferme la connexion.

Les programmes d'écoute de protocole HTTP et HTTPS du connecteur exploitent les modèles client-serveur HTTP et Demande/Réponse quand le client de gestion demande une collaboration. Cependant, le programme d'écoute HTTP n'est pas conçu pour jouer le rôle de serveur HTTP (proxy, intermédiaire ou autre). En fait, il fonctionne comme point d'extrémité utilisé derrière un pare-feu sur un serveur d'entreprise. Ainsi, un serveur Web ou une passerelle distincte doit être déployé dans le pare-feu pour acheminer les demandes clients jusqu'au programme d'écoute. Pour plus d'informations, voir Chapitre 1, «Présentation de l'adaptateur», à la page 1.

Service HTTP(S) synchrone

Du point de vue du traitement des connecteurs, un service HTTP synchrone est un service obéissant à la méthode Demande/Réponse. Si le programme d'écoute de protocole HTTP ou HTTPS parvient à traiter un message de demande HTTP, le corps contiendra la réponse et le code d'état HTTP `200 OK`. En cas d'erreur, le corps contiendra le message d'erreur et le code état `500`.

Service HTTP(S) asynchrone

Du point de vue du traitement des connecteurs, un service HTTP asynchrone est un service obéissant à la méthode de demande uniquement. Si le programme d'écoute de protocole HTTP ou HTTPS reçoit et traite avec succès une opération demande uniquement, un code d'état HTTP `202 Accepted` est généré. Vous pouvez également configurer le connecteur pour générer un code état HTTP `200 OK` — pour plus d'informations, voir la propriété HTTPAsyncResponseCode dans le tableau 35. En cas d'erreur, un code état HTTP `500` est généré. Aucune réponse n'est adressée bien qu'un corps de message d'erreur puisse être renvoyé.

Traitement des événements

Lors du traitement des événements, le connecteur se sert des programmes d'écoute de protocole et du ou des gestionnaires de données configurés pour convertir les messages de demande des clients du service HTTP en objets métier pouvant être manipulés par des collaborations. Les programmes d'écoute de protocole jouent un rôle crucial dans le traitement des événements.

Programmes d'écoute de protocole

Les demandes HTTP peuvent être acheminées par HTTP ou HTTPS. Le programme d'écoute surveille l'arrivée de telles demandes sur son canal de transport. On compte deux programmes d'écoute de protocole et les canaux correspondants :

- programme d'écoute de protocole HTTP
- programme d'écoute de protocole HTTPS

Chacun d'eux consiste en une unité d'exécution qui écoute le trafic sur le protocole de transport. Quand il reçoit un message de demande émanant d'un client, le programme d'écoute enregistre l'événement auprès de la structure d'écoute de protocole.

La structure de programme d'écoute de protocole gère les programmes d'écoute de protocole, planifiant les demandes à mesure que des ressources sont disponibles. Vous configurez les programmes d'écoute et les aspects de la structure de programme d'écoute de protocole lors de la définition de valeurs des propriétés spécifiques de connecteur. Parmi les propriétés de structure de programme d'écoute de protocole, vous pouvez configurer les propriétés suivantes :

- **WorkerThreadCount** Nombre total d'unités d'exécution disponibles pour la structure de programme d'écoute de protocole, correspondant au nombre de demandes qu'il peut traiter en parallèle.
- **RequestPoolSize** Nombre maximal de demandes pouvant être enregistrées avec la structure de programme d'écoute de protocole. Si elle reçoit un nombre excessif de demandes, elle ne peut plus les traiter.

Ces deux propriétés spécifiques au connecteur contrôlent l'allocation de mémoire de sorte qu'elles empêchent les programmes d'écoute de protocole de celui-ci de boucler sur des événements infinis. L'algorithme d'allocation se présente ainsi : à tout moment, le connecteur peut recevoir un nombre total d'événements égal à WorkerThreadCount + RequestPoolSize. Il peut traiter le nombre *WorkerThreadCount* de demandes en parallèle.

Vous pouvez brancher des programmes d'écoute de protocole supplémentaires dans la structure de programme d'écoute de protocole. Pour plus d'informations, voir «Création programmes d'écoute multiples», à la page 79 et «Propriétés de configuration spécifiques au connecteur», à la page 67.

Pingability

Les programmes d'écoute peuvent être configurés pour répondre aux demandes ping avec des codes réponse indiqués par l'utilisateur. En général, les demandes ping surviennent avant des demandes complètes et permettent de valider la réactivité de l'URL du programme d'écoute. La propriété hiérarchique `Pingability` permet de capturer les paramètres de configuration. Si ce service est activé dans le programme d'écoute, le programme d'écoute vérifie que la méthode de la demande correspond à l'une des propriétés enfants de `Pingability` et répond avec un code état et une raison définis par l'utilisateur.

Remarque : Attribuer le caractère "pingable" à une méthode HTTP empêche le programme d'écoute de générer des demandes qui la contiennent. Par conséquent, les méthodes POST et GET ne doivent pas être définies avec la propriété Pingable pour permettre leur traitement.

HTTP et HTTPS traitement du programme d'écoute de protocole

Le programme d'écoute de protocole HTTP(S) consiste en une unité d'exécution qui écoute de façon continue les demandes HTTP(S) émanant de clients. L'unité d'exécution du programme d'écoute lie l'hôte et le port indiqués par les propriétés de configuration Host et Port spécifiques au connecteur. Une autre propriété de configuration —RequestWaitTimeout— définit l'intervalle durant lequel le programme d'écoute attend une demande avant de vérifier si le connecteur s'est arrêté.

La figure 14 illustre le traitement du programme d'écoute de protocole HTTP dans le cadre d'une opération synchrone.

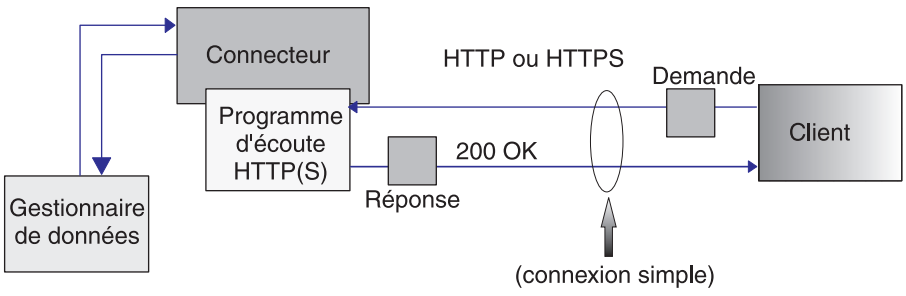


Figure 14. Programme d'écoute de protocole HTTP : traitement des événements synchrones

La figure 15 illustre le traitement du programme d'écoute de protocole HTTP dans le cadre d'une opération asynchrone.

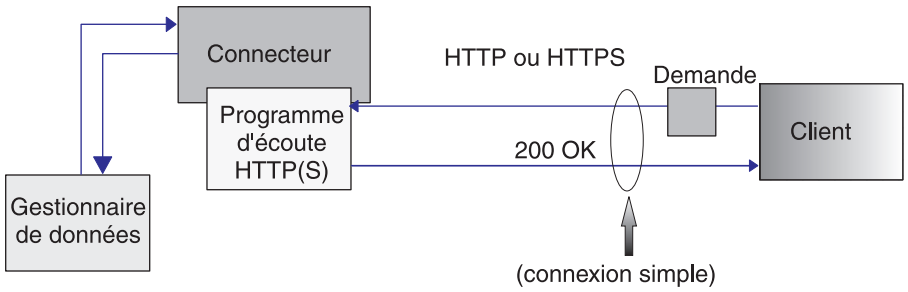


Figure 15. Programme d'écoute de protocole HTTP : traitement des événements asynchrones

En lançant une demande HTTP ou HTTPS, un client envoie un message de demande au programme d'écoute HTTP ou HTTPS. Le client doit utiliser la méthode HTTP POST ou GET pour appeler l'URL du programme d'écoute de protocole.

Lorsqu'il reçoit une demande HTTP(S), le programme d'écoute l'enregistre auprès de la structure de programme d'écoute de protocole, qui planifie le traitement de l'événement à mesure que des ressources se libèrent. Le programme d'écoute extrait alors de la demande les entêtes de protocole et la charge.

Le tableau 26, à la page 53 récapitule la priorité des règles qu'utilise le programme pour déterminer l'en-tête Charset, MimeType, ContentType et Content-Type des messages entrants en cas de **présence de charge**.

Remarque : En cas d'**absence de charge** avec les messages entrants, consultez le tableau 27, à la page 53, qui récapitule l'ordre de priorité des règles qu'utilise le programme d'écoute pour identifier l'en-tête Charset, MimeType, ContentType et Content-Type.

Tableau 26. Règles de traitement des messages entrants en présence de charge

Priorité	Charset	MimeType	ContentType	Entête Content-Type
1	Entête Content-Type du message HTTP entrant	Valeur de propriété URLsConfiguration pour ce programme d'écoute	Entête Content-Type du message HTTP entrant	Entête Content-Type du message HTTP entrant
2	Valeur de propriété URLsConfiguration pour ce programme d'écoute	Valeur par défaut : ContentType		
3	Si le type ContentType du message de demande est `text` avec un sous-type (par exemple, `text/xml`, `text/plain`, etc.), la valeur par défaut est ISO-8859-1. Autrement, charset ne sera pas utilisé.			

Tableau 27. Règles de traitement des messages entrants en l'absence de charge

Priorité	Charset	MimeType	ContentType	Entête Content-Type
1	Lecture à partir de la propriété du programme d'écoute EmptyRequestRule	Lecture à partir de la propriété du programme d'écoute EmptyRequestRule Listener	Le gestionnaire de données appelé peut renvoyer ContentType comme partie de l'en-tête Content-Type	Le gestionnaire de données appelé peut renvoyer l'en-tête Content-Type
2	Charset ne sera pas utilisé.	Echec de la demande		
3				

Comme le montre le tableau 26 :

- Le programme d'écoute de protocole détermine le Charset du message entrant selon les règles suivantes :
  1. Le programme d'écoute tente d'extraire la valeur Charset du paramètre de même de la valeur d'entête Content-Type du message HTTP.
  2. Si aucune valeur Charset n'est obtenue depuis l'en-tête Content-Type, le programme d'écoute de protocole tente de lire la valeur de la propriété URLsConfiguration pour ce programme d'écoute.
  3. Si la valeur Charset n'est pas obtenue à l'aide des méthodes décrites aux étapes précédentes et si le type ContentType du message est `text` avec un sous-type (par exemple, `text/xml`, `text/plain`, etc.), le programme d'écoute utilise ISO-8859-1 comme valeur de Charset par défaut. Autrement, la valeur de Charset ne sera pas utilisée.

- Le programme d'écoute détermine le MimeType du message de demande selon ces règles :
  1. Si vous avez configuré TransformationRules pour l'URL utilisée par le message de demande entrant et si le ContentType de la demande correspond au ContentType d'une TransformationRule, le programme d'écoute utilise la TransformationRule pour extraire le MimeType en vue de la conversion du message de demande en un objet métier de demande. Le programme d'écoute tente de trouver l'occurrence TransformationRule exacte en fonction de la valeur ContentType (`text/xml`, par exemple) dans la propriété URLsConfiguration de l'URL demandée.
  2. En cas d'échec, le programme d'écoute tente de trouver une TransformationRule s'appliquant à plusieurs ContentType sous l'URL de la demande URL (`*/*`, par exemple).
  3. Si les étapes précédentes ne parviennent pas à identifier le MimeType, la valeur de ContentType sera utilisée comme MimeType pour appeler le gestionnaire de données et convertir le message de demande en objet métier de demande.
- Le programme d'écoute détermine le ContentType en extrayant le type/sous-type de l'entête Content-Type du message HTTP entrant.
- Le programme d'écoute détermine l'en-tête Content-Type à partir de celui du message HTTP entrant.

En cas d'appel asynchrone de la collaboration, le programme d'écoute livre l'objet métier de demande au courtier d'intégration et répond au client avec le code état HTTP `202 Accepted`. Cela termine le traitement du programme d'écoute.

En cas d'appel synchrone, le programme d'écoute appelle la collaboration de façon synchrone. La collaboration répond avec un objet métier de réponse.

Le tableau 28 récapitule la priorité des règles qu'utilise le programme pour déterminer l'en-tête Charset, MimeType, ContentType et Content-Type des messages de réponse en cas de présence de **charge**.

Remarque : En cas d'**absence de charge** avec les messages de réponse, consultez le tableau 29, à la page 55, qui récapitule l'ordre de priorité des règles qu'utilise le programme d'écoute pour identifier l'en-tête Charset, MimeType, ContentType et Content-Type.

Tableau 28. Règles de traitement des messages de réponse synchrones sortants en présence de charge

Priorité	Charset	MimeType	ContentType	Entête Content-Type
1	Entête Content-Type ConfigMO du protocole	Propriété MimeType dans le TLO	Entête Content-Type ConfigMO du protocole	Entête Content-Type ConfigMO du protocole
2	Valeur de la propriété Charset dans le TLO	Mimetype du message de demande, uniquement si les propriétés ContentType de la demande et de la réponse sont identiques.	ContentType du message de demande	En-tête Content-Type Construct utilisant ContentType et Charset
3	Mimetype du message de demande, uniquement si les propriétés ContentType de la demande et de la réponse sont identiques.	Valeur par défaut : ContentType		
4	Si le ContentType est `text/*`, la valeur par défaut est ISO-8859-1. Autrement, charset ne sera pas utilisé.			

Tableau 29. Règles de traitement des messages de réponse synchrones sortants en présence de charge de demandes

Priorité	Charset	MimeType	ContentType	Entête Content-Type
1	Entête Content-Type ConfigMO du protocole	Propriété MimeType dans le TLO	Entête Content-Type ConfigMO du protocole	Entête Content-Type ConfigMO du protocole
2	Valeur de la propriété Charset dans le TLO	Mimetype du message de demande, uniquement si les propriétés ContentType de la demande et de la réponse sont identiques.	ContentType du message de demande	En-tête Content-Type Construct utilisant ContentType et Charset
3	Mimetype du message de demande, uniquement si les propriétés ContentType de la demande et de la réponse sont identiques.	Valeur par défaut : ContentType		
4	Si le ContentType est `text/*`, la valeur par défaut est ISO-8859-1. Autrement, charset ne sera pas utilisé.	Echec de la réponse		

Comme le montre le tableau 28,

- Le programme d'écoute détermine le Charset du message de réponse selon ces règles :
  1. Si Charset est indiqué dans le Protocol Config MO de l'objet métier de réponse, sa valeur est utilisée.
  2. Si aucune valeur Charset n'est indiquée dans l'entête Protocol Config MO de l'objet métier de réponse, le programme d'écoute vérifie si Charset est indiqué dans le TLO.
  3. Si aucun Charset n'est indiqué dans le TLO et que la réponse possède le même ContentType que la demande, le Charset de la demande sera employé pour la réponse.
  4. Si les opérations précédentes n'ont pu déterminer la valeur Charset de la réponse et si la partie type du ContentType du message est `text` avec n'importe quel sous-type (par exemple,`text/xml`, `text/plain`, etc.), le programme d'écoute utilise ISO-8859-1 comme valeur de Charset par défaut. Autrement, la valeur de Charset ne sera pas utilisée.
- Le programme d'écoute détermine le MimeType du message de réponse selon ces règles :
  1. Attribut MimeType du TLO
  2. Si l'attribut MimeType de TLO et si le ContentType de la demande et de la réponse coïncide, le programme d'écoute utilise le MimeType de la demande pour le message de réponse.
  3. Dans le cas contraire, le programme d'écoute utilise la valeur de ContentType comme MimeType.
- Le programme d'écoute détermine le ContentType du message de réponse selon ces règles :
  1. Si l'entête Content-Type est indiqué dans le Protocol Config MO de l'objet métier de réponse, la partie type/sous-type de cet entête sera employé comme ContentType.
  2. Si l'entête Content-Type n'est pas indiqué dans le Protocol Config MO de l'objet métier de réponse, le programme d'écoute construit un en-tête Content-Type à l'aide des ContentType et Charset déterminés (si le Charset a été déterminé pour le message de réponse).

Le programme d'écoute traite HTTP Protocol Config MO. La collaboration doit vérifier que les valeurs d'entête transmises à HTTP Protocol Config MO sont correctes dans le contexte de l'événement demande-réponse. Le programme d'écoute complète les entêtes standard et les propriétés personnalisées selon les règles suivantes :

1. Le programme d'écoute inspecte chaque élément de HTTP Protocol Config MO afin de ne pas tenir compte d'attributs spéciaux (tels qu'ObjectEventId).
2. Chaque entête non vide est copié dans le message sortant et tout traitement supplémentaire (de l'entête Content-Type, par exemple) peut avoir lieu.
3. Notez qu'avec cette approche, il se peut que le programme d'écoute définisse des entêtes non standard dans le message et ne vérifie pas la structure logique ou sémantique de celui-ci.
4. Si l'attribut UserDefinedProperties de HTTP Protocol Config MO contient une ou plusieurs propriétés personnalisées, le programme d'écoute les ajoute à la section Entity Headers (dernière section d'entêtes du message). Pour plus d'informations sur les propriétés personnalisées, voir «Propriétés définies par l'utilisateur pour le traitement des événements», à la page 26.

Remarque : La présence de l'un des entêtes suivants : Connection, Trailer, Transfer-Encoding, Content-Encoding, Content-Length, Content-MD5, Content-Range, dans HTTP Protocol Config MO compromet généralement la validité ou le traitement du message.

Le programme d'écoute appelle le gestionnaire de données pour convertir en message de réponse l'objet métier de réponse retourné par la collaboration.

Le programme d'écoute livre le message de réponse au client et inclut un code état HTTP `200 OK`. Si la collaboration renvoie un objet métier d'erreur, celui-ci est converti en message d'erreur. Ce message d'erreur est envoyé au client avec un code erreur `500 Internal Server Error HTTP`.

Le programme d'écoute ferme alors la connexion, ce qui libère l'unité d'exécution qui a traité l'événement.

Fonctions de traitement non prises en charge par le programme d'écoute de protocole HTTP

Le programme d'écoute de protocole HTTP ne prend pas en charge ce qui suit :

- Mise en cache : Le programme d'écoute de protocole n'implémente pas de fonctions de mise en cache comme défini dans les spécifications HTTP (RFC2616)
- Proxy : Le programme d'écoute de protocole n'implémente pas de fonctions proxy comme défini dans les spécifications HTTP (RFC2616)
- Connexions persistantes : Le programme d'écoute de protocole ne prend pas en charge de connexions persistantes comme défini dans les spécifications HTTP (RFC2616). Le programme d'écoute de protocole suppose que la portée de chaque connexion HTTP est une demande de client simple et referme la connexion quand la demande de service est traitée. Le programme d'écoute de protocole ne tente pas de réutiliser la connexion d'un appel de service à un autre.
- Redirections : Le programme d'écoute de protocole ne prend pas en charge les redirections.
- Transfert de fichiers volumineux : Le programme d'écoute de protocole ne peut être utilisé pour les transferts de fichiers volumineux. Vous pouvez envisager de transmettre les fichiers volumineux par référence.
- Gestion d'état : Le programme d'écoute de protocole ne prend pas en charge le mécanisme de gestion d'état HTTP décrit par RFC2965.
- Cookies : Le programme d'écoute de protocole ne prend pas en charge les cookies.

Traitement du protocole d'écoute HTTPS à l'aide de sockets sécurisées

Le traitement du programme d'écoute de protocole HTTPS est similaire à celui du programme d'écoute de protocole HTTPS hormis que HTTPS utilise des sockets sécurisées. Pour plus d'informations, voir «SSL», à la page 64.

Persistance et livraison d'événements

La persistance d'événements dépend du protocole :

- **Programme d'écoute de protocole HTTP** aucune persistance et, par conséquent, aucune livraison garantie

- **Programme d'écoute de protocole HTTPS** aucune persistance et, par conséquent, aucune livraison garantie

Séquençage d'événements

Le connecteur peut livrer des événements dans n'importe quel ordre.

Déclenchement d'événement

Le mécanisme de déclenchement d'événements dépend de la configuration du programme d'écoute de protocole.

- **Programme d'écoute de protocole HTTP** L'écoute s'effectue au niveau des demandes de connexions ServerSocket for HTTP
- **Programme d'écoute de protocole HTTPS** L'écoute s'effectue sur une couche ServerSocket sécurisées pour des demandes de connexion HTTPS

Remarque : Le connecteur ne fait pas la différence entre les opérations Create, Update, Retrieve et Delete. Il en va de même pour de tels événements.

Détection d'événements

La détection d'événements est effectuée par chaque programme d'écoute de protocole. Le mécanisme de détection d'événements dépend essentiellement du transport et de la façon dont vous configurez les propriétés spécifiques de connecteur pour chaque programme d'écoute. Pour plus d'informations sur ces propriétés, voir «Propriétés de configuration spécifiques au connecteur», à la page 67.

Etat d'événement

L'état d'événement est géré par le programme d'écoute de protocole et dépend du transport mais aussi de la façon dont vous configurez le programme d'écoute.

- **Programme d'écoute de protocole HTTP** Par nature, HTTP est synchrone et non persistant. De même, l'état d'événement n'est pas conservé.
- **Programme d'écoute de protocole HTTPS** Par nature, HTTP est synchrone et non persistant. De même, l'état d'événement n'est pas conservé.

Récupération d'événement

La récupération d'événement est gérée par le programme d'écoute de protocole et dépend du transport mais aussi de la façon dont vous configurez le programme d'écoute.

- **Programme d'écoute de protocole HTTP** Les événements sont récupérés en extrayant des demandes HTTP de la connexion.
- **Programme d'écoute de protocole HTTPS** Les événements sont récupérés en extrayant des demandes HTTP de la connexion.

Archivage d'événement

L'archivage d'événement est géré par le programme d'écoute de protocole et dépend du transport mais aussi de la façon dont vous configurez le programme d'écoute.

- **Programme d'écoute de protocole HTTP** En raison de la nature synchrone non persistante du transport, l'archivage n'a pas lieu.
- **Programme d'écoute de protocole HTTPS** En raison de la nature synchrone non persistante du transport, l'archivage n'a pas lieu.

Récupération d'événements

La reprise d'événement est gérée par le programme d'écoute de protocole et dépend du transport mais aussi de la façon dont vous configurez le programme d'écoute.

- **Programme d'écoute de protocole HTTP** En raison de la nature non persistante du transport, la récupération d'événements n'a pas lieu.
- **Programme d'écoute de protocole HTTPS** En raison de la nature non persistante du transport, la récupération d'événements n'a pas lieu.

Traitement des demandes

La fonction de traitement des demandes du connecteur permet à une collaboration d'appeler un service HTTP. Vous devez configurer le connecteur et ses composants de traitement de la demande : la structure du gestionnaire de protocole et les gestionnaires de protocole.

Lors de l'exécution, le connecteur reçoit des demandes émanant de collaboration sous la forme d'objets métier. Les objets métier — demandes et, le cas échéant, objets métier de réponse et d'erreur — sont contenus par le TLO générés par une collaboration qui est configurée pour utiliser les services HTTP. Le TLO et ses objets métier enfants contiennent des attributs et des ASI indiquant le mode de traitement(synchrone ou asynchrone), le type mime du gestionnaire de données, le gestionnaire de données à utiliser ainsi que l'adresse de la cible. A l'aide de ces informations, le gestionnaire de protocole appelle une instance du gestionnaire de données, convertit l'objet métier de demande en message de demande, et appelle le service HTTP cible. Si le mode est synchrone, le gestionnaire de protocole appelle à nouveau le gestionnaire de données pour convertir le message de réponse en objet métier qu'il renvoie à la collaboration.

En réponse à un message de demande, le connecteur peut recevoir l'un des messages suivants de la part du partenaire d'échanges éloigné :

- Un message de réponse contenant des données
- Un message de réponse contenant des informations d'erreur

Les gestionnaires de protocole jouent un rôle clé dans le traitement des demandes.

Gestion de protocole

Une collaboration peut appeler un service HTTP sur HTTP ou HTTPS. Le connecteur possède un seul gestionnaire de protocole et un canal correspondant : un gestionnaire de protocole HTTP-HTTPS pour l'appel de services HTTP et HTTPS

La structure de gestionnaire de protocole gère le gestionnaire de protocole en le chargeant lors du démarrage. Quand le connecteur reçoit un objet métier de demande, l'unité d'exécution de demande (notez que chaque demande de collaboration est accompagnée de son unité d'exécution) appelle la structure de gestionnaire de protocole pour traiter la demande.

La structure de gestionnaire de protocole lit les informations ASI d'attribut Handler des TLO pour déterminer quel gestionnaire de protocole utiliser. Applicant une série de règles (voir «HTTP-HTTPS traitement du gestionnaire de protocole», à la page 60), le gestionnaire de protocole appelle un gestionnaire de

données pour convertir l'objet métier de demande en message de demande. Le gestionnaire de protocole intègre le message de demande dans le message du transport HTTP(S).

Le gestionnaire de protocole lit ensuite l'attribut Destination du Protocol Config MO de l'objet métier de demande pour déterminer l'adresse cible. La propriété Method de l'attribut HTTP Config MO indiquera la méthode HTTP à utiliser lors de l'exécution. La méthode par défaut est POST. Le gestionnaire de protocole appelle ensuite le service HTTP cible avec le message de demande.

Lisant les ASI de `ws_mode TLO`, le gestionnaire de protocole détermine si le mode de traitement est synchrone ou asynchrone. Si ces informations ASI sont réglées sur `asynch`, le traitement du gestionnaire de protocole est terminé. Autrement, le gestionnaire de protocole attend un message de réponse. Si un message de réponse arrive, le gestionnaire de protocole extrait les entêtes de protocole et la charge. Il appelle ensuite le gestionnaire de données (indiqué par l'attribut MimeType TLO) pour convertir le message en objet métier d'erreur ou de réponse. Toujours à l'aide de Protocol Config MO, le gestionnaire de protocole définit les entêtes de protocole dans l'objet métier. Le gestionnaire de protocole renvoie ensuite l'objet métier de réponse ou d'erreur à la collaboration.

Selon la configuration du connecteur, un ou plusieurs gestionnaires peuvent être reliés au connecteur. Les propriétés spécifiques au connecteur permettent de configurer des gestionnaires de protocole.

HTTP-HTTPS traitement du gestionnaire de protocole

Le gestionnaire de protocole HTTP-HTTPS s'exécute conformément à ce qui est décrit dans «Gestion de protocole», à la page 59 et avec les exceptions indiquées dans cette section. La figure 16 représente le gestionnaire de protocole HTTP-HTTPS fonctionnant en mode synchrone.

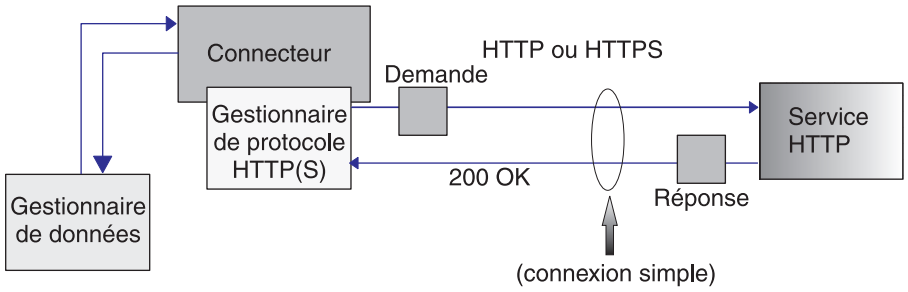


Figure 16. Gestionnaire de protocole HTTP-HTTPS : traitement des demandes synchrones

La figure 17 représente le gestionnaire de protocole HTTP-HTTPS d'un processus de demandes asynchrones

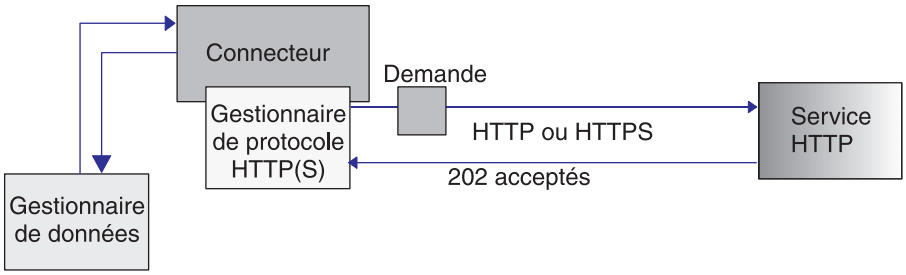


Figure 17. Gestionnaire de protocole HTTP-HTTPS : traitement des demandes asynchrones

Remarque : Cette section décrit la gestion de protocole HTTP uniquement.

Le gestionnaire de protocole HTTP-HTTPS utilise les ASI de niveau objet (`cw_mo_http`) de l'objet métier de demande pour déterminer le Protocol Config MO. Le gestionnaire de protocole HTTP-HTTPS détermine l'URL du service HTTP cible en lisant l'attribut Destination de HTTP Protocol Config MO. Si l'URL est manquante ou incomplète, le gestionnaire de protocole fait échouer l'appel du service. Pour plus d'informations sur HTTP Protocol Config MO et ses attributs, voir «HTTP Protocol Config MO pour le traitement des demandes», à la page 36.

Le gestionnaire de protocole HTTP-HTTPS appelle le service HTTP à l'aide du message de demande renvoyé par le gestionnaire de données. Le comportement du gestionnaire de protocole est conforme aux propriétés de configuration de connecteur HTTP ayant été indiquées, le cas échéant. Le gestionnaire de protocole HTTP-HTTPS lit la réponse éventuellement reçue.

Le tableau 30 récapitule la priorité des règles qu'utilise le gestionnaire de protocole HTTP-HTTPS pour déterminer l'en-tête Charset, MimeType, ContentType et Content-Type des messages de demande sortants.

Tableau 30. Règles de traitement du gestionnaire de protocole HTTP-HTTPS pour les messages sortants

Priorité	Charset	MimeType	ContentType	Entête Content-Type
1	Entête Content-Type de Protocol Config MO	Propriété MimeType de l'attribut TLO	Entête Content-Type de Protocol Config MO	Entête Content-Type de Protocol Config MO
2	Propriété Charset de l'attribut TLO	Valeur par défaut : ContentType		En-tête Content-Type Construct utilisant ContentType et Charset
3	Si le ContentType est `text/*`, la valeur par défaut est ISO-8859-1. Autrement, charset ne sera pas utilisé.			

Comme le montre le tableau 30 :

- Le gestionnaire de protocole HTTP-HTTPS détermine le Charset du message de demande selon ces règles :
  1. Si Charset est indiqué dans les entêtes Protocol Config MO de l'objet métier de demande, sa valeur est utilisée.
  2. Si Charset n'est pas déterminé par l'étape précédente, le gestionnaire de protocole tente d'extraire le Charset de l'attribut TLO.

3. Si l'opération décrite à l'étape précédente échoue, le ContentType représenté dans le tableau 31 permet de déterminer le Charset :

Tableau 31. Charsets du traitement des demandes par défaut

ContentType	Charset par défaut
`text/*`	ISO-8859-1 Pour plus d'informations, voir RFC2616
`application/*`	Aucune valeur par défaut
Tous les autres	Aucune valeur par défaut

4. S'il a été déterminé à l'étape précédente, le Charset est réglé sur le gestionnaire de données.
5. Le gestionnaire de données est appelé avec les API de tableau Stream ou Byte selon la structure de données requise pour l'écriture de la demande.

- Le gestionnaire de protocole HTTP-HTTPS détermine le MimeType de la demande selon ces règles :
  1. Attribut MimeType du TLO.
  2. Si l'attribut MimeType du TLO est manquant, le gestionnaire de protocole utilise le ContentType pour déterminer le MimeType.
- Le gestionnaire de protocole HTTP-HTTPS détermine le ContentType du message de demande selon ces règles :
  - Si l'entête Content-Type est indiqué dans Protocol Config MO de l'objet métier de demande, la partie type/sous-type de cet entête sera employé comme ContentType.
- Le gestionnaire de protocole HTTP-HTTPS détermine le Content-Type du message de demande selon ces règles :
  - Si l'entête Content-Type figure dans le Protocol Config MO de l'objet métier de demande, sa valeur est réglée sur le message sortant.

Le tableau 32 récapitule la priorité des règles qu'utilise le gestionnaire lors de la détermination de l'en-tête Charset, MimeType, ContentType et Content-Type des messages de réponse.

Tableau 32. Règles de traitement du message de réponse synchrone entrant mises en oeuvre par le gestionnaire de protocole HTTP(s)

Priorité	Charset	MimeType	ContentType	Entête Content-Type
1	Entête Content-Type du message HTTP entrant	TransformationMap de message ou objet métier enfant EmptyResponseRule (si la réponse est vide) dans le Protocol Config MO de l'objet métier de réponse	Entête Content-Type du message HTTP entrant	Entête Content-Type du message HTTP entrant
2	TransformationMap de message ou objet métier enfant EmptyResponseRule (si la réponse est vide) dans le Protocol Config MO de l'objet métier de réponse	Mimetype du message de demande, uniquement si les propriétés ContentType de la demande et de la réponse sont identiques.		

Tableau 32. Règles de traitement du message de réponse synchrone entrant mises en oeuvre par le gestionnaire de protocole HTTP(s) (suite)

3	Mimetype du message de demande, uniquement si les propriétés ContentType de la demande et de la réponse sont identiques.	Propriété MimeType dans TLO		
4	Propriété Charset dans TLO.	Valeur par défaut : ContentType		
5	Si le Content-Type est `text/*`, la valeur par défaut est ISO-8859-1. Autrement, Charset ne sera pas utilisé.			

Comme le montre le tableau 32 :

- Le gestionnaire de protocole détermine le Charset du message de réponse synchrone selon les règles suivantes :
  1. Si le paramètre Charset est réglé dans l'entête Content-Type du message de réponse entrant, le gestionnaire de protocole utilise la valeur Charset à définir sur le gestionnaire de données.
  2. Si l'entête de message de réponse ne contient pas de valeur Charset, la valeur Charset définie sur la collaboration à partir du MessageTranformationMap de Protocol Config MO de demande du TLO.
  3. Si aucun Charset n'est indiqué dans le MessageTransformationMap pour la demande donnée et que la réponse possède le même ContentType que la demande, le Charset de la demande sera employé pour la réponse.
  4. Si les étapes précédentes ne parviennent pas à déterminer une valeur Charset, le gestionnaire de protocole tente de lire l'attribut Charset du TLO.
  5. Si la valeur Charset n'est pas obtenue à l'aide des méthodes décrites aux étapes précédentes et si le type ContentType du message est `text` avec un sous-type (par exemple, `text/xml`, `text/plain`, etc.), la valeur par défaut est ISO-8859-1. Autrement, la valeur de charset ne sera pas utilisée.
- Le gestionnaire de protocole détermine le MimeType du message de réponse synchrone selon les règles suivantes :
  1. Le gestionnaire de protocole tente tout d'abord d'extraire le MimeType du MessageTransformationMap de Protocol Config MO de demande du TLO. Plus précisément, le gestionnaire de protocole tente de trouver une occurrence ContentType exacte dans la MTM afin d'extraire MessageTransformationRule et d'utiliser la valeur de propriété MimeType correspondante. Autrement, le gestionnaire de protocole recherche un MessageTransformationRule qui s'applique à plusieurs ContentType (ContentType est égal à */*).
  2. Si le MimeType n'est pas déterminé à l'aide de MessageTransformationMap, le gestionnaire de protocole utilise le MimeType de la demande pour la réponse si et seulement si les ContentTypes de la demande et de la réponse coïncident.
  3. Si le MimeType ne peut être extrait lors des étapes précédentes, le gestionnaire de protocole utilise l'attribut MimeType du TLO.

4. Si toutes les étapes précédentes échouent, le gestionnaire de protocole utilise le ContentType pour définir le MimeType.

- Le gestionnaire détermine le ContentType en extrayant le type/sous-type de l'entête Content-Type du message HTTP entrant.

Le gestionnaire traite HTTP Protocol Config MO. La collaboration doit vérifier que les valeurs d'entête transmises à HTTP Protocol Config MO sont correctes dans le contexte de l'événement demande-réponse. Le gestionnaire complète les entêtes standard et les propriétés personnalisées selon les règles suivantes :

1. Le gestionnaire inspecte chaque élément de HTTP Protocol Config MO afin de ne pas tenir compte d'attributs spéciaux (tels qu'ObjectEventId).
2. Chaque entête non vide est copié dans le message sortant et tout traitement supplémentaire (de l'entête Content-Type, par exemple) peut avoir lieu.
3. Notez qu'avec cette approche, il se peut que le gestionnaire définisse des entêtes non standard dans le message et ne garantit pas la structure logique ou sémantique de celui-ci.
4. Si l'attribut UserDefinedProperties de HTTP Protocol Config MO contient une ou plusieurs propriétés personnalisées, le gestionnaire les ajoute à la section Entity Headers (dernière section d'entêtes du message). Pour plus d'informations sur les propriétés personnalisées, voir «Propriétés définies par l'utilisateur pour le traitement des demandes», à la page 38.

Remarque : La présence de l'un des entêtes suivants : Connection, Trailer, Transfer-Encoding, Content-Encoding, Content-Length, Content-MD5, Content-Range, dans HTTP Protocol Config MO compromet généralement la validité ou le traitement des messages.

SSL

Cette section aborde l'implémentation d'une fonction SSL par le connecteur. Pour plus d'informations, voir la documentation SSL JSSE. Cette section suppose que vous soyez familiarisé avec la technologie SSL.

JSSE

Le connecteur utilise JSSE pour prendre en charge HTTPS et SSL. IBM JSSE est fourni avec le connecteur. Pour activer cette fonction, assurez-vous que l'un des fichiers `java.security`,installés avec le connecteur, comprend l'entrée suivante :

```
security.provider.5=com.ibm.jsse.IBMJSSEProvider
```

Notez que `java.security` réside dans le répertoire `$ProductDir\lib\security` de l'installation de votre connecteur. Le connecteur utilise la valeur de la propriété du connecteur JavaProtocolHandlerPackages pour définir la propriété système java.protocol.handler.pkgs. Notez que pour l'outil IBM JSSE fourni avec le connecteur, la valeur de cette propriété doit être `com.ibm.net.ssl.internal.www.protocol`.

La propriété de configuration JavaProtocolHandlerPackages est affectée par défaut à cette valeur. Cependant, si votre ordinateur possède une propriété système java.protocol.handler.pkgs avec une valeur non vide, le connecteur l'écrasera uniquement si la propriété de connecteur JavaProtocolHandlerPackages est également définie.

Lors de l'initialisation, le connecteur désactive toutes les suites de chiffrement anonyme prises en charge par JSSE.

KeyStore et TrustStore

Pour SSL avec le connecteur, vous devez configurer des magasins de clés et des magasins de certificats. Aucun outil n'est fourni pour configurer magasins de clés, certificats et génération de clés. Pour effectuer ces tâches, vous devez utiliser des outils tiers.

Propriétés SSL

Vous pouvez indiquer les propriétés propres au connecteur SSL suivantes :

- SSLVersion
- SSLDebug
- KeyStore
- KeyStoreAlias
- KeyStorePassword
- TrustStore
- TrustStorePassword

Notez que ces propriétés s'appliquent à une instance de connecteur. Le même ensemble de valeurs de propriété SSL est employé par tous les programmes d'écoute de protocole HTTPS branchés au connecteur et par le gestionnaire de protocole HTTP-HTTPS pour chaque instance de connecteur. Pour plus d'informations sur la configuration de HTTPS/SSL, voir Annexe D, «Configuration de HTTPS/SSL», à la page 143.

SSL et le programme d'écoute de protocole HTTPS

Pour utiliser le programme d'écoute de protocole HTTPS, vous devez utiliser des propriétés spécifiques du connecteur SSL. Les valeurs que vous affectez à ces propriétés doivent refléter vos exigences SSL :

- **SSLVersion** Assurez-vous que la version SSL (SSLVersion) à utiliser est prise en charge par JSSE.
- **KeyStore** Du fait que le programme d'écoute de protocole HTTPS joue le rôle de serveur dans les communications SSL, vous devez indiquer le magasin de clés. Le programme d'écoute utilise le magasin de clés indiqué dans la propriété de configuration SSL->KeyStore. La valeur de cette propriété doit être le chemin d'accès complet au fichier du magasin de clés. Assurez-vous que le magasin de clés possède une paire de clés (clé privée et clé publique) pour le connecteur. L'alias de la clé privée doit être indiqué comme propriété SSL->KeyStoreAlias. Vous devez indiquer le mot de passe requis pour accéder au magasin de clés (propriété SSL-> KeyStorePassword). Vérifiez également que le mot de passe requis pour accéder au magasin de clés et que la clé privée (dans le magasin de clés) sont identiques. Enfin, vous devez distribuer le certificat numérique du connecteur à vos clients pour qu'ils puissent authentifier ce dernier.
- **TrustStore** Si vous souhaitez que le programme d'écoute de protocole HTTPS authentifie des clients, vous devez activer l'authentification client. Pour cela, réglez la propriété SSL ->UseClientAuth sur `true`. Vous devez également indiquer :
  - l'emplacement de votre truststore comme valeur de la propriété de configuration SSL->TrustStore
  - le mot de passe requis pour accéder à votre truststore comme valeur de la propriété SSL->TrustStorePassword

  Assurez-vous que votre truststore contient le certificat numérique de vos clients. Les certificats numériques utilisés par vos clients peuvent être auto-signés ou

émis par une autorité de certification (CA). Notez que si votre truststore fait confiance au certificat racine du CA, JSSE authentifiera tous les certificats numériques émis par cette autorité.

Pour plus d'informations sur la configuration de HTTPS/SSL, voir Annexe D, «Configuration de HTTPS/SSL», à la page 143.

SSL et le gestionnaire de protocole HTTP-HTTPS

Si vous utilisez SSL avec le gestionnaire de de protocole HTTP-HTTPS, vous devez opter pour des propriétés spécifiques du connecteur SSL. Les valeurs que vous affectez à ces propriétés doivent refléter les exigences HTTPS/SSL de votre fournisseur HTTP :

- **SSLVersion** Assurez-vous que la version SSL (SSLVersion) à utiliser est prise en charge par votre fournisseur et par JSSE.
- **TrustStore** Du fait que le gestionnaire de protocole HTTP-HTTPS jour le rôle de client dans les communications SSL, vous devez configurer un truststore. Le gestionnaire utilise le truststore indiqué dans la propriété de configuration SSL->Truststore. La valeur de cette propriété doit être le chemin d'accès complet au fichier du truststore. Vous devez indiquer le mot de passe requis pour accéder au truststore(propriété SSL-> TrustStorePassword). Assurez-vous que votre truststore contient le certificat numérique de votre fournisseur. Les certificats numériques utilisés par votre fournisseur peuvent être auto-signés ou émis par une autorité de certification. Notez que si votre truststore fait confiance au certificat racine du CA, JSSE authentifiera tous les certificats numériques émis par cette autorité.
- **KeyStore** Si votre fournisseur de services HTTP requiert une authentification client, vous devez configurer un magasin de clés. Le gestionnaire de protocole HTTP-HTTPS utilise le magasin de clés indiqué dans la propriété de configuration SSL->KeyStore. Cette valeur doit être le chemin d'accès complet au fichier du magasin de clés. Assurez-vous que ce magasin de clés possède une paire de clés (clé privée et clé publique) pour le connecteur. L'alias de la clé privée doit être indiqué comme propriété SSL->KeyStoreAlias. Le mot de passe requis pour accéder au magasin de clés doit être indiqué comme propriété SSL->KeyStorePassword. Enfin, vérifiez que le mot de passe requis pour accéder au magasin de clés et que la clé privée (dans le magasin de clés) sont identiques. Vous devez distribuer le certificat numérique du connecteur à votre fournisseur de services HTTP pour authentification.

Pour plus d'informations sur la configuration de HTTPS/SSL, voir Annexe D, «Configuration de HTTPS/SSL», à la page 143.

Configuration du connecteur

Après l'installation des fichiers de connecteur sur votre système à l'aide du programme d'installation, vous devez définir les propriétés de configuration standard et spécifiques d'application pour le connecteur.

Définition des propriétés de configuration

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 System Manager (SM) avant d'exécuter le connecteur.

Propriétés standard de configuration

Les propriétés standard de configuration fournissent des informations utilisées par tous les connecteurs. Voir l'Annexe A, «Propriétés de configuration standard pour

les connecteurs», à la page 87 pour plus d'informations sur ces propriétés. Le tableau donne des informations spécifiques à ce connecteur et concernant les propriétés de configuration de l'annexe.

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

Du fait que ce connecteur ne prend en charge qu'InterChange Server (ICS) comme courtier d'intégration, les seules propriétés de configuration appropriées concernent ICS.

Vous devez définir au moins les propriétés standard de configuration du connecteur :

- AgentTraceLevel
- ApplicationName
- ControllerTraceLevel
- DeliveryTransport

Propriétés de configuration spécifiques au connecteur

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

Le tableau 33 dresse la liste des propriétés de configuration spécifiques au connecteur. Pour obtenir une explication des propriétés, voir les sections suivantes. Certaines propriétés contiennent elles-mêmes d'autres propriétés. Le caractère + indique la position de l'entrée dans la hiérarchie des propriétés.

Tableau 33. Propriétés de configuration spécifiques au connecteur

Nom	Valeurs possibles	Valeur par défaut	Obligatoire
DataHandlerMetaObjectName	*Nom du méta-objet du gestionnaire de données*	`MO_DataHandler_ Default`	Oui
UseDefaults	*UseDefaults*	`false`	Non
JavaProtocolHandlerPackages	*Packages du gestionnaire de protocole Java valides*	`com.ibm.net.ssl. internal.www.protocol`	Non
ProtocolHandlerFramework	*Il s'agit d'une propriété hiérarchique ne possédant pas de valeur*	Néant	Non
+ProtocolHandlers	*Il s'agit d'une propriété hiérarchique ne possédant pas de valeur*		Non
++Handler1	Il s'agit d'une propriété hiérarchique. Pour plus d'informations sur ses sous-propriétés, voir «Handler1», à la page 69.		Oui
+++Protocol	http		Oui
+++HandlerSpecific	Chaque gestionnaire de protocole peut posséder des propriétés obligatoires.		

Tableau 33. Propriétés de configuration spécifiques au connecteur (suite)

Nom	Valeurs possibles	Valeur par défaut	Obligatoire
ProtocolListenerFramework	Il s'agit d'une propriété hiérarchique ne possédant pas de valeur.		Non
+WorkerThreadCount	Entier supérieur ou égal à 1 définissant le nombre d'unités d'exécution disponibles du programme d'écoute.	`10`	Non
+RequestPoolSize	Entier supérieur à WorkerThreadCount indiquant la taille du pool de ressources.	`20`	Non
+ProtocolListeners	Il s'agit d'une propriété hiérarchique ne possédant pas de valeur		
++Listener1	Programme d'écoute de protocole portant un nom unique		Oui
+++Protocol	`http` ou `https`		Oui
+++ListenerSpecific	Propriétés uniques du programme d'écoute ou requises par celui-ci. Voir «ListenerSpecific», à la page 71.		
ProxyServer	Il s'agit d'une propriété hiérarchique ne possédant pas de valeur		Non
+HttpProxyHost	Nom d'hôte du serveur proxy HTTP		Non
+HttpProxyPort	Numéro de port du serveur proxy HTTP	`80`	Non
+HttpNonProxyHosts	Hôte(s) HTTP nécessitant une connexion directe		Non
+HttpsProxyHost	Nom d'hôte du serveur proxy HTTPS		Non
+HttpsProxyPort	Numéro de port du serveur proxy HTTPS	`443`	Non
+HttpsNonProxyHosts	Hôte(s) HTTPS nécessitant une connexion directe		Non
+SocksProxyHost	Nom du serveur proxy Socks		Non
+SocksProxyPort	Port du serveur proxy Socks		Non
+HttpProxyUsername	Nom d'utilisateur du serveur proxy Http		Non
+HttpProxyPassword	Mot de passe du serveur proxy Http		Non
+HttpsProxyUsername	Nom d'utilisateur du serveur proxy Https		Non
+HttpsProxyPassword	Mot de passe du serveur proxy Https		Non
SSL	Il s'agit d'une propriété hiérarchique ne possédant pas de valeur		Non
+SSLVersion	`SSL`, `SSLv2`, `SSLv3`, `TLS`, `TLSv1`	SSL	Non
+SSLDebug	`true`, `false`	`false`	Non
+KeyStoreType	N'importe quel type de magasin de clés correct	`JKS`	Non
+KeyStore	Chemin d'accès du fichier KeyStore		Non

Tableau 33. Propriétés de configuration spécifiques au connecteur (suite)

Nom	Valeurs possibles	Valeur par défaut	Obligatoire
+KeyStorePassword	Mot de passe de clé privée dans KeyStore		Non
+KeyStoreAlias	Alias pour une paire de clés dans KeyStore		Non
+TrustStore	Chemin d'accès au fichier TrustStore		Non
+TrustStorePassword	Mot de passe de TrustStore		Non
+UseClientAuth	`true false`	`false`	Non

DataHandlerMetaObjectName : Il s'agit du nom du méta-objet utilisé par le gestionnaire de données pour définir les propriétés de configuration.

Default = `MO_DataHandler_Default`.

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

Valeur par défaut = `false`.

JavaProtocolHandlerPackages : La valeur de cette propriété définit les packages Java Protocol Handler. Le connecteur utilise la valeur de cette propriété pour définir la propriété système java.protocol.handler.pkgs.

Default = `com.ibm.net.ssl.internal.www.protocol`.

ProtocolHandlerFramework : La structure Protocol Handler Framework utilise cette propriété pour charger et configurer ses gestionnaires de protocole. Il s'agit d'une propriété hiérarchique ne possédant pas de valeur.

Valeur par défaut = none.

ProtocolHandlers : Cette propriété hiérarchique ne possède pas de valeur. Ses enfants de premier niveau représentent des gestionnaires de protocole discrets.

Valeur par défaut = none.

Handler1 : Nom d'un gestionnaire de protocole HTTP-HTTPS. Il s'agit d'une propriété hiérarchique. A la différence des programmes d'écoute, les gestionnaires de protocole ne peuvent être dupliqués et on ne compte qu'un gestionnaire par protocole. Le tableau 34 représente les sous-propriétés du gestionnaire de protocole HTTP-HTTPS. Le caractère + indique la position de l'entrée dans la hiérarchie des propriétés.

Tableau 34. Propriétés de configuration du gestionnaire de protocole HTTP-HTTPS

Nom	Valeurs possibles	Valeur par défaut	Obligatoire
++HTTPHTTPSHandler	Il s'agit d'une propriété hiérarchique ne possédant pas de valeur.		Oui

Tableau 34. Propriétés de configuration du gestionnaire de protocole HTTP-HTTPS (suite)

Nom	Valeurs possibles	Valeur par défaut	Obligatoire
+++Protocol	Type du protocole implémenté par le gestionnaire. Pour HTTP et HTTPS, la valeur est `http`. **Remarque :** Si vous n'indiquez pas de valeur pour cette propriété, le connecteur n'initialisera pas le gestionnaire de protocole.	`http`	Oui
+++HTTPReadTimeout	Propriété spécifique à HTTP qui indique l'intervalle du délai (en millisecondes) lors de la lecture depuis l'hôte éloigné. Si cette propriété n'est pas indiquée ou définie à 0, le protocole HTTP-HTTPS bloque indéfiniment lors de la lecture depuis l'hôte éloigné.	`0`	Non

Protocol : La valeur de cette propriété fournit le protocole que doit implémenter le gestionnaire.

HandlerSpecific : Cette propriété indique toute propriété obligatoire du gestionnaire de protocole.

ProtocolListenerFramework : La structure du programme d'écoute du protocole utilise cette propriété pour charger des programmes d'écoute de protocole. Il s'agit d'une propriété hiérarchique ne possédant pas de valeur.

WorkerThreadCount : Cette propriété, qui doit être un entier supérieur ou égal à 1, établit le nombre d'unités d'exécution de travail du programme d'écoute de protocole disponibles pour la structure du programme d'écoute de protocole. Pour plus d'informations, voir «Programmes d'écoute de protocole», à la page 50. Default = `10`.

RequestPoolSize : Cette propriété, qui doit être un entier supérieur à WorkerThreadCount, définit la taille du pool de ressources de la structure du programme d'écoute de protocole. La structure peut traiter simultanément le nombre maximal de demandes défini par la somme de WorkerThreadCount et de RequestPoolSize.

Default = `20`.

ProtocolListeners : Il s'agit d'une propriété hiérarchique ne possédant pas de valeur. Chaque enfant de premier niveau de cette propriété représente un programme d'écoute de protocole discret.

Listener1 : Nom d'un programme d'écoute de protocole. Il peut exister plusieurs programmes d'écoute de protocole. Il s'agit d'une propriété hiérarchique. Vous pouvez créer plusieurs instances de cette propriété et créer des programmes d'écoute portant chacun un nom différent. Dans ce cas, vous pouvez modifier les propriétés du programme d'écoute et non pas la propriété de protocole. Chaque nom de programme d'écoute doit être unique. Noms possibles (il ne s'agit pas de valeurs): `HTTPListener1`, `HTTPSListener1`.

Protocol : Cette propriété indique le protocole implémenté par le programme d'écoute. Valeurs possibles : `http`, `https`.

Remarque : Si vous n'indiquez pas de valeur pour cette propriété, le connecteur n'initialisera pas le programme d'écoute de protocole.

ListenerSpecific : Les propriétés spécifiques du programme d'écoute sont spécifiques au programme d'écoute de protocole indiqué ou requises par celui-ci. Par exemple, le programme d'écoute HTTP possède une propriété qui lui est spécifique, Port, qui représente le numéro de port sur lequel il contrôle les demandes. Le tableau 35 dresse la liste des propriétés spécifiques au programme d'écoute HTTP-HTTPS. Le caractère + indique la position de l'entrée dans la hiérarchie des propriétés.

Tableau 35. Propriétés de configuration spécifiques au programme d'écoute du protocole HTTP et HTTPS

Nom	Valeurs possibles	Valeur par défaut	Obligatoire
+++*HTTPListener1*	Nom unique d'un programme d'écoute de protocole HTTP. Il s'agit d'un enfant de la propriété hiérarchique ProtocolListenerFramework -> ProtocolListeners. Plusieurs programmes d'écoute peuvent exister : vous avez la possibilité de brancher des programmes d'écoute HTTP supplémentaires en créant une autre instance de cette propriété et la hiérarchie correspondante.		Oui
++++Protocol	`http` s'il s'agit d'un programme d'écoute de protocole HTTP `https` s'il s'agit d'un programme d'écoute de protocole HTTPS **Remarque :** Si vous n'indiquez pas de valeur pour cette propriété, le connecteur n'initialisera pas le programme d'écoute de protocole.		Oui
++++HostInfoValidation	FailRequest - si les informations hôte/port de la demande ne correspondent pas aux informations hôte/port du programme d'écoute, un message d'erreur est consigné et la demande échoue avec le code état 400 (demande erronée). LogError - si les informations hôte/port de la demande ne correspondent pas aux informations hôte/port du programme d'écoute, le message d'erreur est consigné et l'exécution de la demande se poursuit. Ignore - aucune validation n'est réalisée sur les informations hôte/port de la demande. Remarque : Si la propriété n'existe pas, les programmes d'écoute HTTP(S) règlent la propriété sur FailRequest. Si la propriété ne possède pas de valeur correcte, le programme d'écoute HTTP(S) ne parviendra pas à effectuer l'initialisation.	FailRequest	N
++++BOPrefix	La valeur de cette propriété est transmise au gestionnaire de données.		Non
++++Host	Le programme d'écoute écoute le trafic à l'adresse IP indiquée par la valeur de cette propriété. Si Host n'est pas indiqué, la valeur par défaut est localhost. Notez que vous pouvez soit indiquer soit un nom d'hôte(nom DNS) soit une adresse IP pour la machine sur laquelle le programme d'écoute s'exécute. Une machine peut posséder plusieurs adresses IP ou plusieurs noms.	`localhost`	Non

Tableau 35. Propriétés de configuration spécifiques au programme d'écoute du protocole HTTP et HTTPS (suite)

Nom	Valeurs possibles	Valeur par défaut	Obligatoire
++++Port	Le port sur lequel le programme d'écoute écoute les demandes. S'il n'est pas indiqué, le port utilisé par défaut est le port `80` pour HTTP et le port `443` pour HTTPS. Si vous copiez le programme d'écoute dans un connecteur, la combinaison des propriétés Host et Port peut être incapable d'établir des liens avec le port pour accepter des demandes.	`80` pour le programme d'écoute HTTP `443` pour le programme d'écoute HTTPS	Non
++++SocketQueueLength	Longueur de la file d'attente (file d'attente de socket) pour les demandes de connexion entrantes. Indique le nombre de connexions entrantes pouvant être stockées simultanément avant que l'hôte refuse des connexions. La longueur maximale de file d'attente dépend du système d'exploitation.	`5`	Non
++++RequestWaitTimeout	Exprimé en milli-secondes, intervalle pendant lequel le module d'exécution du programme d'écoute sera bloqué sur l'hôte et le port en attendant l'arrivée de demandes. S'il reçoit une demande avant cette intervalle, le programme d'écoute la traitera. Dans le cas contraire, l'unité d'exécution du programme d'écoute vérifie si l'indicateur d'arrêt du connecteur est défini. Si celui-ci est défini, le connecteur s'arrête. Dans le cas contraire, il continue à être bloqué pendant l'intervalle RequestWaitTimeout. Si cette propriété est réglée sur 0, il est bloqué indéfiniment. S'il n'est pas indiqué, la valeur par défaut est 60000 millisecondes.	`60000` (ms)	Non
++++HTTPReadTimeout	Exprimé en milli-secondes, intervalle pendant lequel le module d'exécution du programme d'écoute sera bloqué pendant la lecture d'une demande émanant d'un client. Si ce paramètre est réglé sur 0, le programme d'écoute sera bloqué indéfiniment jusqu'à la réception du message de demande complet.	`0`	Non
++++HttpAsyncResponseCode	Code réponse HTTP pour les demandes asynchrones adressées au programme d'écoute : 200 (OK) 202 (ACCEPTED)	`202` (ACCEPTED)	Non
++++Pingability	Propriété comprenant 0 spécification ou plus pour les services ping pour ce programme d'écoute. Cette propriété ne possède pas de valeur.		Non
+++++PingService1	Cette propriété comprend une spécification pour le service ping. Cette propriété ne comprend pas de valeur et le nom peut être n'importe quel nom de propriété correct. Remarque : il peut exister plusieurs instances structurellement identiques avec des valeurs différentes de cette propriété. En cas d'erreur lors de la validation de cette propriété, un message d'erreur est consigné et le programme d'écoute de protocole ne s'initialise pas.		Non

Tableau 35. Propriétés de configuration spécifiques au programme d'écoute du protocole HTTP et HTTPS (suite)

Nom	Valeurs possibles	Valeur par défaut	Obligatoire
++++++Method	Méthode HTTP de la demande, casse indifférenciée.		Oui
++++++StatusCode	Code état de la réponse à la demande.		Oui
++++++ReasonPhrase	Motif-Phrase en réponse à la demande.		Oui
++++URLsConfiguration	Il s'agit d'une propriété hiérarchique ne possédant pas de valeur. Elle contient une ou plusieurs configurations pour les URL prises en charge par ce programme d'écoute et, le cas échéant, les valeurs mime type et charset. Notez qu'il s'agit de la propriété enfant de la propriété hiérarchique ProtocolListenerFramework->ProtocolListeners->HTTPListener1. En l'absence de cette propriété, le programme d'écoute utilise des valeurs par défaut. Si cette propriété n'est pas indiquée, le programme d'écoute utilise la propriété URLsConfiguration définie sur >ContextPath: "/". >Enabled: true >DataHandler MimeType: égal au ContentType de la demande >Charset : selon les règles de la section de globalisation.	ContextPath : / Enabled : true MimeType du gestionnaire de données : égal au ContentType de la demande Charset : NONE. Pour plus d'informations, voir «HTTP et HTTPS traitement du programme d'écoute de protocole», à la page 51.	Non
+++++URL1	Il s'agit d'une propriété hiérarchique ne possédant pas de valeur. Ses enfants fournissent le nom de l'URL prise en charge par ce programme d'écoute. Plusieurs URL peuvent être prises en charge. Notez que vous pouvez brancher plusieurs URL supplémentaires en copiant cette propriété et sa hiérarchie.		Non
++++++ContextPath	L'URI des demandes HTTP reçues par le programme d'écoute. Cette valeur doit être unique parmi les valeurs ContextPath de la propriété URLsConfiguration. Autrement, le connecteur consignera une erreur et ne pourra démarrer. La casse des valeurs ContextPath est différenciée. Cependant, elles peuvent contenir le protocole, le nom d'hôte et le port dont la casse est indifférenciée. Si le protocole est indiqué dans ContextPath, il doit s'agir de http. Si l'hôte est indiqué, il doit s'agir de la valeur de la propriété Host du programme d'écoute. Si le port est indiqué, il doit s'agir de la valeur de la propriété Port du programme d'écoute.		Non
++++++Enabled	La valeur de cette propriété détermine si la propriété hiérarchique de l'URL parent est activée pour le connecteur.	True	Non
++++++TransformationRules	Il s'agit d'une propriété hiérarchique ne possédant pas de valeur. Elle peut comporte une ou plusieurs règles de transformation.		Non
+++++++EmptyRequestRule	Il s'agit d'une propriété hiérarchique ne possédant pas de valeur. Cette propriété comporte la règle de transformation à utiliser en cas de charge de demande vide.		

Tableau 35. Propriétés de configuration spécifiques au programme d'écoute du protocole HTTP et HTTPS (suite)

Nom	Valeurs possibles	Valeur par défaut	Obligatoire
++++++++MimeType	Type mime DataHandler à utiliser quand le gestionnaire de données appelant traite des demandes à charge vide. (Notez que le gestionnaire de données recevra Hashtable d'entêtes de demande au lieu de l'objet de configuration)		Oui
++++++++Charset	Charset qui doit être utilisé lors de la transformation de la demande avec charge vide.		Non
++++++++BOPrefix	Si elle est définie, la valeur de cette propriété remplacera BOPrefix de ce programme d'écoute de protocole pour le transmettre au gestionnaire de données.		Non
++++++++BOName	Si elle est définie, la valeur de cette propriété sera transmise au gestionnaire de données dans la table de hachage des entêtes avec BOName comme nom de l'élément. Si un entête de protocole porte le même nom, la valeur d'entête de protocole est prioritaire sur cette valeur de propriété.		Non
++++++++BOVerb	Si elle est définie, la valeur de cette propriété sera transmise au gestionnaire de données dans la table de hachage des entêtes avec BOVerb comme nom de l'élément. Si un entête de protocole porte le même nom, la valeur d'entête de protocole est prioritaire sur cette valeur de propriété.		Non
+++++++TransformationRule1	Il s'agit d'une propriété hiérarchique ne possédant pas de valeur. Elle comprend la règle de transformation.		Non
++++++++ContentType	La valeur de cette propriété indique le ContentType de la demande entrante à laquelle une gestion spéciale (type mime ou charset du gestionnaire de données) doit être appliquée. Si ContentType n'est pas indiqué par la propriété hiérarchique TransformationRuleN, le connecteur consigne un message d'avertissement et ne tient pas compte de la propriété TransformationRuleN. La valeur spéciale `*/*` de cette propriété permet aux programmes d'écoute de protocole d'appliquer cette règle à n'importe quel ContentType. Notez que si un programme d'écoute trouve plusieurs règles pour le même chemin d'accès de contexte partageant un ContentType, le programme d'écoute consigne une erreur et ne parvient pas à s'initialiser.		Non
++++++++MimeType	Type mime à utiliser lors de l'appel d'un gestionnaire de données afin de traiter les demandes du ContentType indiqué.		Non
++++++++Charset	Charset à utiliser lors de la transformation de la demande du ContentType indiqué en objet métier.		Non
++++++++BOPrefix	Si elle est définie, la valeur de cette propriété remplacera BOPrefix de ce programme d'écoute de protocole pour le transmettre au gestionnaire de données.		Non

Tableau 35. Propriétés de configuration spécifiques au programme d'écoute du protocole HTTP et HTTPS (suite)

Nom	Valeurs possibles	Valeur par défaut	Obligatoire
++++++++BOName	Si elle est définie, la valeur de cette propriété sera transmise au gestionnaire de données dans la table de hachage des entêtes avec BOName comme nom de l'élément. Si un entête de protocole porte le même nom, la valeur d'entête de protocole est prioritaire sur cette valeur de propriété.		Non
++++++++BOVerb	Si elle est définie, la valeur de cette propriété sera transmise au gestionnaire de données dans la table de hachage des entêtes avec BOVerb comme nom de l'élément. Si un entête de protocole porte le même nom, la valeur d'entête de protocole est prioritaire sur cette valeur de propriété.		Non

La figure 18 représente les propriétés telles qu'elles figurent dans Connector Configurator.

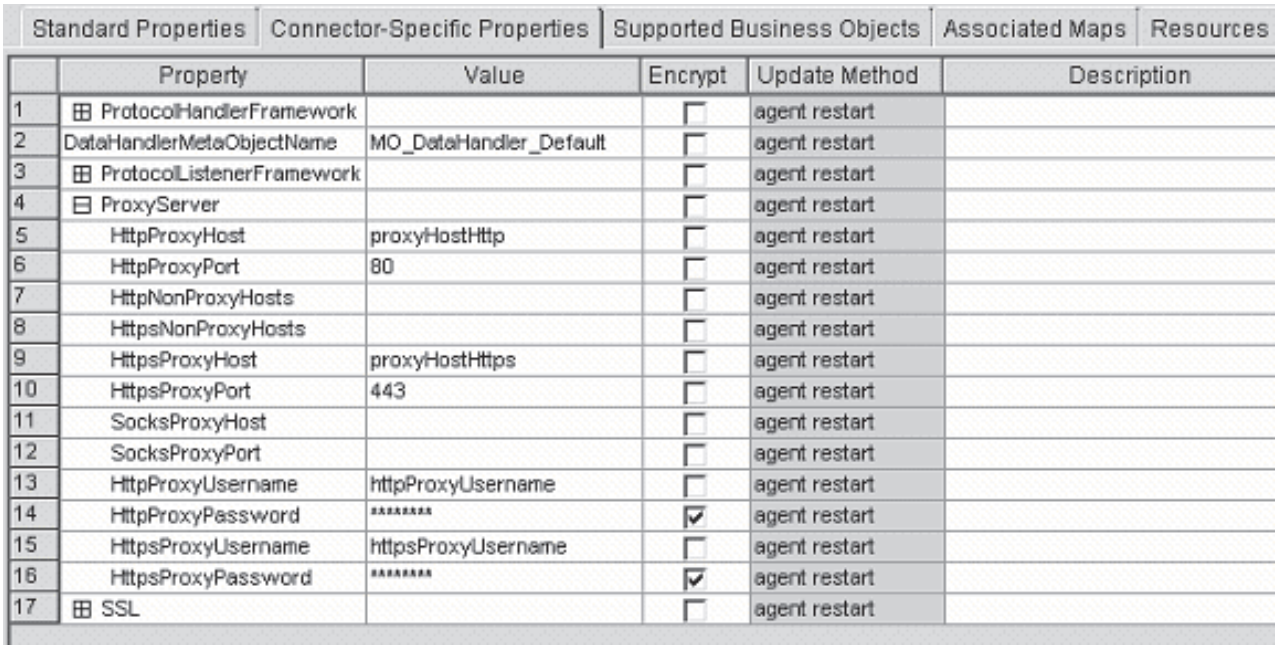
Standard Properties | Connector-Specific Properties | Supported Business Objects | Associated Maps | Resources

	Property	Value	Encrypt	Update Method	Description
1	⊞ ProtocolHandlerFramework		☐	agent restart	
2	DataHandlerMetaObjectName	MO_DataHandler_Default	☐	agent restart	
3	⊟ ProtocolListenerFramework		☐	agent restart	
4	WorkerThreadCount	10	☐	agent restart	
5	RequestPoolSize	20	☐	agent restart	
6	⊟ ProtocolListeners		☐	agent restart	
7	⊞ HTTPListener1		☐	agent restart	
8	⊟ HTTPSListener1		☐	agent restart	
9	Protocol	https	☐	agent restart	
10	Host	localhost	☐	agent restart	
11	Port	8443	☐	agent restart	
12	SocketQueueLength	5	☐	agent restart	
13	HTTPReadTimeout	0	☐	agent restart	
14	RequestWaitTimeout	60000	☐	agent restart	
15	BOPrefix		☐	agent restart	
16	⊞ URLsConfiguration		☐	agent restart	
17	⊞ ProxyServer		☐	agent restart	
18	⊞ SSL		☐	agent restart	

Figure 18. Propriétés du programme d'écoute de protocole HTTP(S)

ProxyServer : Configurez les valeurs enfants de cette propriété quand le réseau utilise un serveur proxy. Il s'agit d'une propriété hiérarchique ne possédant pas de valeur. Les valeurs enfants de cette propriété sont utilisées par les gestionnaires de protocole HTTP-HTTPS.

La figure 19 représente les propriétés ProxyServer telles qu'elles sont affichées dans Connector Configurator.



Standard Properties | Connector-Specific Properties | Supported Business Objects | Associated Maps | Resources

	Property	Value	Encrypt	Update Method	Description
1	⊞ ProtocolHandlerFramework		☐	agent restart	
2	DataHandlerMetaObjectName	MO_DataHandler_Default	☐	agent restart	
3	⊞ ProtocolListenerFramework		☐	agent restart	
4	⊟ ProxyServer		☐	agent restart	
5	HttpProxyHost	proxyHostHttp	☐	agent restart	
6	HttpProxyPort	80	☐	agent restart	
7	HttpNonProxyHosts		☐	agent restart	
8	HttpsNonProxyHosts		☐	agent restart	
9	HttpsProxyHost	proxyHostHttps	☐	agent restart	
10	HttpsProxyPort	443	☐	agent restart	
11	SocksProxyHost		☐	agent restart	
12	SocksProxyPort		☐	agent restart	
13	HttpProxyUsername	httpProxyUsername	☐	agent restart	
14	HttpProxyPassword	********	☑	agent restart	
15	HttpsProxyUsername	httpsProxyUsername	☐	agent restart	
16	HttpsProxyPassword	********	☑	agent restart	
17	⊞ SSL		☐	agent restart	

Figure 19. Propriétés ProxyServer

HttpProxyHost : Nom d'hôte du serveur proxy HTTP. Indiquez cette propriété si le réseau utilise un serveur proxy pour le protocole HTTP.

Valeur par défaut = aucune

HttpProxyPort : Numéro du port que le connecteur utilise pour se connecter au serveur proxy HTTP.

Valeur par défaut = 80

HttpNonProxyHosts : Le valeur de cette propriété définit un ou plusieurs hôtes (HTTP) qui doivent être connectés directement et non pas par l'intermédiaire du serveur proxy. Il peut s'agir d'une liste d'hôtes séparés les uns des autres par "|".

Valeur par défaut = aucune

HttpsProxyHost : Nom d'hôte du serveur proxy HTTPS.

Valeur par défaut = aucune

HttpsProxyPort : Numéro du port que le connecteur utilise pour se connecter au serveur proxy HTTPS.

Valeur par défaut = 443

HttpsNonProxyHosts : Le valeur de cette propriété définit un ou plusieurs hôtes (HTTPS) qui doivent être connectés directement et non pas par l'intermédiaire du serveur proxy. Il peut s'agir d'une liste d'hôtes séparés les uns des autres par "|".

Valeur par défaut = aucune

SocksProxyHost : Nom d'hôte du serveur proxy Socks. Indiquez cette propriété quand le réseau utilise un proxy socks.

Remarque : Le JDK sous-jacent doit prendre en charge Socks.

Valeur par défaut = aucune

SocksProxyPort : Numéro de port de connexion au serveur proxy Socks. Indiquez cette propriété quand le réseau utilise un proxy socks.

Valeur par défaut = aucune

HttpProxyUsername : Nom d'utilisateur du serveur proxy HTTP. Si vous indiquez ProxyServer ->HttpProxyUsername alors que la destination de la demande est une URL HTTP, le gestionnaire de protocole HTTP-HTTPS crée un entête Proxy-Authorization lors de l'authentification avec le proxy. Le gestionnaire utilise la méthode CONNECT pour l'authentification.

L'entête proxy-authentication est codée en base64 et possède la structure suivante :
Proxy-Authorization: Basic
Base64EncodedString

Le gestionnaire concatène les valeurs de propriété du nom d'utilisateur et du mot de passe séparées par le signe deux-points (:) afin de créer la chaîne base64.

Valeur par défaut = aucune

HttpProxyPassword : Mot de passe du serveur proxy HTTP. Pour plus d'informations sur l'utilisation de cette valeur, voir «HttpProxyUsername».

Valeur par défaut = aucune

HttpsProxyUsername : Nom d'utilisateur pour le serveur proxy HTTPS. Si vous indiquez ProxyServer ->HttpsProxyUsername alors que la destination de la demande est une URL HTTPS, le gestionnaire de protocole HTTP-HTTPS crée un entête Proxy-Authorization pour l'authentification avec le proxy. Le gestionnaire concatène les valeurs de propriété de configuration HttpsProxyUsername et HttpsProxyPassword séparées par le signe deux-points (:) afin de créer la chaîne base64.

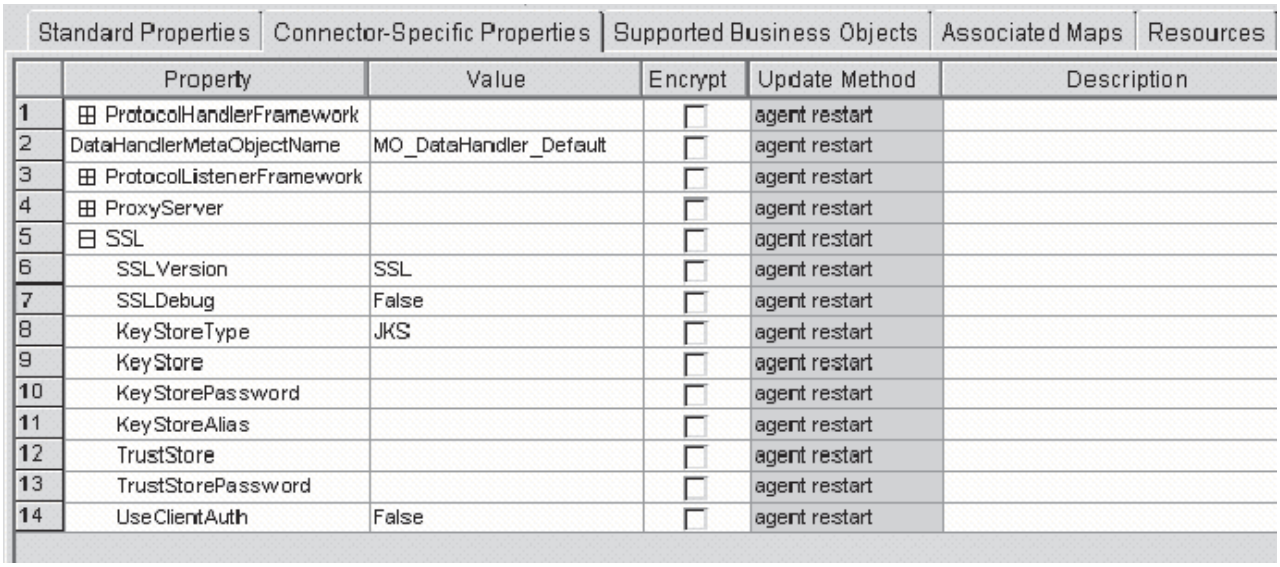
Valeur par défaut = aucune

HttpsProxyPassword : Mot de passe du serveur proxy HTTPS. Pour plus d'informations sur l'utilisation de cette valeur, voir «HttpsProxyUsername».

Valeur par défaut = aucune

SSL : Indiquez des valeurs sous cette propriété pour configurer SSL pour le connecteur. Il s'agit d'une propriété hiérarchique ne possédant pas de valeur.

La figure 20 représente les propriétés SSL telles qu'elles sont affichées dans Connector Configurator.



	Property	Value	Encrypt	Update Method	Description
1	⊞ ProtocolHandlerFramework		☐	agent restart	
2	DataHandlerMetaObjectName	MO_DataHandler_Default	☐	agent restart	
3	⊞ ProtocolListenerFramework		☐	agent restart	
4	⊞ ProxyServer		☐	agent restart	
5	⊟ SSL		☐	agent restart	
6	SSLVersion	SSL	☐	agent restart	
7	SSLDebug	False	☐	agent restart	
8	KeyStoreType	JKS	☐	agent restart	
9	KeyStore		☐	agent restart	
10	KeyStorePassword		☐	agent restart	
11	KeyStoreAlias		☐	agent restart	
12	TrustStore		☐	agent restart	
13	TrustStorePassword		☐	agent restart	
14	UseClientAuth	False	☐	agent restart	

Figure 20. Propriétés SSL

SSLVersion : Version SSL à utiliser par le connecteur. Pour plus d'informations sur les versions SSL prises en charge, voir la documentation JSSE IBM.

Valeur par défaut = `SSL`

SSLDebug : Si cette propriété est réglée sur `true`, le connecteur définit la valeur de la propriété système `javax.net.debug` sur `true`. IBM JSSE emploie cette propriété pour activer la fonction de trace. Pour plus d'informations, voir la documentation IBM JSSE.

Default = `false`

KeyStoreType : La valeur de cette propriété définit le type de KeyStore et de TrustStore. Pour plus d'informations sur les types de magasin de clés corrects, voir la documentation JSSE IBM.

Valeur par défaut = `JKS`

KeyStore : Cette propriété définit le chemin d'accès complet au fichier du magasin de clés. En l'absence de KeyStore et/ou de KeyStoreAlias, les propriétés KeyStorePassword, KeyStoreAlias, TrustStore, TrustStorePassword ne sont pas prises en compte. Le connecteur ne parvient pas à démarrer s'il ne peut charger le magasin de clés à l'aide du chemin d'accès indiqué dans cette propriété. Le chemin d'accès indiqué doit être le chemin d'accès complet au fichier du magasin de clés.

Valeur par défaut = aucune

KeyStorePassword : Cette propriété définit le mot de passe de la clé privée de Keystore.

Valeur par défaut = aucune

KeyStoreAlias : Cette propriété définit l'alias de la paire de clés dans le KeyStore. Les programmes d'écoute HTTPS utilise cette clé privée stockée dans le magasin de clés. De même, le gestionnaire de protocole HTTP-HTTPS utilise cet alias extrait

du magasin de clés lors de l'appel de services HTTPS nécessitant l'authentification client. La propriété doit être réglée sur un alias JSSE correct.

Valeur par défaut = aucune

TrustStore : Cette propriété définit le chemin d'accès complet au TrustStore. TrustStore permet de stocker les certificats approuvés par le connecteur. TrustStore doit être du même type que KeyStore. Vous devez définir le chemin d'accès complet au fichier TrustStore.

Valeur par défaut = aucune

TrustStorePassword : Cette propriété définit le mot de passe de Truststore.

Valeur par défaut = aucune

UseClientAuth : Cette propriété indique si l'authentification client SSL est employée. Quand elle est réglée sur `true`, les programmes d'écoute HTTPS utilisent l'authentification client.

Valeur par défaut = `false`

Création programmes d'écoute multiples

Vous pouvez créer plusieurs instances de programmes d'écoute. Les programmes d'écoute de protocole sont configurés en tant que propriétés enfants de la propriété de connecteur ProtocolListenerFramework -> ProtocolListeners. Chaque enfant(de ProtocolListenerFramework -> ProtocolListeners) identifie un programme d'écoute de protocole distinct pour le connecteur. Ainsi, vous pouvez créer des programmes d'écoute supplémentaires en configurant de nouvelles propriétés enfants de la propriété ProtocolListeners. Assurez-vous d'indiquer toutes les propriétés enfants de la propriété du programme d'écoute que vous venez de créer. Chaque programme d'écoute doit porter un nom unique. Cependant, vous ne modifiez pas la propriété Protocol du programme d'écoute (`http` ou `https`), qui doit rester la même pour les multiples instances d'un programme d'écoute.

Remarque : La propriété Protocole est essentiel car elle joue le rôle de commutateur. Si vous ne souhaitez pas utiliser de programme d'écoute ou de gestionnaire, laissez cette propriété vide.

Si vous créez plusieurs instances d'un programme d'écoute HTTP ou HTTPS, assurez-vous d'indiquer des propriétés Port et Host pour chaque instance.

Vous ne pouvez pas créer plusieurs instances d'un gestionnaire. Il ne doit exister plus d'un protocole par gestionnaire.

Connecteur au démarrage

Lorsque vous démarrez le connecteur, la méthode `init()` lit les propriétés de configuration qui ont été définies à l'aide du configurateur de connecteur du gestionnaire système. Pour un bon fonctionnement, veillez à ne pas désactiver la collecte de connecteurs (cette option est activée par défaut). Les sections ci-après décrivent ce qui se passe.

Configuration de proxy

Si vous indiquez la propriété spécifique du connecteur ProxyServer, le connecteur configure les propriétés du système proxy. Un serveur proxy est utilisé avec le gestionnaire de protocole HTTP-HTTPS pour le traitement des demandes uniquement. Le connecteur effectue également le suivi des propriétés système qu'il configure. Pour plus d'informations sur la propriété ProxyServer, voir «Propriétés de configuration spécifiques au connecteur», à la page 67.

Initialisation de la structure de programme d'écoute de protocole

Lors du démarrage, le connecteur instancie la structure de programme d'écoute de protocole et l'initialise. Cette structure lit la propriété spécifique au connecteur ProtocolListenerFramework. Le connecteur lit ensuite la valeur des propriétés WorkerThreads et RequestPoolSize. Si la propriété ProtocolListenerFramework n'est pas spécifiée ou manquante, le connecteur ne peut recevoir de demandes émanant de clients et, par conséquent, consigne un avertissement dans l'historique.

Le connecteur lit ensuite la propriété ProtocolListenerFramework -> ProtocolListeners. Toutes les propriétés de premier niveau de la propriété ProtocolListeners représentent des programmes d'écoute de protocole. La structure de programme d'écoute de protocole tente de charger et d'initialiser chaque programme d'écoute et d'en effectuer le suivi. S'il prend en charge les événements persistants, le programme d'écoute tente une reprise d'événement.

Initialisation de la structure de gestionnaire de protocole

Le connecteur lit la propriété spécifique ProtocolHandlerFramework, puis instancie et initialise la structure de gestionnaire de protocole. Si cette propriété est manquante ou n'est pas définie correctement, le connecteur ne peut traiter les demandes et consigne un avertissement. Le connecteur lit ensuite toutes les propriétés ProtocolHandlerFramework -> ProtocolHandlers correspondant aux gestionnaires de protocole et tente de les charger, de les initialiser et d'effectuer leur suivi. Notez que les gestionnaires de protocole sont chargés lors de l'initialisation du connecteur et ne sont pas instanciés quand une collaboration effectue une demande de service. Les gestionnaires de protocole sont protégés face aux unités d'exécution multiples.

Consignation

Le connecteur consigne un avertissement quand :

- la propriété ProtocolListenerFramework n'est pas indiquée. Le connecteur avertit qu'il ne procède pas à la notification des événements.
- la propriété ProtocolHandlerFramework n'est pas indiquée. Le connecteur avertit qu'il ne peut effectuer le traitement des demandes (collaboration).

Tracage

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

Les niveaux de trace de connecteur sont les suivants :

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

Niveau 1
Utilisez la fonction de trace chaque fois que la méthode pollForEvents est appelée. Utilisez la fonction de trace sur le nom TLO créé par les programmes de trace pour les livraisons à ICS. Utilisez le nom d'objet métier de requête et le nom d'attribut correspondant dans le TLO.

Niveau 2
Utilisez ce niveau pour les messages de trace consignés à chaque fois qu'un objet métier est posté dans Interchange Server, à partir de `gotApplEvent()` ou `executeCollaboration()`. Utilisez également la fonction de trace pour déterminer quel gestionnaire de protocole traite la demande.

Niveau 3
Utilisez la fonction de trace pour les ASI de l'objet métier en cours de traitement. Utilisez la fonction de trace pour les attributs de l'objet métier en cours de traitement. Utilisez la fonction de trace sur le TLO de l'objet métier de la demande lors de la notification d'événements. Utilisez la fonction de trace pour l'objet métier renvoyé par le gestionnaire de données.

Niveau 4
Utilisez la fonction de trace pour les entêtes de transport associés à :

- un message de demande récupéré par le programme d'écoute de protocole auprès du transport
- un message de réponse envoyé au client par le programme d'écoute de protocole.

Utilisez la fonction de trace pour les intercommunications d'unités d'exécution, les ASI traités et toutes les entrées et sorties de fonctions importantes.

Niveau 5
Utilisez la fonction de trace pour :

- les entrées et sorties de chaque méthode importante
- toutes les propriétés spécifiques de configuration
- le chargement de chaque programme d'écoute de protocole
- le message de demande récupéré par le programme d'écoute de protocole auprès du transport
- le message de réponse envoyé au client sur le transport par le programme d'écoute de protocole
- le chargement de chaque gestionnaire de protocole
- les messages retournés par le gestionnaire de données
- les vidages d'objets métier du TLO envoyés à la collaboration
- les vidages des objets métier retournés par le gestionnaire de données

Chapitre 5. Identification et résolution des incidents

Le présent chapitre décrit les problèmes que vous êtes susceptible de rencontrer lorsque vous démarrez ou exécutez le connecteur.

Incidents de démarrage

Incident	Solution potentielle / explication
Algorithme non pris en charge/Algorithme 'SSL' indisponible	Cette erreur survient quand la version SSL indiquée dans le configurateur de connecteur n'est pas pris en charge par votre fournisseur JSSE. Solution : vérifiez les versions SSL prises en charge dans la documentation du fournisseur JSSE. Pour IBM JSSE, assurez-vous que le fichier java.security du répertoire *ProductDir*/lib/security comporte la ligne suivante `security.provider.<number>=com.ibm.jsse.IBMJSSEProvider` où `<number>` représente l'ordre de préférence de chargement du fournisseur de sécurité.
Erreur de chargement du magasin de clés : chemin du fichier Keystore : "<*chemin*>" spécifié de façon incorrect : magasin de clés introuvable	Cette erreur survient si vous indiquez un chemin d'accès incorrect pour les fichiers de magasins de clés et/ou truststore. Solution : vérifiez le chemin d'accès des fichiers de magasins de clés indiqué dans la propriété SSL->KeyStore du configurateur Connector. De même, si vous utilisez truststore, vérifiez le chemin d'accès du fichier truststore indiqué dans la propriété SSL->TrustStore du configurateur Connector.
KeyManagementError : KeyStore est détourné, erreur KeyManagement	Cette erreur se produit en cas de détournement ou d'endommagement de votre magasin de clés et/ou de votre truststore. Elle peut également se produire si vous avez indiqué une valeur incorrecte pour le mot de passe. Solution : assurez-vous que le magasin de clés n'a pas été détourné. Essayez de recréer le magasin de clés. Vérifiez également que vous avez enté un mot de passe correct au niveau des propriétés SSL->KeyStorePassword et SSL->TrustStorePassword du connecteur.
Erreur de chargement de certificats depuis le magasin de clés	Cette erreur se produit en cas de détournement de vos certificats, de votre magasin de clés et/ou de votre truststore. Elle peut également se produire si vous avez indiqué une valeur incorrecte pour le mot de passe. Solution : vérifiez si le certificat, le magasin de clés ou le truststore a été détourné. Vérifiez également que vous avez enté un mot de passe correct au niveau des propriétés SSL->KeyStorePassword et SSL->TrustStorePassword du connecteur.

Incident	Solution potentielle / explication
Erreur de création du socket serveur, fin du traitement : erreur	Cette erreur se produit si le programme d'écoute de protocole HTTP ou HTTPS ne parvient pas à se connecteur au port indiqué dans les propriétés de connecteur. Solution : vérifiez les ports indiqués pour tous les programmes d'écoute de protocole HTTP et HTTPS. Si le même port est indiqué pour plusieurs programmes d'écoute, un seul programme d'écoute peut démarrer. De plus, vérifiez si un autre service s'exécute actuellement sur ce port. Dans l'affirmative, vous pouvez choisir un port différent pour les programmes d'écoute de protocole.
KeyManagementError:UnrecoverableKeyException, impossible de récupérer les clés	Cette erreur se produit s'il n'est pas possible d'utiliser le magasin de clés ou le truststore. Solution : créez un magasin de clés.
Exception de négociation SSL : CA inconnu	Ceci se produit si votre truststore ne comporte pas de certificat de CA. Solution : vérifiez si le certificat du CA et ses certificats auto-signés résident dans le truststore. Vérifiez également que le DN du certificat comporte le nom d'hôte (de préférence, l'adresse IP).
Le fichier journal comporte un nombre excessif d'enregistrements générés par JSSE.	Si vous ne souhaitez pas afficher toutes les informations détaillées JSSE sous-jacentes à la console, réglez la valeur de la propriété SSL->SSLDebug sur `false` dans le configurateur de connecteur.
Vous avez indiqué un programme d'écoute de protocole dont l'initialisation n'a pas lieu ; un message d'avertissement similaire s'affiche dans le connecteur : `Ensemble de propriétés ignoré pour le programme d'écoute de protocole "NOM_PROGRAMME_ECOUTE" avec la propriété de protocole "" : impossible d'identifier la classe du programme d'écoute de protocole.]`	Le connecteur n'a pas réussi à extraire une valeur valide de la propriété Protocol du programme d'écoute de programme. Les valeurs valides sont `http` ou `https`. Solution : il ne s'agit pas d'une condition d'erreur. Cependant, si vous souhaitez que le connecteur utilise ce programme d'écoute, indiquez une valeur de propriété Protocol correct.
Vous avez indiqué un gestionnaire de protocole mais il n'est pas en cours d'initialisation ; le message d'avertissement s'affiche dans le connecteur. `Impossible de déterminer le type du gestionnaire ; initialisation du gestionnaire actuel ignorée. Détails des propriétés du gestionnaire : Nom : <Nom du gestionnaire>; Valeur : Nom : Protocole; Valeur : Nom : ResponseWaitTimeout; Valeur : Nom : ReplyToQueue; Valeur : .]`	Le connecteur n'a pas réussi à extraire une valeur valide de la propriété Protocol du gestionnaire. Les valeurs valides sont `http` et `https`. Solution : il ne s'agit pas d'une condition d'erreur. Cependant, si vous souhaitez que le connecteur utilise ce gestionnaire, indiquez une valeur de propriété Protocol correct.

Erreurs d'exécution

Incident	Solution potentielle / explication
`Erreur d'analyse de réponse HTTP : Fin du flux atteinte pendant la lecture de l'entête de réponse HTTP`	Cette erreur survient lorsque le connecteur appelle un service HTTP. En effet, le service HTTP cible a envoyé une réponse HTTP incorrecte. Solution : assurez-vous que l'adresse du service HTTP cible est correcte.
`Erreur dans l'URL indiquée, impossible d'extraire les informations détaillées d'hôte et de port, destination <URL> erronée`	Cette erreur survient lorsque le connecteur appelle un service HTTP. Elle se produit car vous avez indiqué une adresse de point d'extrémité incorrecte pour le service HTTP. Solution : assurez-vous que vous avez indiqué l'adresse correcte du service HTTP.

Incident	Solution potentielle / explication
Echec de l'envoi de l'objet métier d'événement <*Nom d'objet métier*> avec l'instruction <*Instruction*> au courtier. Etat d'exécution reçu "-1" et message d'erreur : `MapException : Impossible de trouver la mappe vers les objets métier <nom BO> pour le contrôleur de connecteur HTTPConnector`	Cette erreur se produit quand le courtier d'intégration ne parvient pas à traiter l'événement car la collaboration à laquelle le connecteur envoie l'événement de façon synchrone n'existe pas ou n'accepte pas l'instruction d'objet métier. Solution : si vous utilisez un TLO pour la notification d'événement, examinez les informations ASI de niveau objet `ws_collab`. (Le nom du TLO figure dans le message d'erreur.) Vérifiez la valeur ASI de `ws_collab`. Assurez-vous que cette collaboration existe et s'exécute. Si les informations ASI de niveau objet métier `ws_mode` sont réglées sur `synch`, les informations ASI `ws_collab` sont requises. Vérifiez la valeur ASI de niveau objet de `ws_verb`. Assurez-vous que la collaboration indiquée par les informations ASI `ws_collab` peuvent être déclenchées par l'instruction indiquée dans les informations ASI `ws_verb`. Assurez-vous que cette collaboration existe et s'exécute.
Impossible de transformer une demande en objet métier de demande. Erreur : `Echec de la génération d'objet de demande - aucune instruction n'a pu être définie sur l'objet métier de demande`	Cette erreur se produit lors de la notification d'événement quand le connecteur est incapable de déterminer l'instruction de l'objet métier que le connecteur tente d'envoyer au courtier d'intégration. Solution : assurez-vous d'avoir indiqué les informations ASI de niveau objet `ws_verb` pour ce TLO. Indiquez l'instruction comme valeur de ces informations ASI.

Annexe A. Propriétés de configuration standard pour les connecteurs

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

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

Si votre adaptateur prend en charge DB2 Information Integrator, utilisez les options et les propriétés standard DB2 II (voir la colonne Remarques du tableau 36, à la page 89.)

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 Méthode de mise à jour dans le tableau 36, à la page 89.

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 36 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 36, 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 36. 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 36. Récapitulatif des propriétés de configuration standard (suite)

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

Tableau 36. 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 36. 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 36. 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 36. 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 36. 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 98.

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 «Exécution de Connector Configurator en mode autonome», à la page 114).

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 116 pour configurer un nouveau modèle.

Utilisation des connecteurs sous UNIX

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

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

Démarrage de Connector Configurator

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

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

Exécution de Connector Configurator en mode autonome

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

Pour ce faire, procédez comme suit :

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

Vous pouvez choisir d'exécuter Connector Configurator en mode autonome pour créer le fichier, puis de vous connecter à System Manager afin de l'enregistrer dans un projet System Manager (voir «Remplissage d'un fichier de configuration», à la page 121).

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 116.
- 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 124.
4. Une fois la configuration terminée, cliquez sur **File>Save>To Project** ou sur **File>Save>To File**.

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

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

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

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

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

Définition des propriétés d'un fichier de configuration

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

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

- Propriétés standard
- Propriétés spécifiques au connecteur

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

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

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

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

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

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

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

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

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

Définition des propriétés standard du connecteur

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

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

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

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

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

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

Configuration des propriétés spécifiques au connecteur

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

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

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

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

   Remarque : Si le bouton n'apparaît pas, c'est qu'il n'existe pas d'aide étendue pour cette propriété.
5. Vous pouvez enregistrer ou ignorer les modifications, comme décrit pour «Définition des propriétés standard du connecteur», à la page 123.

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 88.

Indication des définitions d'objets métier pris en charge

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

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

Si ICS est votre courtier

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

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

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

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

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

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

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

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

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

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

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

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

Si votre courtier est un courtier de messages WebSphere

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

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

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

Si WAS est votre courtier

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

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

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

Mappes associées (ICS)

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

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

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

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

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

- **Business Object Name**

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

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

- **Associated Maps**

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

- **Liaison explicite**

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

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

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

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

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

Ressources (ICS)

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

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

Messagerie (ICS)

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

Validation des files d'attente de messages

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

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

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

Sécurité (ICS)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Définition des valeurs du fichier de trace ou du fichier journal

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

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

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

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

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

Gestionnaires de données

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

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

Enregistrement de votre fichier de configuration

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

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

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

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

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

Modification d'un fichier de configuration

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

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

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

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

Exécution de la configuration

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

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

Utilisation de Connector Configurator dans un environnement globalisé

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

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

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

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

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

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

Annexe C. Tutoriel Adapter for HTTP

Les procédures de cette annexe :

- décrivent la transmission d'événements asynchrones et synchrones pour le traitement des demandes et des événements
- illustre la configuration de l'adaptateur HTTP pour un exemple HTTP
- illustre la configuration de l'adaptateur HTTP pour un exemple HTTPS

A propos de ce tutoriel

Ce tutoriel a pour but de décrire la transmission d'événements synchrones et asynchrones pour le traitement des demandes et des événements effectués par HTTP et HTTPS au niveau d'Adapter for HTTP. Dans chaque scénario, les adaptateurs agissent en tant que :

- client HTTP appelant une URL externe
- proxy qui écoute les demandes HTTP sur une URL et les achemine vers une collaboration WebSphere ICS.

Le tutoriel est conçu pour illustrer les fonctions de base de l'adaptateur dans des scénarios exemples :

- **Scénario asynchrone** illustrant un événement asynchrone de type HTTP POST (demande uniquement). Ce scénario comprend deux exemples —pour la simplicité de la configuration, le même adaptateur est employé pour écouter les demandes HTTP et appeler une URL en tant que client HTTP.
  - **Proxy qui écoute les demandes HTTP sur une URL** Dans cet exemple, la demande entrante est acheminée vers la collaboration `SERVICE_ASYNCH_Order_Collab` dans WebSphere ICS. La collaboration est dénommée Asynch Order. Si l'adaptateur est configuré correctement, cette collaboration peut être appelée à l'aide de l'un des protocoles : HTTP ou HTTPS. `SERVICE_ASYNCH_Order_Collab` est une collaboration passerelle acceptant `SERVICE_ASYNCH_TLO_Order`. Le port déclenchant (From) de la collaboration est lié à HTTPConnector. Le port de service (To) est lié à SampleSiebelConnector.
  - **Client HTTP appelant une URL externe** Dans cet exemple, le client HTTP est une autre collaboration `CLIENT_ASYNCH_Order_Collab` dans WebSphere ICS qui appellera une URL externe de façon asynchrone à l'aide de l'adaptateur HTTP. Si l'adaptateur est configuré correctement, ce client HTTP peut appeler l'URL externe à l'aide de l'un des protocoles : HTTP ou HTTPS. `CLIENT_ASYNCH_Order_Collab` est une collaboration passerelle acceptant `CLIENT_ASYNCH_TLO_Order`. Le port déclenchant (From) de la collaboration est lié à SampleSAPConnector. Le port de service (To) est lié à HTTPConnector.

  Les deux exemples de ce scénario asynchrone impliquent deux applications :
  - SampleSiebel : crée une commande pour ses clients.
  - SampleSAP : crée une commande

- **Scénario synchrone** illustrant un événement synchrone de type HTTP POST (demande- réponse). Ce scénario comprend deux exemples —pour la simplicité de la configuration, le même adaptateur est employé pour écouter les demandes HTTP et appeler une URL en tant que client HTTP.
  - **Proxy qui écoute les demandes HTTP sur une URL** Dans cet exemple, la demande entrante est acheminée vers la collaboration `SERVICE_SYNCH_OrderStatus_Collab` dans WebSphere ICS. La collaboration est dénommée Synch OrderStatus. Si l'adaptateur est configuré correctement, cette collaboration peut être appelée à l'aide de l'un des protocoles : HTTP ou HTTPS. `SERVICE_SYNCH_OrderStatus_Collab` est une collaboration passerelle acceptant `SERVICE_SYNCH_TLO_OrderStatus`. Le port déclenchant (From) de la collaboration est lié à HTTPConnector. Le port de service (To) est lié à SampleSiebelConnector.
  - **Client HTTP appelant une URL externe** Dans cet exemple, le client HTTP est une autre collaboration `CLIENT_SYNCH_OrderStatus_Collab` dans WebSphere ICS qui appellera une URL externe à l'aide de l'adaptateur HTTP. Si l'adaptateur est configuré correctement, ce client HTTP peut appeler l'URL externe à l'aide de l'un des protocoles : HTTP ou HTTPS. `CLIENT_SYNCH_OrderStatus_Collab` est une collaboration passerelle acceptant `CLIENT_SYNCH_TLO_OrderStatus`. Le port déclenchant (From) de la collaboration est lié à SampleSAPConnector. Le port de service (To) est lié à HTTPConnector.

  Les deux exemples de ce scénario synchrone impliquent deux applications :
  - SampleSiebel : récupère l'état des commandes pour ses clients.
  - SampleSAP : demande l'état de la commande

Les deux scénarios impliquent la simulation de SampleSiebelConnector et de SampleSAPConnector à l'aide des deux connecteurs de test.

Avant de commencer

Avant de commencer, vérifiez que :

- Vous avez avez installé WebSphere ICS 4.2.2 ou ultérieur et que vous êtes familiarisé avec ce produit.
- Vous avez installé WebSphere Business Integration Adapter for HTTP dans le répertoire de base de WebSphere ICS.
- Vous êtes familiarisé avec la technologie HTTP.
- Vous êtes familiarisé avec la technologie XML.

Installation et configuration

Dans les sections qui suivent, *WBI_folder* désigne le dossier contenant votre installation WebSphere ICS. Les variables d'environnement et les séparateurs de fichier figurent tous au format Windows 2000 ou 2003. Apportez les modifications requises si vous travaillez sous AIX ou Solaris. (par exemple, WBI_folder\`connectors` serait WBI_folder/`connectors`).

Démarrage du serveur et de l'outil

1. Démarrez WebSphere InterChange Server (ICS) à partir du raccourci.
2. Démarrez WebSphere Business Integration System Manager et ouvrez Component Navigator Perspective.
3. Enregistrez et connectez votre serveur en tant qu'instance serveur dans la vue Interchange Server.

Chargement du contenu exemple

A partir de Component Navigator Perspective :

1. Créez une nouvelle bibliothèque Integration Component Library.
2. Importez le fichier référentiel `HTTPSample.jar` situé dans : WBI_folder\connectors\HTTP\samples\WebSphereICS\

Compilation des modèles de collaboration

A l'aide de WebSphere Business Integration System Manager :

- **Compilez tous** les modèles de collaboration qui ont été importés depuis le fichier référentiel `HTTPSample.jar`.

Configuration du connecteur

1. Si ce n'est déjà fait, configurez le connecteur tel que le décrit ce guide et selon votre système.
2. A l'aide de WebSphere Business Integration System Manager, ouvrez HTTPConnector dans Connector Configurator.
3. Vous devez également configurer HTTPConnector pour le protocole à utiliser dans cet exemple :
   - Si vous souhaitez utiliser HTTP, consultez «Configuration du scénario de protocole HTTP» afin de configurer le connecteur pour HTTP.
   - Si vous souhaitez utiliser HTTPS, consultez «Configuration du scénario de protocole HTTPS», à la page 136 afin de configurer le connecteur pour HTTPS.

Configuration du scénario de protocole HTTP

Cette section décrit la configuration du connecteur pour le scénario HTTP exemple. Comme indiqué dans ce document, le connecteur comprend un programme d'écoute HTTP et un gestionnaire de protocole HTTP-HTTPS.

Dans les étapes et descriptions qui suivent, les propriétés hiérarchiques de configuration du connecteur sont représentées par le symbole ->. Par exemple, A-> B signifie que A est une propriété hiérarchique, et B est propriété enfant de A.

Pour configurer le programme d'écoute HTTP pour cet exemple :

1. Dans Connector Configurator, cliquez sur **Connector-Specific Properties** pour le connecteur HTTPConnector.
2. Développez la propriété **ProtocolListenerFramework** pour afficher la propriété enfant ProtocolListeners.
3. Développez la propriété enfant **ProtocolListeners** pour afficher la propriété enfant **HTTPListener1**.
4. Vérifiez les propriétés **HTTPListener1->Host** et **HTTPListener1->Port**. Assurez-vous qu'aucun autre processus ne s'exécute sur l'hôte et n'écoute les communications sur ce port TCP/IP. Le cas échéant, vous pouvez définir la valeur de **HTTPListener1->Host** sur le nom de l'ordinateur qui exécutera le connecteur.

Il n'est pas nécessaire de configurer le gestionnaire de protocole HTTP-HTTPS pour l'exemple ; cependant, pour configurer la fonctionnalité SSL (des exemples sont fournis avec les composants dépendant de SSL), consultez «Configuration du scénario de protocole HTTPS», à la page 136.

Configuration du scénario de protocole HTTPS

Cette section décrit la configuration du connecteur pour le scénario HTTPS exemple. Le connecteur inclut le programme d'écoute de protocole HTTPS et le gestionnaire de protocole HTTP-HTTPS.

Dans les étapes et descriptions qui suivent, les propriétés hiérarchiques de configuration du connecteur sont représentées par le symbole ->. Par exemple, A-> B signifie que A est une propriété hiérarchique, et B est propriété enfant de A.

Remarque : Outre les éléments de pré-installation décrits dans «Avant de commencer», à la page 134, vous devez également avoir créé et testé votre magasin de clés et votre truststore à l'aide du logiciel de gestion des clés et des certificats.

Configuration des propriétés spécifiques au connecteur SSL : Pour HTTPS, le connecteur nécessite que vous configuriez la propriété hiérarchique spécifique au connecteur SSL.

1. Dans Connector Configurator, cliquez sur l'onglet **Connector-Specific Properties** pour le connecteur HTTPConnector.
2. Développez la propriété hiérarchique **SSL** pour afficher toutes ses propriétés enfants. De plus, vérifiez ou modifiez les propriétés enfants de la propriété hiérarchique spécifique au connecteur SSL.
   - **SSL-> KeyStore** Indiquez le chemin d'accès complet à votre fichier magasin de clés, que vous devez créer à l'aide de votre logiciel de gestion des clés et des certificats.
   - **SSL->KeyStorePassword** Définissez le mot de passe requis pour accéder au magasin de clés.
   - **SSL->KeyStoreAlias** Définissez l'alias de la clé privée dans le magasin de clés.
   - **SSL->TrustStore** Définissez le chemin d'accès complet de votre fichier truststore que vous avez créé à l'aide du logiciel de gestion des clés et des certificats.
   - **SSL->TrustStorePassword** Définissez le mot de passe d'accès au TrustStore.

Remarque : N'oubliez pas de sauvegarder les modifications dans Connector Configurator.

Configuration du programme d'écoute de protocole HTTPS :

1. Dans Connector Configurator, cliquez sur **Connector-Specific Properties** pour le connecteur HTTPConnector.
2. Développez la propriété **ProtocolListenerFramework** pour afficher la propriété enfant **ProtocolListeners**.
3. Développez la propriété enfant **ProtocolListeners** pour afficher la propriété enfant **HTTPSListener1**. Vérifiez la valeur des propriétés **HTTPSListener1->Host** et **HTTPSListener1->Port**. Assurez-vous qu'aucun autre processus ne s'exécute sur l'hôte et n'écoute les communications sur ce port TCP/IP. Le cas échéant, vous pouvez définir la valeur de **HTTPSListener1->Host** sur le nom de l'ordinateur qui exécute le connecteur.

Vous ne devez pas configurer le gestionnaire de protocole HTTP-HTTPS pour l'exemple.

Configuration de KeyStore et de TrustStore : Vous pouvez rapidement configurer KeyStore et TrustStore afin de les utiliser avec le scénario exemple. Pour les

systèmes de production, utilisez un logiciel tiers de configuration et de gestion des magasins de clés et de génération de certificats et de clés. Adapter for HTTP ne comporte pas d'outil de configuration et de gestion de ces ressources.

Dans cette section, on suppose que la machine virtuelle Java (JVM) est installée sur votre système et que vous êtes familiarisé avec l'outil keytool fourni avec cette JVM. Pour plus d'informations sur l'outil keytool et sur les incidents liés, consultez la documentation de votre JVM.

Pour configurer KeyStore :

1. Créez KeyStore à l'aide de keytool. Vous devez créer une paire de clés dans KeyStore. Pour cela, entrez ce qui suit sur la ligne de commande :

   ```
   keytool -genkey -alias httpadapter -keystore c:\security\keystore
   ```

2. keytool vous demande alors un mot de passe. Indiquez le mot de passe que vous avez saisi pour la valeur de la propriété du connecteur SSL->KeyStorePassword.
   Notez que si vous avez saisi `-keystore c:\security\keystore` sur la ligne de commande dans l'exemple précédent, vous devez saisir `c:\security\keystore` comme valeur de la propriété SSL->KeyStore. De même, si vous avez entré `-alias httpadapter` sur la ligne de commande, vous devez entrer `httpadapter` comme valeur de la propriété SSL->KeyStoreAlias. keytool vous demande alors d'entrer des informations détaillées sur le certificat. Voici que vous pourriez entrer à chaque invite (pour toute information, consultez la documentation de keytool).

   ```
   What is your first and last name?
        [Unknown]:  HostName
   What is the name of your organizational unit?
        [Unknown]:  myunit
   What is the name of your organization?
        [Unknown]:  myorganization
   What is the name of your City or Locality?
        [Unknown]:  mycity
   What is the name of your State or Province?
        [Unknown]:  mystate
   What is the two-letter country code for this unit?
        [Unknown]:  mycountryIs <CN=HostName, OU=myunit, O=myorganization,
        L=mycity, ST=mystate, C=mycountry> correct?
        [no]:  yes
   ```

3. Notez que pour `What is your first and last name?`, vous devez entrer le nom de l'ordinateur sur lequel vous exécutez le connecteur. keytool affiche alors une invite :

   ```
   Entrez le mot de passe de la clé de <httpadapter> (appuyez sur RETOUR si le mot
   de passe est identique à celui du magasin de clés) :
   ```

4. Appuyez sur **Retour** pour utiliser le même mot de passe. Si vous souhaitez utiliser un certificat auto-signé, vous pouvez exporter le certificat créé précédemment. Pour cela, entrez ce qui suit sur la ligne de commande :

   ```
   C:\security>keytool -export -alias httpadapter -keystore c:\security\keystore
   -file c:\security\httpadapter.cer
   ```

5. keytool vous demande alors le mot de passe du magasin de clés. Entrez le mot de passe saisi précédemment.

Pour configurer TrustStore :

1. Pour importer les certificats approuvés dans TrustStore, entrez la commande suivante :

   ```
   keytool -import -alias trusted1 -keystore c:\security\truststore
   -file c:\security\httpadapter.cer
   ```

2. keytool vous demande alors le mot de passe du magasin de clés. Si vous avez saisi `-keystore c:\security\truststore`, vérifiez que la propriété SSL->TrustStore est réglée sur `c:\security\truststore`. Réglez également la propriété SSL->TrustStorePassword sur le mot de passe que vous avez saisi précédemment.

Création d'un projet utilisateur

- A l'aide de WebSphere Business Integration System Manager, créez un nouveau **Projet utilisateur**. Sélectionnez tous les composants dans la bibliothèque Integration Component Library créée dans «Chargement du contenu exemple», à la page 135.

Ajout et déploiement du projet

1. Dans la vue Instance de serveur, ajoutez à WebSphere ICS le **Projet utilisateur** créé dans «Création d'un projet utilisateur».
2. Déployez tous les composants de ce projet utilisateur vers le serveur ICS.

Redémarrage d'ICS

1. Redémarrez ICS pour que toutes les modifications prennent effet.
2. A l'aide de l'outil Moniteur système, assurez-vous que tous les objets de collaboration, contrôleurs de connecteur et mappes sont opérationnels.

Exécution du scénario asynchrone

Ce scénario appelle le service HTTP de commande asynchrone. Avant d'exécuter le scénario, examinez soigneusement son flux de données.

1. Un événement `CLIENT_ASYNCH_TLO_Order.Create` se produit dans l'application `SampleSAP` qui s'exécute dans une instance de Test Connector.
2. L'événement est envoyé de `SampleSAP` vers la collaboration `CLIENT_ASYNCH_Order_Collab`.
3. L'événement est ensuite envoyé de la collaboration vers HTTPConnector.
4. HTTPconnector recherche ensuite l'objet `XML_Order` qui est enfant de l'objet `CLIENT_ASYNCH_TLO_Order`.
5. L'objet métier Request est converti en message XML à l'aide du gestionnaire de données XML. HTTPconnector envoie le message XML à l'URL fournie par l'attribut Destination du méta-objet Protocol Config. Le méta-objet Protocol Config utilisé par le connecteur dépend de la valeur de l'attribut Handler de `CLIENT_ASYNCH_TLO_Order`. Cette valeur doit être réglée sur `http` ou `https`.
6. La demande XML est envoyée à l'URL par la méthode POST. Comme indiqué précédemment, le même HTTPConnector écoute la demande XML sur la même URL. Le programme d'écoute de protocole du connecteur reçoit le message XML.
7. Le connecteur convertit le message XML en XML_Order et crée ensuite un objet `SERVICE_ASYNCH_TLO_Order`. L'objet XML_Order est défini comme enfant de l'objet SERVICE__ASYNCH_TLO_Order.
8. HTTPConnector adresse à présent SERVICE_TLO_Order à ICS de façon asynchrone. Cela termine l'appel URL asynchrone.

En raison du caractère asynchrone de l'appel (demande uniquement), aucune réponse n'est renvoyée au client HTTP. Quand `SERVICE_ASYNCH_Order_Collab` reçoit cet objet, la collaboration envoie l'objet métier alors à l'application dénommée `SampleSiebel`, qui s'exécute comme seconde instance de Test Connector. L'objet est

affiché dans Test Connector. Quand Reply Success est sélectionné à partir de l'application SampleSiebel, l'événement est renvoyé à SERVICE_ASYNCH_Order_Collab.

Pour exécuter le scénario asynchrone :

1. Démarrez le courtier d'intégration ICS s'il n'est pas déjà en cours d'exécution.
2. Démarrez le connecteur HTTP.
3. Démarrez deux instances de Test Connector.
4. A l'aide de Test Connector, définissez un profil pour SampleSAPConnector et pour SampleSiebelConnector.
5. Vous devez sauvegarder la définition de connecteur dans un fichier afin d'émuler un connecteur utilisant Test Connector. Pour sauvegarder une définition de connecteur dans un fichier :
   a. Ouvrez la définition de connecteur dans Connector Configurator.
   b. Sélectionnez **Fichier** > **Enregistrer sous** > Vers fichier dans la barre de menus.
   c. Accédez au répertoire dans lequel vous souhaitez enregistrer le fichier, entrez un nom dans la zone **Nom du fichier**, assurez-vous que la valeur Configuration (*.cfg) figure dans le menu déroulant Sauvegarder comme type, puis cliquez sur **Sauvegarder**.
6. Sélectionnez **FILE->CONNECT AGENT** dans chaque menu Test Connector pour commencer la simulation des agents.
7. Lors de la simulation de SampleSAPConnector à l'aide de Test Connector, sélectionnez **EDIT->LOAD BO** dans le menu. Chargez le fichier suivant :

   WBI_folder\connectors\HTTP\samples\WebSphereICS\CLIENT_ASYNCH_TLO_Order.bo

   Test Connector doit indiquer que CLIENT_ASYNCH_TLO_Order est chargé.
8. Vérifiez l'URL HTTP :
   - Pour exécuter l'exemple HTTP :
     a. Dans Test Connector, assurez-vous que la valeur de l'attribut Handler de l'objet métier CLIENT_ASYNCH_TLO_Order est réglé sur http.
     b. Développez l'attribut Request de CLIENT_ASYNCH_TLO_Order. Cet attribut est de type objet métier CLIENT_ASYNCH_Order.
     c. Développez l'attribut HTTPCfgMO de XML_Order. Cet attribut est de type XML_Order_HTTP_CfgMO.
     d. Assurez-vous que la valeur de l'attribut Destination de XML_Order_HTTP_CfgMO est réglé sur http://localhost:8080/wbia/http/samples.
   - **Pour exécuter l'exemple HTTPS**
     a. Assurez-vous que la valeur de l'attribut Handler de l'objet métier CLIENT_ASYNCH_TLO_Order est réglé sur http même s'il s'agit d'un appel HTTPS.
     b. Développez l'attribut Request de CLIENT_ASYNCH_TLO_Order. Cet attribut est de type objet métier XML_Order.
     c. Développez l'attribut HTTPCfgMO de XML_Order. Cet attribut est de type XML_Order_HTTP_CfgMO.
     d. Assurez-vous que la valeur de l'attribut Destination de XML_Order_HTTP_CfgMO est réglé sur https://localhost:443/wbia/http/samples.

9. Pendant la simulation de SampleSAPConnector avec Test Connector, cliquez sur l'objet métier **Test BO** chargé. Sélectionnez **REQUEST->SEND** dans le menu. Pour en savoir plus sur le flux de l'événement, reportez-vous au déroulement du scénario évoqué précédemment.
10. Pendant la simulation de SampleSiebelConnector avec Test Connector, sélectionnez **REQUEST->ACCEPT REQUEST**. Un événement libellé SERVICE_ASYNCH_TLO_Order.Create s'affiche dans le volet droit de Test Connector.
11. Cliquez deux fois sur l'objet métier. L'objet métier s'ouvre dans une fenêtre.
12. Développez l'attribut Request de l'objet métier. L'attribut Request est de type SERVICE_ASYNCH_Order. Inspectez les attributs OrderId, CustomerId et autres de SERVICE_ASYNCH_Order de la commande Order reçue. Cela termine l'exécution du scénario asynchrone.
13. Après avoir inspecté l'objet métier, refermez la fenêtre. Sélectionnez **REQUEST ->REPLY-> SUCCESS**.

Exécution du scénario synchrone

Ce scénario appelle le service HTTP Synch OrderStatus. Avant d'exécuter le scénario, examinez soigneusement son flux de données.

1. Un événement CLIENT_SYNCH_TLO_OrderStatus.Retrieve se produit dans l'application SampleSAP qui s'exécute dans une instance de Test Connector.
2. L'événement est envoyé de SampleSAP vers la collaboration CLIENT_SYNCH_OrderStatus_Collab.
3. L'événement est ensuite envoyé de la collaboration vers le connecteur HTTP.
4. Le connecteur HTTP recherche l'objet XML_OrderStatus, qui est enfant de requête de l'objet CLIENT_SYNCH_TLO_OrderStatus.
5. Le connecteur HTTP appelle le gestionnaire de données XML pour convertir l'objet métier XML_OrderStatus en message XML.
6. La demande XML est envoyée à l'URL par la méthode POST. Comme indiqué précédemment, le même HTTPConnector écoute la demande XML sur la même URL. Le programme d'écoute de protocole du connecteur reçoit le message XML.
7. Le programme d'écoute de protocole du connecteur convertit le message XML en XML_OrderStatus et crée ensuite un objet SERVICE_SYNCH_TLO_Order. L'objet XML_OrderStatus est défini comme enfant de l'objet SERVICE__SYNCH_TLO_Order.
8. Le connecteur HTTP envoie alors de façon synchrone l'objet SERVICE_SYNCH_TLO_OrderStatus à la collaboration SERVICE_SYNCH_OrderStatus_Collab s'exécutant dans WebSphere ICS. Comme l'exécution est synchrone, le connecteur HTTP reste bloqué jusqu'à ce que la collaboration s'exécute et renvoie la réponse.
9. Le connecteur HTTP envoie alors de façon synchrone l'objet SERVICE_TLO_OrderStatus à la collaboration SERVICE_SYNCH_OrderStatus_Collab s'exécutant dans WebSphere ICS. Comme l'exécution est synchrone, le connecteur HTTP reste bloqué jusqu'à ce que la collaboration s'exécute et renvoie la réponse.
10. Après avoir modifié les valeurs et sélectionné Reply Success depuis l'application SampleSiebel, l'événement est renvoyé à la collaboration SERVICE_SYNCH_OrderStatus_Collab.

11. SERVICE_SYNCH_OrderStatus_Collab reçoit l'objet SERVICE_SYNCH_TLO_OrderStatus. La collaboration envoie alors l'objet métier à HTTPConnector.
12. HTTPConnector recherche l'objet métier XML_OrderStatus qui est enfant de SERVICE_SYNCH_OrderStatus_TLO. Cet objet métier est converti en message de réponse XML par le gestionnaire de données XML.
13. La réponse XML est renvoyée au client HTTP.
14. Le client HTTP, qui dans ce cas est le gestionnaire de protocole du connecteur HTTP, reçoit la réponse. Le connecteur appelle le gestionnaire de données XML avec le message de réponse. Le gestionnaire de données XML convertir le message de réponse en objet métier XML_OrderStatus. HTTPConnector définit cet objet comme enfant de CLIENT_SYNCH_OrderStatus_TLO.
15. CLIENT_SYNCH_OrderStatus_TLO est renvoyé à la collaboration CLIENT_SYNCH_OrderStatus_Collab.
16. CLIENT_SYNCH_OrderStatus_Collab envoie ensuite CLIENT_SYNCH_OrderStatus_TLO à l'application SampleSAP, qui s'exécute comme première instance de Test Connector. Test Connector affiche cet objet.

Pour exécuter le scénario synchrone :

1. Démarrez le courtier d'intégration ICS s'il n'est pas déjà en cours d'exécution.
2. Démarrez le connecteur HTTP.
3. Démarrez deux instances de Test Connector.
4. A l'aide de Test Connector, définissez un profil pour SampleSAPConnector et pour SampleSiebelConnector.
5. Vous devez sauvegarder la définition de connecteur dans un fichier afin d'émuler un connecteur utilisant Test Connector. Pour sauvegarder une définition de connecteur dans un fichier :
   a. Ouvrez la définition de connecteur dans Connector Configurator.
   b. Sélectionnez **Fichier** > **Enregistrer sous** > Vers fichier dans la barre de menus.
   c. Accédez au répertoire dans lequel vous souhaitez enregistrer le fichier, entrez un nom dans la zone **Nom du fichier**, assurez-vous que la valeur Configuration (*.cfg) figure dans le menu déroulant Sauvegarder comme type, puis cliquez sur **Sauvegarder**.
6. Sélectionnez **FILE->CONNECT AGENT** dans chaque menu Test Connector pour commencer la simulation des agents.
7. Lors de la simulation de SampleSAPConnector à l'aide de Test Connector, sélectionnez **EDIT->LOAD BO** dans le menu. Chargez le fichier suivant :

   WBI_folder\connectors\HTTP\samples\WebSphereICS\CLIENT_SYNCH_TLO_OrderStatus.bo

   Test Connector doit indiquer que CLIENT_SYNCH_TLO_OrderStatus est chargé.
8. Vérifiez l'URL HTTP :
   - **Pour exécuter l'exemple HTTP** :
     a. Dans Test Connector, assurez-vous que la valeur de l'attribut Handler de l'objet métier CLIENT_SYNCH_TLO_OrderStatus est réglé sur http.
     b. Développez l'attribut Request de CLIENT_SYNCH_TLO_OrderStatus. Cet attribut est de type objet métier XML_OrderStatus.
     c. Développez l'attribut HTTPCfgMO de XML_OrderStatus. Cet attribut est de type XML_Order_HTTP_CfgMO.

d. Assurez-vous que la valeur de l'attribut Destination de XML_Order_HTTP_CfgMO est réglé sur http://localhost:8080/wbia/http/samples.

- **Pour exécuter l'exemple HTTPS** :
  a. Dans Test Connector, assurez-vous que la valeur de l'attribut Handler de l'objet métier CLIENT_SYNCH_TLO_OrderStatus est réglé sur http même s'il s'agit d'un appel https.
  b. Développez l'attribut Request de CLIENT_SYNCH_TLO_OrderStatus. Cet attribut est de type objet métier XML_OrderStatus.
  c. Développez l'attribut HTTPCfgMO de XML_OrderStatus. Cet attribut est de type XML_Order_HTTP_CfgMO.
  d. Assurez-vous que la valeur de l'attribut Destination de XML_Order_HTTP_CfgMO est réglé sur https://localhost:443/wbia/http/samples.

9. Pendant la simulation de SampleSAPConnector avec Test Connector, cliquez sur l'objet métier **Test BO** chargé. Sélectionnez **REQUEST->SEND** dans le menu. Pour en savoir plus sur le flux de données, reportez-vous au déroulement du scénario évoqué précédemment.
10. Un événement libellé SERVICE_SYNCH_TLO_OrderStatus.Retrieve s'affiche dans le volet droit de l'instance Test Connector simulant SampleSiebelConnector. Cliquez deux fois sur l'objet métier pour l'afficher dans une fenêtre.
11. Développez l'attribut Request de l'objet métier. Vérifiez l'intégrité des valeurs de demande transmises depuis SampleSAPConnector.
12. Dans la même fenêtre ouverte à l'étape 10 précédemment, attribuez une valeur à l'attribut de réponse de cet objet métier en sélectionnant **LOAD BO**. Chargez le fichier suivant :
    - WBI_folder\connectors\HTTP\samples\WebSphereICS\SERVICE_SYNCH_TLO_OrderStatus.bo

    Test Connector doit indiquer que SERVICE_SYNCH_TLO_OrderStatus est chargé.
13. Sélectionnez **REQUEST ->REPLY-> SUCCESS**.
14. Un événement libellé CLIENT_SYNCH_TLO_OrderStatus.Retrieve s'affiche dans le volet droit de l'instance Test Connector simulant SampleSAPConnector.
15. Cliquez deux fois sur l'objet métier **CLIENT_SYNCH_TLO_OrderStatus.Retrieve** qui s'affiche alors dans une fenêtre. Si SampleSiebelConnector a renvoyé un état de commande, l'attribut Response de l'objet métier comporte une valeur. Développez l'attribut **Response** pour vérifier l'état de commande.
16. Après avoir inspecté l'objet métier, refermez la fenêtre. Sélectionnez **REQUEST ->REPLY-> SUCCESS**.

Cela termine l'exécution du scénario synchrone.

Annexe D. Configuration de HTTPS/SSL

Si vous envisagez d'utiliser SSL, vous devez recourir à un logiciel tiers de gestion des magasins de clés, des certificats et de la génération de clés. Le connecteur HTTP n'offre pas d'outil pour ces tâches. Cependant, vous pouvez choisir d'utiliser keytool, fourni avec IBM JRE, afin de créer des certificats auto-signés et de créer des magasins de clés.

En tant que gestionnaire de clés et de certificats, keytool vous permet d'administrer vos paires de clés publique/privée et les certificats associés. Ces ressources sont utilisées pour l'auto-authentification (pour vous authentifier auprès d'autres utilisateurs ou services) ou l'intégrité et l'authentification de données employant des signatures numériques. L'utilitaire keytool permet de stocker les clés publiques (sous la forme de certificats) pour les homologues avec lesquels vous communiquez.

Cette annexe décrit la procédure de configuration des magasins de clés à l'aide de keytool. Elle est proposée à titre d'exemple et ne vise pas à remplacer la documentation de keytool ou de produits connexes. La documentation source de ces outils est à privilégier lors de la configuration de magasins de clés. Pour plus d'informations sur keytool, voir :

- http://java.sun.com/j2se/1.3/docs/tooldocs/tools.html#security

Configuration de magasins de clés

Pour créer KeyStore à l'aide de keytool, vous devez tout d'abord créer une paire de clés dans le magasin de clés. Par exemple, si vous entrez la commande suivante :

```
keytool -genkey -alias httpadapter -keystore c:\security\keystore
```

keytool vous demande alors un mot de passe immédiatement. Vous pouvez entrer un mot de passe de votre choix (dans les paramètres de keytool), mais devez indiquer le mot de passe entré dans keytool dans la propriété de connecteur SSL" KeyStorePassword. Pour plus d'informations, voir «KeyStorePassword», à la page 78.

La commande exemple crée le magasin de clés `keystore` dans le répertoire `c:\security\keystore`. Ainsi, vous saisirez `c:\security\keystore` comme valeur de la propriété hiérarchique de connecteur SSL" KeyStore. Toujours en reprenant l'exemple précédent, vous entrerez `-alias httpadapter` comme valeur de la propriété hiérarchique de connecteur SSL" KeyStoreAlias. keytool vous demande alors d'entrer des informations détaillées sur le certificat. Voici que vous pourriez entrer à chaque invite. (Consultez la documentation de keytool.)

```
What is your first and last name?
     [Unknown]:  HostName
What is the name of your organizational unit?
     [Unknown]:  wbi
What is the name of your organization?
     [Unknown]:  IBM
```

```
What is the name of your City or Locality?
     [Unknown]:  Burlingame
What is the name of your State or Province?
     [Unknown]:  CA
What is the two-letter country code for this unit?
    [Unknown]:  US
Is <CN=HostName, OU=wbi, O=IBM, L=Burlingame,
ST=CA, C=US> correct?
    [no]:  yes
```

keytool vous invite alors à saisir un mot de passe :

```
Entrez le mot de passe de la clé de <httpadapter> (appuyez sur RETOUR si le mot
de passe est identique à celui du magasin de clés) :
```

Appuyez sur Retour pour utiliser le même mot de passe. Si vous souhaitez utiliser un certificat auto-signé, vous pouvez exporter le certificat créé précédemment. Dans ce cas, entrez ce qui suit sur la ligne de commande :

```
keytool -export -alias httpadapter -keystore c:\security\keystore -file
wsadapter.cer
```

keytool vous demande alors le mot de passe du magasin de clés. Entrez le mot de passe saisi précédemment.

Configuration de TrustStore

Vous pouvez souhaiter configurer TrustStore pour ce qui suit :

Si vous souhaitez que le programme d'écoute de protocole HTTPS authentifie le client, réglez la propriété de configuration de connecteur SSL" UseClientAuth sur `true`. Dans ce cas, le fichier TrustStore doit contenir les certificats de tous les clients approuvés pour être exploité par le programme d'écoute de protocole HTTPS. Notez que le connecteur utilise le mécanisme par défaut de JSSE pour faire confiance aux clients.

Si vous appelez des services HTTPS, le gestionnaire de protocole HTTP-HTTPS requiert que TrustStore fasse confiance au service. Cela signifie que TrustStore doit contenir les certificats de tous les services HTTP dignes de confiance. Notez que le connecteur utilise le mécanisme par défaut de JSSE pour faire confiance aux clients.Pour importer les certificats approuvés dans TrustStore, entrez une commande similaire :

```
keytool -import -alias trusted1 -keystore c:\security\truststore -file
 c:\security\trusted1.cer
```

keytool vous demande alors le mot de passe du magasin de clés. Si vous avez saisi `-keystore c:\security\truststore`, vérifiez que la propriété hiérarchique SSL->TrustStore est réglée sur `c:\security\truststore`. Réglez également la propriété hiérarchique SSL->TrustStorePassword sur le mot de passe que vous avez saisi précédemment.

Génération d'une demande de signature de certificat (CSR) pour des certificats de clé publique

Si l'échange de données SSL s'effectue entre partenaires dignes de confiance dont l'identité n'est pas à prouver, les certificats auto-signés peuvent être une solution adéquate. Cependant, il est plus évident de faire confiance à un certificat quand il est signé par une autorité de certification (CA).

Pour obtenir un certificat signé par la CA à l'aide de keytool, vous devez tout d'abord générer une demande CSR, puis attribuez celle-ci à une CA. La CA signe alors le certificat et le renvoie.

Pour générer une demande CSR, entrez la commande suivante :

```
keytool -certreq -alias wsadapter -file httpadapter.csr
 -keystore c:\security\keystore
```

Dans la commande, `alias` correspond à l'alias du magasin de clés que vous avez créé pour la clé privée. keytool génère le fichier CSR que vous transmettez à votre CA. La CA vous transmet alors le certificat signé. Vous devrez importer ce certificat dans le magasin de clés. Pour cela, il est nécessaire de saisir la commande suivante :

```
keytool -import -alias wsadapter -keystore c:\security\keystore -trustcacerts
-file casignedcertificate.cer
```

Après l'importation, le certificat auto-signé dans le magasin de clés est remplacé par le certificat signé par la CA.

Annexe E. Common Event Infrastructure

WebSphere Business Integration Server Foundation inclut Common Event Infrastructure Server Application, nécessaire au fonctionnement de Common Event Infrastructure. WebSphere Application Server Foundation peut être installé sur tout système (pas nécessairement sur la même machine que l'adaptateur.)

WebSphere Application Server Application Client contient les bibliothèques nécessaires à l'interaction entre l'adaptateur et Common Event Infrastructure Server Application. Vous devez installer WebSphere Application Server Application Client sur le même système que l'adaptateur. L'adaptateur se connecte à WebSphere Application Server (dans WebSphere Business Integration Server Foundation) par le biais d'une URL configurable.

La prise en charge de Common Event Infrastructure est possible avec tout courtier d'intégration supporté par cette édition.

Logiciels requis

En plus des logiciels exigés pour l'adaptateur, les applications suivantes doivent être installées pour que Common Event Infrastructure puisse fonctionner :

- WebSphere Business Integration Server Foundation 5.1.1
- WebSphere Application Server Application Client 5.0.2, 5.1 ou 5.1.1.

  (WebSphere Application Server Application Client 5.1.1 est fourni avec WebSphere Business Integration Server Foundation 5.1.1.)

Remarque : Common Event Infrastructure n'est pris en charge par aucune plateforme HP-UX ou Linux.

Activation de Common Event Infrastructure

La fonctionnalité Common Event Infrastructure est activée avec les propriétés standard `CommonEventInfrastructure` et `CommonEventInfrastructureContextURL`, configurées avec Connector Configurator. Par défaut, Common Event Infrastructure n'est pas activé. La propriété `CommonEventInfrastructureContextURL` vous permet de configurer l'URL du serveur Common Event Infrastructure. (Pour plus d'informations, voir l'annexe "Propriétés standard" du présent document.)

Obtention d'événements d'adaptateur Common Event Infrastructure

Si Common Event Infrastructure est activé, l'adaptateur génère des événements Common Event Infrastructure qui se mappent sur les événements d'adaptateur suivants :

- Démarrage de l'adaptateur
- Arrêt de l'adaptateur
- Réponse de l'application à un dépassement de délai de l'agent de l'adaptateur
- Tout appel `doVerbFor` émis depuis l'agent de l'adaptateur
- Appel `gotApplEvent` depuis l'agent de l'adaptateur

Pour qu'une autre application (l'"application client") puisse recevoir les événements Common Event Infrastructure générés par l'adaptateur, elle doit

utiliser le catalogue d'événements Common Event Infrastructure pour déterminer les définitions des événements appropriés et leurs propriétés. Pour que l'application client puisse utiliser les événements de l'application d'origine, les événements doivent être définis dans le catalogue d'événements.

L'annexe "Définitions du catalogue des événements Common Event Infrastructure" du présent document contient des métadonnées au format XML. Elles indiquent, pour les adaptateurs WebSphere Business Information, les descripteurs d'événements et les propriétés que l'application client doit rechercher.

Pour plus d'informations

Pour plus d'informations sur Common Event Infrastructure, voir la documentation WebSphere Business Integration Server Foundation, disponible à l'adresse :

http://publib.boulder.ibm.com/infocenter/ws51help

Pour consulter des exemples de métadonnées XML indiquant les propriétés et descripteurs d'événement générés par l'adaptateur qu'une application client doit utiliser, voir «Définitions du catalogue d'événements Common Event Infrastructure».

Définitions du catalogue d'événements Common Event Infrastructure

Le catalogue d'événements Common Event Infrastructure contient des définitions d'événements qui peuvent être interrogées par d'autres applications. Ci-dessous figurent des exemples de définitions d'événements utilisant des métadonnées XML pour des événements d'adaptateur types. Si vous écrivez une autre application, elle peut utiliser les interfaces de catalogue d'événements pour interroger la définition de l'événement. Pour plus d'informations sur les définitions d'événements et savoir comment les interroger, voir la documentation Common Event Infrastructure disponible dans le Centre de documentation en ligne IBM WebSphere Server Foundation.

Pour les adaptateurs WebSphere Business Integration, les éléments de données étendues qui doivent être définis dans le catalogue des événements sont les clés de l'objet métier. Chaque clé d'objet métier exige une définition d'événement. Par conséquent, pour un adaptateur donné, les divers événements tels que start adapter, stop adapter, timeout adapter et tout événement`doVerbFor` (comme create, update ou delete) doivent avoir une définition dans le catalogue d'événements.

Les sections suivantes contiennent des exemples des métadonnées pour start adapter, stop adapter et la requête ou la livraison d'événement.

Format XML des métadonnées de "start adapter"

```
<eventDefinition name="startADAPTER"
      parent="event">
    <property name ="creationTime" //Commentaire : Par exemple, la valeur peut être
 "2004-05-13T17:00:16.319Z"
         required="true" />
    <property name="globalInstanceId" //Commentaire : Généré automatiquement
 par Common Event Infrastructure
         required="true"/>
    <property name="sequenceNumber"    //Commentaire : Numéro défini par la source
pour les messages qui seront envoyés/triés de façon logique
         required="false"/>
    <property name="version"    //Commentaire : Version de l'événement
```

```
         required="false"
         defaultValue="1.0.1"/>
    <property name="sourceComponentId"
         path="sourceComponentId"
         required="true"/>
    <property name="application"   //Commentaire : Le name#version de
l'application source générant l'événement. Par exemple,  "SampleConnector#3.0.0"
         path="sourceComponentId/application"         required="false"/>
    <property name="component"   //Commentaire : Ce sera le name#version
 du composant source.
         path="sourceComponentId/component"
         required="true"
         defaultValue="ConnectorFrameWorkVersion#4.2.2"/>
    <property name="componentIdType"    //Commentaire : Précise le format
et la signification du composant
          path="sourceComponentId/componentIdType"
          required="true"
          defaultValue="Application"/>
    <property name="executionEnvironment"
 //Commentaire : Identifie l'environnement dans lequel l'application est exécutée
...par exemple "Windows 2000#5.0"
          path="sourceComponentId/executionEnvironment"
          required="false" />
     <property name="location"    //Commentaire : Cette valeur correspond au
 nom du serveur... par exemple "WQMI"
          path="sourceComponentId/location"
          required="true"/>
    <property name="locationType"   //Commentaire : Indique le format et
     la signification de l'emplacement
          path="sourceComponentId/locationType"
          required="true"
          defaultValue="Hostname"/>
     <property name="subComponent"     //Commentaire : Distinction supplémentaire
du composant logique
          path="sourceComponentId/subComponent"
          required="true"
          defaultValue="AppSide_Connector.AgentBusinessObjectManager"/>
     <property name="componentType"      //Commentaire : Nom correctement défini
utilisé pour caractériser toutes les instances de ce composant
          path="sourceComponentId/componentType"
          required="true"
          defaultValue="ADAPTER"/>
     <property name="situation"   //Commentaire : Définit le type de
 situation ayant entraîné le signalement de l'événement
          path="situation"
          required="true"/>
     <property name="categoryName="    //Commentaire : Indique le type
de situation pour l'événement
         path="situation/categoryName"
         required="true"
         defaultValue="StartSituation"/>
    <property name="situationType"    //Commentaire : Indique le type
de situation et la disposition de l'événement
         path="situation/situationType"
         required="true"
    <property name="reasoningScope" //Commentaire : Indique
 l'impact de l'événement
         path="situation/situationType/reasoningScope"
         required="true"
         permittedValue="INTERNAL"
         permittedValue="EXTERNAL"/>
    <property name="successDisposition" //Commentaire : Indique la
 réussite de l'événement
         path="situation/situationType/successDisposition"
         required="true"
         permittedValue="SUCCESSFUL"
         permittedValue="UNSUCCESSFUL" />
```

```
    <property name="situationQualifier"  //Commentaire : Indique les
 qualificatifs de la situation pour cet événement
         path="situation/situationType/situationQualifier"
         required="true"
         permittedValue="START_INITIATED"
         permittedValue="RESTART_INITIATED"
         permittedValue="START_COMPLETED" />
</eventDefinition>
```

Format XML des métadonnées de ″stop adapter″

Les métadonnées de “stop adapter” sont identiques à celles de “start adapter” aux exceptions suivantes près :

- La valeur par défaut de la propriété `categoryName` est `StopSituation` :

```
<property name="categoryName="
 //Commentaire : Indique le type
 de situation pour l'événement
              path="situation/categoryName"
              required="true"
              defaultValue="StopSituation"/>
```

- Les valeurs autorisées pour la propriété `situationQualifier` sont différentes et sont les suivantes pour “stop adapter” :

```
<property name="situationQualifier"
 //Commentaire : Précise les qualificatifs de la situation pour cet événement
           path="situation/situationType/situationQualifier"
           required="true"
           permittedValue="STOP_INITIATED"
           permittedValue="ABORT_INITIATED"
           permittedValue="PAUSE_INITIATED"
           permittedValue="STOP_COMPLETED"
 />
```

Format XML des métadonnées de “timeout adapter”

Les métadonnées de “timeout adapter” sont identiques à celles de “start adapter” et “stop adapter” aux exceptions suivantes près :

- La valeur par défaut de la propriété `categoryName` est `ConnectSituation` :

```
<property name="categoryName="
 //Commentaire : Indique le type
 de situation pour l'événement
              path="situation/categoryName"
              required="true"
              defaultValue="ConnectSituation"/>
```

- Les valeurs autorisées pour la propriété `situationQualifier` sont différentes et sont les suivantes pour “timeout adapter” :

```
<property name="situationQualifier"  //Commentaire : Précise
 les qualificatifs de la situation pour cet événement
           path="situation/situationType/situationQualifier"
           required="true"
           permittedValue="IN_USE"
           permittedValue="FREED"
           permittedValue="CLOSED"
           permittedValue="AVAILABLE"
 />
```

Format XML des métadonnées de ″request″ ou ″delivery″

A la fin de ce format XML figurent les éléments de données étendues. Les éléments de données étendues des événements de livraison et de requête d'adaptateur sont les données de l'objet métier en cours de traitement. Ces données incluent le nom de l'objet métier, sa clé (étrangère ou locale) et les objets métier qui sont des enfants d'objets métier parents. Les objets métier enfants sont ensuite décomposés dans les mêmes données que le parent (nom, clé et tout objet métier enfant). Ces données sont représentées dans un élément de données étendues de la définition de l'événement. Ces données changeront selon l'objet métier, les clés et les objets métier enfants traités. Les données étendues de cette définition d'événement sont un simple exemple et représentent un objet métier nommé `Employee` avec une clé `EmployeeId` et un objet métier enfant `EmployeeAddress` avec une clé `EmployeeId`. Ce modèle peut s'appliquer à toutes les données d'un objet métier particulier.

```
<eventDefinition name="createEmployee"     //Commentaire : Ce
 nom d'extension est toujours l'instruction de l'objet métier suivi de
 son nom
          parent="event">
    <property name ="creationTime"  //Commentaire : Par exemple, la valeur
peut être "2004-05-13T17:00:16.319Z"
         required="true" />
    <property name="globalInstanceId" //Commentaire : Généré automatiquement
 par Common Event Infrastructure
         required="true"/>
    <property name="localInstanceId"    //Commentaire : La valeur est égale à
l'instruction de l'objet métier+nom de l'objet métier+#+nom app+ identificateur
de l'objet métier
         required="false"/>
    <property name="sequenceNumber"    //Commentaire : Numéro défini par la source
pour les messages qui seront envoyés/triés de façon logique
         required="false"/>
    <property name="version"  //Commentaire : Version de l'événement... la valeur
est définie sur 1.0.1
         required="false"
         defaultValue="1.0.1"/>
    <property name="sourceComponentId"
         path="sourceComponentId"
         required="true"/>
    <property name="application"   //Commentaire : Le name#version de
 l'application source qui génère l'événement... par exemple
 "SampleConnector#3.0.0"
         path="sourceComponentId/application"
         required="false"/>
    <property name="component"   //Commentaire : Ce sera le name#version
du composant source.
         path="sourceComponentId/component"
         required="true"
         defaultValue="ConnectorFrameWorkVersion#4.2.2"/>
    <property name="componentIdType"    //Commentaire : Précise le format
 et la signification du composant
         path="sourceComponentId/componentIdType"
         required="true"
         defaultValue="Application"/>
    <property name="executionEnvironment" //Commentaire : Identifie le
 environment#version dans lequel l'application est exécutée...
 par exemple "Windows 2000#5.0"
         path="sourceComponentId/executionEnvironment"
         required="false" />
    <property name="instanceId" //Commentaire : La valeur est égale à l'instruction
  de l'objet métier+nom de l'objet métier+#+nom app+identificateur de l'objet métier
          path="sourceComponentId/instanceId"
          required="false"
    <property name="location"    //Commentaire : Cette valeur correspond au
nom du serveur... par exemple "WQMI"
```

```
          path="sourceComponentId/location"
          required="true"/>
     <property name="locationType"   //Commentaire : Indique le format et
 la signification de l'emplacement
          path="sourceComponentId/locationType"
          required="true"
          defaultValue="Hostname"/>
     <property name="subComponent"  //Commentaire : Distinction supplémentaire du
 composant logique (dans ce cas, la valeur est égale au nom de l'objet
 métier)
          path="sourceComponentId/subComponent"
          required="true"/>
     <property name="componentType"      //Commentaire : Nom correctement défini utilisé
 pour caractériser toutes les instances de ce composant
          path="sourceComponentId/componentType"
          required="true"
          defaultValue="ADAPTER"/>
     <property name="situation"   //Commentaire : Définit le type de
situation ayant entraîné le signalement de l'événement
          path="situation"
          required="true"/>
    <property name="categoryName"    //Commentaire : Indique le type
 de situation pour l'événement
         path="situation/categoryName"
         required="true"
         permittedValue="CreateSituation"
         permittedValue="DestroySituation"
         permittedValue="OtherSituation" />
    <property name="situationType"    //Commentaire : Indique le type
de situation et la disposition de l'événement
         path="situation/situationType"
         required="true"
    <property name="reasoningScope" //Commentaire : Indique
l'impact de l'événement
         path="situation/situationType/reasoningScope"
         required="true"
         permittedValue="INTERNAL"
         permittedValue="EXTERNAL"/>
    <property name="successDisposition" //Commentaire : Indique la
 réussite de l'événement
         path="situation/situationType/successDisposition"
         required="true"
         permittedValue="SUCCESSFUL"
         permittedValue="UNSUCCESSFUL" />
    <extendedDataElements name="Employee" //Commentaire : Nom de l'objet métier
 lui-même
            type="noValue"
            <children name="EmployeeId"
                 type="string"/>  //Commentaire : Le type est l'une des
 valeurs autorisées dans la documentation Common Event Infrastructure
            <children name="EmployeeAddress"
                 type="noValue"/>
                    <children name="EmployeeId"
                        type="string"/>
                     -
                     -
                     -
    </extendedDataElements
</eventDefinition>
```

Annexe F. Application Response Measurement

Cet adaptateur est compatible avec l'API Application Response Measurement, qui permet de gérer la disponibilité, les accords de niveau de service et la planification de la capacité des applications. Une application avec appels ARM peut participer à IBM Tivoli Monitoring for Transaction Performance, autorisant ainsi la collecte et la consultation des données relatives aux indicateurs de transaction.

Prise en charge des appels Application Response Measurement

Cet adaptateur est compatible avec l'API Application Response Measurement, qui permet de gérer la disponibilité, les accords de niveau de service et la planification de la capacité des applications. Une application avec appels ARM peut participer à IBM Tivoli Monitoring for Transaction Performance, autorisant ainsi la collecte et la consultation des données relatives aux indicateurs de transaction.

Logiciels requis

En plus des logiciels exigés par l'adaptateur, les applications suivantes doivent être installées pour qu'ARM puisse fonctionner :

- WebSphere Application Server 5.0.1 (contient le serveur IBM Tivoli Monitoring for Transaction Performance). Il ne doit pas forcément être installé sur le même système que l'adaptateur.
- IBM Tivoli Monitoring for Transaction Performance v. 5.2 Fixpack 1. Il doit être installé sur le même système que l'adaptateur, et configuré pour pointer vers la machine qui héberge le serveur IBM Tivoli Monitoring for Transaction Performance.

La prise en charge de Application Response Measurement est possible avec tout courtier d'intégration supporté par cette édition.

Remarque : Les appels Application Response Measurement sont pris en charge par tous les systèmes d'exploitation supportés par la présente édition de IBM WebSphere Business Integration Adapters, *sauf* par HP-UX (toutes versions) et Red Hat Linux 3.0.

Activation de Application Response Measurement

Les appels ARM sont activés en définissant sur "True" la propriété standard `TivoliMonitorTransactionPerformance` de Connector Configurator. Par défaut, la prise en charge ARM est désactivée. (Pour plus d'informations, voir l'annexe "Propriétés standard" de ce document.)

Surveillance des transactions

Lorsque ARM est activé, les transactions surveillées sont les événements de service et les livraisons d'événements. La transaction est mesurée à partir du début de la demande de service ou de la livraison d'événement et jusqu'à la fin. Le nom de la transaction, affiché sur la console Tivoli Monitoring for Transaction Performance, commencera par `SERVICE REQUEST` ou `EVENT DELIVERY`. La suite du nom sera l'instruction de l'objet métier (comme `CREATE`, `RETRIEVE`, `UPDATE` ou `DELETE`). La fin du nom sera le nom de l'objet métier, comme "`EMPLOYEE`." Par exemple, le nom d'une transaction pour une livraison d'événement correspondant à la création d'un employé pourra être `EVENT DELIVERY CREATE EMPLOYEE` ou `SERVICE REQUEST UPDATE ORDER`.

Les indicateurs ci-après sont collectés par défaut pour chaque type de demande de service ou de livraison d'événement :

- Durée minimale de transaction
- Durée maximale de transaction
- Durée moyenne de transaction
- Total d'exécutions de la transaction

Vous (ou l'administrateur système de WebSphere Application Server) pouvez sélectionner les indicateurs à afficher, et pour quels événements d'adaptateur, en configurant Discovery Policies et Listener Policies pour des transactions données, depuis la console Tivoli Monitoring for Transaction Performance. (Voir «Pour plus d'informations».)

Pour plus d'informations

Pour plus de détails, voir la documentation IBM Tivoli Monitoring for Transaction Performance. Pour obtenir des informations sur la surveillance et la gestion des indicateurs générés par l'adaptateur, voir *IBM Tivoli Monitoring for Transaction Performance User's Guide*.

Index

R

S

T

Informations légales

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

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

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

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

Pour le Canada, veuillez adresser votre courrier à :

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

LICENCE DE COPYRIGHT :

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

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

Informations sur les interfaces de programmation

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

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

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

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

Marques et marques de service

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

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

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

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

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

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

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

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

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

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

WebSphere Business Integration Adapter Framework version 2.6.0.3