MQSeries® Adapter Kernel for Multiplatforms

Mise en route

version 1.1

GC11-1809-01

MQSeries® Adapter Kernel for Multiplatforms

Mise en route

version 1.1

GC11-1809-01

Important

Avant d'utiliser le présent document et le produit associé, prenez connaissance des informations générales figurant à la section «Remarques» à la page 131.

Sixième édition - mai 2001

Réf. US : GC34-5855-05

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

Figures

Tableaux

Bienvenue dans le Guide d'Initiation de MQSeries Adapter Kernel

Ce document présente MQSeries® Adapter Kernel et explique comment planifier, installer et utiliser le produit.

Pour pouvoir utiliser le noyau, effectuez les opérations générales suivantes :

1. Consultez «Chapitre 1. Information produit MQSeries Adapter Offering» à la page 1.
2. Préparez l'installation. Pour plus d'informations, reportez-vous à la section «Préparation de l'installation» à la page 42.
3. Installez le noyau. Pour plus d'informations, reportez-vous à la section «Installation du noyau» à la page 43.
4. Vérifiez l'installation. Pour plus d'informations, reportez-vous à la section «Vérification de l'installation» à la page 49.
5. Configurez le noyau. Pour plus d'informations, reportez-vous à la section «Configuration du noyau» à la page 62.
6. Si vous le désirez, configurez le logiciel en option pour le faire fonctionner avec le noyau. Pour plus d'informations, reportez-vous à la se ction «Configuration de MQSeries et MQSeries Integrator» à la page 96.
7. Créez vos adaptateurs à l'aide de MQSeries Adapter Builder, puis testez-les et déployez-les. Pour plus de détails, voir la documentation de MQSeries Adapter Builder.
8. Démarrez le noyau. Pour plus d'informations, reportez-vous à la section «Démarrage du noyau» à la page 97.

Pour utiliser ces informations, vous devez également connaître les conditions préalables et les produits en option. Reportez-vous à la section «Chapitre 2. Planification de l'installation du noyau» à la page 33. Reportez-vous également à la section «Références» à la page 105.

Qui doit utiliser ces informations

Ces informations sont destinées à ceux qui doivent planifier, installer, ou utiliser MQSeries Adapter Kernel.

Informations associées

Vous trouverez des informations supplémentaires dans les éléments suivants :

- Le fichier `readme.txt`. Ce fichier peut contenir des informations devenues disponibles après l'achèvement de ce manuel. Avant l'installation, le fichier `readme.txt` se trouve dans le répertoire principal du CD-ROM du produit. Après l'installation, le fichier `readme.txt` se trouve dans le répertoire principal d'installation de MQSeries Adapter Kernel.
- Le *Problem Determination Guide* type de document GC34-5897, décrivant les outils, y compris le programme trace, destinés à résoudre les problèmes spécifiques rencontrés avec MQSeries Adapter Kernel. Le *Problem Determination Guide* est disponible dans le centre d'aide et d'informations de MQSeries Adapter Kernel, qui est installé avec le produit.
- La documentation en ligne de l'interface de programmation de l'application (API) fournie dans le centre d'aide et d'information de MQSeries Adapter Kernel. Ces informations sont fournies uniquement à titre d'aide pour la compréhension du fonctionnement du noyau et comme aide aux diagnostics. Pour plus d'informations, reportez-vous à la section «Chapitre 5. Utilisation des Interfaces de Programmation d'Applications (API) de MQSeries Adapter Kernel» à la page 103.
- Informations relatives à MQSeries Adapter Builder, comprenant les manuels et le système d'aide.
- Site web de la famille de produits MQSeries à l'adresse : www.ibm.com/software/ts/mqseries/.

  En suivant les liens à partir de ce site web, vous pouvez :

  – Obtenir les dernières informations concernant la famille de produits MQSeries, y compris MQSeries Adapter Offering.

  – Accéder aux manuels MQSeries en format HTML et PDF, comprenant éventuellement une édition plus récente de ces manuels.

  – Télécharger MQSeries SupportPacs.

Conventions

La documentation relative à MQSeries Adapter Kernel utilise les conventions typographiques et d'entrée suivantes.

Tableau 1. Conventions utilisées dans ce manuel

Convention	Signification
Gras	Indique les noms de commande. Lorsqu'il s'agit d'interfaces graphiques utilisateur (GUI), indique les menus, les options de menus, les rubriques et les boutons.
`Espace simple`	Indique le texte que vous devez entrer au niveau d'une invite et les valeurs que vous devez utiliser littéralement, telles que les noms de fichiers, les chemins d'accès et les éléments des langages de programmation tels que fonctions, classes et méthodes. Espace simple indique également des exemples de code et de texte écran.
Italique	Indique les valeurs des variables que vous devez fournir (par exemple, vous fournissez le nom d'un fichier pour *Nom de fichier*). Italique indique également la mise en évidence et les titres des manuels.
`%`	Représente l'invite shell de commandes UNIX pour une commande n'exigeant pas de droits d'accès de niveau superutilisateur.
`#`	Représente l'invite shell de commandes UNIX pour une commande exigeant des droits d'accès de niveau superutilisateur.
`C:\>`	Représente les invites de commandes sur les systèmes Windows®.
>	Lorsqu'utilisé pour décrire un menu, affiche une série de sélections de menus. Par exemple, "Cliquer**Fichier > Nouveau**" signifie "Du menu **Fichier**, cliquer sur la commande **Nouveau**."
Entrée de commandes	Lorsqu'il vous est demandé d'"entrer" ou d'"émettre" une commande, tapez la commande et appuyez sur Retour. Par exemple, l'instruction "Entrer la commande **ls** " signifie taper **ls** au niveau d'une invite de commande, puis appuyer sur Retour.
[ ]	Délimiter des éléments facultatifs par des crochets dans des descriptions syntaxiques.
{ }	Délimiter des listes dans lesquelles vous devez choisir un élément par des accolades dans des descriptions syntaxiques.
|	Séparer les éléments d'une liste d'options délimités par des accolades ({ }) dans des descriptions syntaxiques.
...	Les points de suspension dans les descriptions syntaxiques indiquent que vous pouvez répéter l'élément précédent une ou plusieurs fois. Les points de suspension indiquent que des informations sont omises dans l'exemple par souci de brièveté.

Remarque : Le terme Epic apparaît dans certaines valeurs et noms du logiciel du noyau dans ce manuel. En ce qui concerne MQSeries Adapter Offering, ce terme n'a aucune signification par lui-même.

Sommaire des modifications

La sixième édition (l'édition actuelle) comprend les ajouts et modifications suivants par rapport à la cinquième édition :

- Mise à jour de la description du déroulement de l'exécution pour prendre en compte diverses modifications. Reportez-vous à la section«Déroulement de l'exécution» à la page 15.
- Informations sur l'utilisation de MQSeries Adapter Kernel avec WebSphere® Business Integrator. Pour plus d'informations, reportez-vous au «Utilisation de MQSeries Adapter Kernel avec WebSphere Business Integrator et WebSphere Application Server» à la page 29.
- Informations sur le niveau de support en langue nationale fourni avec divers types d'adaptateurs. Pour plus d'informations, reportez-vous au «Support en langue nationale» à la page 31.
- Clarification des instructions d'installation. Reportez-vous à la section «Installation du noyau» à la page 43.
- Informations sur l'installation automatique. Pour plus d'informations, reportez-vous à la section «Installation automatique» à la page 54.
- Présentation conceptuelle de la configuration pour vous aider à configurer le noyau. Pour plus d'informations, reportez-vous au «Généralités sur la configuration» à la page 63.
- Informations sur les nouvelles valeurs d'en-tête. Pour plus d'informations, reportez-vous à la section «En-tête descripteur de message MQSeries Adapter Kernel» à la page 115.

La cinquième édition comprenait les ajouts et modifications suivants par rapport à la quatrième édition :

- Des informations concernant l'utilisation du noyau sur les plates-formes Windows® 2000, OS/400®, HP-UX et Solaris. La prise en charge de ces plates-formes était nouvelle dans la version 1.1 de MQSeries Adapter Kernel. Le noyau n'était disponible auparavant que sur Windows NT® et AIX®.
- Les mises à jour de toutes les instructions d'installation doivent tenir compte de la version 1.1 de MQSeries Adapter Kernel.
- Informations concernant l'utilisation du fichier `aqmconfig.xml` pour la configuration de MQSeries Adapter Kernel. Le noyau était précédemment configuré à l'aide du fichier `aqmconfig.properties`. Pour plus d'informations, reportez-vous au «Le fichier de configuration» à la page 69.

- Informations sur les nouveaux modes de communication MQ et JMS (Java Message Service). Pour plus d'informations, reportez-vous à la section «Annexe A. Modes de communication» à la page 107.
- Les informations sur le traçage contenues dans ce document apparaissent désormais dans le nouveau document *Problem Determination Guide*. Reportez-vous à la section *Problem Determination Guide* pour plus d'informations.

Chapitre 1. Information produit MQSeries Adapter Offering

Le noyau IBM MQSeries Adapter Kernel fait partie d'un ensemble de produits d'intégration d'application qui, regroupés, sont appelés IBM MQSeries Adapter Offering. MQSeries Adapter Offering fonctionne avec l'application de messagerie MQSeries et d'autres services de messagerie pour vous permettre de limiter les risques, la complexité et le coût de la gestion de l'intégration point à point de vos opérations.

Dans l'intégration *point à point*, chaque application communique avec chacune des autres applications individuellement. Chaque interface est différente et il existe de nombreuses interfaces. Si une modification est effectuée dans une application, cela entraîne généralement des modifications au niveau de nombreuses interfaces. A mesure que le nombre d'applications s'accroît, le coût de l'intégration point à point augmente rapidement. L'intégration de chaque nouvelle application entraîne généralement un travail plus important que l'intégration de l'application précédente.

Grâce à MQSeries Adapter Offering, vous pouvez passer de l'utilisation de l'intégration point à point à celle de l'intégration *un à N*. Ce dernier type d'intégration offre les avantages suivants :

- Toutes les applications peuvent utiliser une interface commune.
- Les données provenant d'une *application source*, sous forme de*message*, sont *acheminées* vers une ou plusieurs*applications cible*.
- Une modification effectuée dans une application n'affecte généralement qu'une interface.
- L'utilisation d'une interface commune, d'application neutre—par exemple,une norme de l'industrie telle que Extensible Markup Language (XML)—peut être encore plus rentable. Le nombre d'applications pouvant être prises en charge est plus important et les efforts moindres.
- A mesure que le nombre d'applications s'accroît, la rentabilité de l'intégration un à N augmente. De manière générale, l'ajout de chaque nouvelle application ne nécessite pas de modifications importantes au niveau des interfaces de toutes les autres applications.
- Le travail d'intégration peut être automatisé et basé sur des modèles.

MQSeries Adapter Offering peut être déployé sans aucun changement d'applications ni d'opérations. Tout le travail d'intégration est généralement effectué dans MQSeries Adapter Offering, limitant ainsi le besoin d'écrire un code personnalisé.

Dans MQSeries Adapter Offering, l'interface vers ou en provenance d'une application est fournie par un *adaptateur*. Toutes les applications exigent au moins un adaptateur pour assurer l'interface entre l'environnement d'application et l'environnement de messagerie. Chaque adaptateur est propre à une application et à un type de message.

MQSeries Adapter Kernel peut éventuellement être déployé avec MQSeries Integrator pour le courtage et la transformation de messages. MQSeries Adapter Offering peut être complété par des offres de services d'IBM et autres.

Des exemples d'utilisation d'adaptateurs sont indiqués ci-dessous :

- Ajout d'un bordereau de vente.
- Synchronisation d'un enregistrement client.
- Synchronisation d'un enregistrement d'inventaire.
- Synchronisation d'un article.
- Synchronisation d'un bordereau de vente.

Phase de création et phase d'exécution

MQSeries Adapter Offering possède deux composants principaux, Adapter Builder (également appelé le générateur) et Adapter Kernel (également appelé le noyau). Cette section décrit ces composants ainsi que les adaptateurs qui sont créés et exécutés par Adapter Offering.

adaptateur

Logiciel fournissant une interface vers ou en provenance d'une application. Les adaptateurs sont créés à l'aide de MQSeries Adapter Builder. Généralement, chaque adaptateur est créé pour être spécifique à un type de message envoyé d'une application ou vers celle-ci. Les adaptateurs eux-mêmes ne font pas partie de MQSeries Adapter Offering.

Un adaptateur comprend un code source C ou Java qui permet la compilation dans une bibliothèque partagée. Lorsque les adaptateurs et MQSeriesAdapter Kernel sont lancés simultanément, ils exécutent les fonctions d'exécution de MQSeries Adapter Offering.

Suivant la manière dont MQSeries Adapter Builder est modélisé, l'adaptateur peut comprendre un large éventail de fonctionnalités telles que le flux de commande, le flot de données, la navigation séquentielle, le branchement conditionnel, dont la décision et l'itération, la saisie de données, le stockage du contexte des données, la transformation de données élémentaires, le contrôle transactionnel, les opérations logiques et le code personnalisé.

Les adaptateurs peuvent être réutilisés.

Il existe deux types principaux d'adaptateur :

- Les adaptateurs source destinés aux applications qui envoient des données.
- Les adaptateurs cible destinés aux applications qui reçoivent des données.

L'envoi d'un type de message d'une application vers une seconde application exige généralement l'utilisation d'un adaptateur source et d'un adaptateur cible. Si la seconde application doit envoyer un type de message vers la première application, un autre adaptateur source et un autre adaptateur cible doivent être utilisés. Ainsi, quatre adaptateurs sont généralement déployés pour envoyer un type de message de la première application vers la seconde application, puis pour envoyer un autre type de message de la seconde application vers la première application.

Un adaptateur distinct est exigé pour chaque type de message.

Un troisième type d'adaptateur, l'adaptateur de bean de session de service Java, est utilisé lorsqu'il est fait appel à des beans IBM WebSphere Application Server et d'entreprise côté cible du noyau. L'implémentation WebSphere Application Server de la spécification EJB (Enterprise JavaBeans) de Sun Microsystems permet l'utilisation d'adaptateurs de bean de session de service Java et d'autres beans d'entreprise. Pour plus d'informations, reportez-vous à la section «Utilisation de MQSeries Adapter Kernel avec WebSphere Business Integrator et WebSphere Application Server» à la page 29 et à la documentation MQSeries Adapter Builder.

MQSeries Adapter Builder
Interface graphique utilisateur (GUI) qui vous permet de créer un adaptateur pour pratiquement n'importe quelle application. L'interface utilisateur est similaire à l'interface utilisateur de MQSeries Integrator. Pour plus d'informations, consultez le centre d'aide et d'information MQSeries Adapter Builder.

MQSeries Adapter Kernel
Ensemble d'interfaces de programmation d'application (API), de plusieurs programmes exécutables en C et Java et plusieurs fichiers de configuration. Le noyau permet le déploiement et l'exécution des adaptateurs. Outre la prise en charge directe des adaptateurs, le noyau exécute des fonctions associées, dont le simple acheminement de messages et les services d'infrastructure tels que la création de

messages, le contrôle transactionnel, la fonction de trace et l'interfaçage avec MQSeries ou d'autres logiciels de communication en mode message, bus applicatif.

Le noyau est installé sur chaque ordinateur sur lequel est exécuté un adaptateur source ou un adaptateur cible.

Avec MQSeries Adapter Offering, les opérations et chaque application peuvent rester isolées par rapport aux caractéristiques du middleware, aux détails des messages et aux autres applications. Une interface commune pour l'application de messagerie permet l'ajout de nouvelles applications sans changer les applications ou opérations existantes.

MQSeries Adapter Kernel peut être déployé sur deux plates-formes. L'une des plates-formes correspond au côté source d'exécution et l'autre correspond au côté cible d'exécution. Le déploiement sur deux plates-formes permet une exploitation rentable et de faibles surcharges administratives. La présence d'une troisième plate-forme résidente destinée à l'acheminement et à la communication n'est pas nécessaire entre les deux côtés de l'exécution. Cependant, MQSeries Integrator peut être ajouté, en option, pour le courtage tel que l'acheminement complexe, la transformation de données et la médiation de données.

Sauf indication contraire, le reste de ce document s'applique uniquement à MQSeries Adapter Kernel. Pour des informations détaillées sur MQSeries Adapter Builder, consultez le Centre d'aide et d'information concernant ce produit.

Information produit sur le noyau

De manière simplifiée, la phase d'exécution—correspondant à la création du noyau et des adaptateurs—a pour but de :

1. Transférer des données d'une application source à une application cible.
2. Convertir les données de l'application source en message, généralement dans un format d'application neutre, qui est acheminé par le noyau à l'aide de MQSeries ou d'un autre logiciel de communication en mode message, bus applicatif.
3. Acheminer le message vers l'application cible.
4. Déterminer comment envoyer les données vers l'application cible.
5. Convertir les données du format du message qui est acheminé par le noyau,via un adaptateur, au format de l'application cible.

Dans cette section, la fonctionnalité du noyau est traitée à un haut niveau. La fonctionnalité est traitée de manière plus détaillée dans la section «Déroulement de l'exécution» à la page 15.

Le noyau comprend deux côtés :

- Le *côté source*, dont l'exécution est lancée lorsque le message est reçu de l'application source et s'achève lorsque le message est mis en file d'attente des messages.
- Le *côté cible*, dont l'exécution est lancée lorsque le message est récupéré dans la file d'attente des messages et s'achève lorsque le message est envoyé à la cible.

Chaque côté réside généralement sur un ordinateur différent, mais ils peuvent tous deux résider sur un même ordinateur.

Reportez-vous à la sectionfigure 1 à la page 6. Elle décrit la séquence suivante.

Côté source du noyau

1. Du côté source du noyau, l'application source envoie les données dans son *format d'application source*, en utilisant une *interface propre à l'application,* vers un adaptateur source qui a été créé à l'aide de MQSeries Adapter Builder. Un adaptateur source différent est exigé pour chaque type de message, par exemple, pour «ajouter un bordereau de vente» ou pour «synchroniser un enregistrement client. »

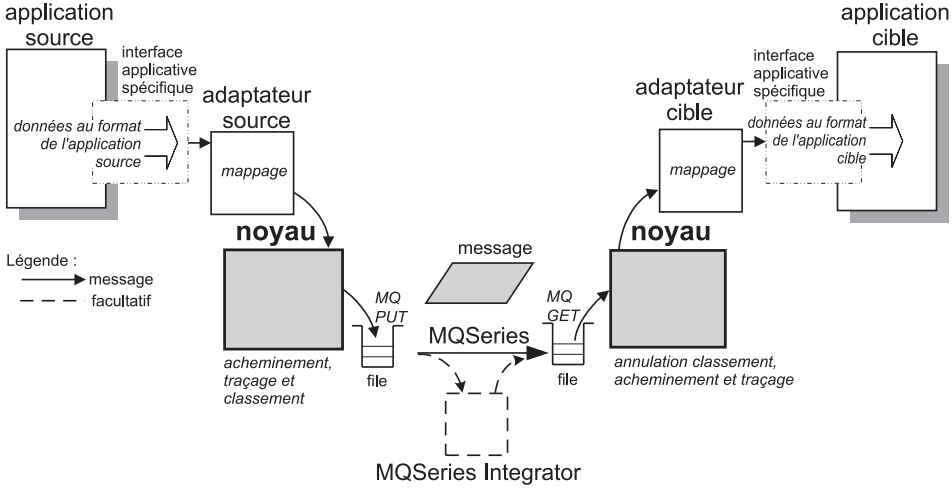
   L'interface propre à l'application doit être développée en dehors de MQSeries Adapter Offering. La nature exacte de l'interface propre à l'application dépend des caractéristiques de l'application source ou de l'application cible. Parmi les exemples figurent les appels API et les routines utilisateur, les lectures/écritures de fichiers, les déclencheurs de base de données et les files d'attente de messages.

   Notez que l'adaptateur source est exécuté dans le processus de l'application source. Tout démon ou serveur contenant l'adaptateur source doit être en cours d'exécution pour que cet adaptateur fonctionne.
2. L'adaptateur source remplit sa fonction suivant la manière dont il a été créé. L'une des fonctions types est la transformation des données élémentaires, c'est-à-dire le mappage des éléments du format d'application source au *format d'application de messagerie d'intégration* pour les données de corps. Les données de corps et les métadonnées supplémentaires représentant les valeurs de contrôle sont classées dans un *objet collecteur de messages du noyau*.
3. Lorsque l'adaptateur source transmet l'objet collecteur de messages au noyau à l'aide de l'*adaptateur natif*, les valeurs de contrôle de l'objet collecteur de messages (*valeurs de contrôle de message*) sont utilisées par le noyau pour contrôler le classement de l'objet collecteur de messages dans un format de messages de communication et l'acheminement de ces messages de communication.

Si le message ne contient pas certaines valeurs de contrôle de message, le noyau peut utiliser des valeurs par défaut ou des valeurs de contrôle de message obtenues à partir du fichier de configuration. Pour les définitions des valeurs de contrôle de message, reportez-vous à la section «Valeurs de contrôle de message» à la page 16.

4. Le noyau exécute ses fonctions, dont *classement des messages*, *acheminement* simple et, en option, *fonction de trace*. Reportez-vous à la section «Message et format de message» à la page 12, «Acheminement et communication» à la page 14 et à la section «Fonction de trace» à la page 29.

Figure 1. Généralités sur MQSeries Adapter Offering.



Communication du côté source au côté cible du noyau

5. A l'aide de son adaptateur natif, le noyau met le message dans la file d'attente de messages appropriée.

   Il existe deux méthodes d'envoi utilisées du côté source :

   - `sendMsg`, qui envoie le message et revient immédiatement. La méthode `sendMsg` peut également être utilisée avec les méthodes de `lancement`, `validatioinet annulation` pour envoyer des messages *de façon transactionnelle*; c'est-à-dire que des messages peuvent être envoyés si (et seulement si) les autres opérations ont abouti. Reportez-vous à la section «Capacités transactionnelles» à la page 28 pour plus d'informations.
   - `sendRequestResponse`, qui envoie le message et attend une réponse. La méthode `sendRequestResponse` ne peut être envoyée de façon transactionnelle.

     Notez qu'une troisième méthode `sendResponse` est utilisée du côté cible du noyau lorsque l'émetteur demande une réponse.

MQSeries ou un autre logiciel de communication en mode message, bus applicatif, transporte le message. Reportez-vous à la section «Rôle de MQSeries ou d'un autre logiciel de communication en mode message, bus applicatif» à la page 8. Notez que le logiciel de communication en mode message doit déjà être configuré pour assurer le support de MQSeries Adapter Offering.

En option, si MQSeries Integrator a été configuré dans le noyau comme étant la destination, MQSeries Integrator peut exécuter des fonctions de courtage. Reportez-vous à la section «Rôle de MQSeries Integrator» à la page 9. Si la destination finale, une file d'attente de messages, a été configurée selon les règles ou les flux de messages de MQSeries Integrator, MQSeries Integrator envoie le message dans la file d'attente des messages.

Le message arrive dans la file d'attente des messages appropriée.

Côté cible du noyau

6. Du côté cible du noyau, il existe deux *modèles de communication* potentiels pour l'interface entre l'exécution et l'application cible.
   - Le modèle le plus courant est le modèle *insertion*, sur lequel le noyau est responsable du lancement et de la gestion de la communication du message à l'application cible. Généralement, le modèle insertion n'exige pas de changement d'application cible pour assurer le support de MQSeries Adapter Offering.
   - Avec le modèle *extraction*, l'application cible est responsable de la gestion de la réception du message. Le modèle extraction nécessite un changement d'application cible pour assurer le support de MQSeries Adapter Offering. L'application cible doit gérer l'interface du noyau avec l'application cible.

   Dans le modèle insertion, notez que du côté cible, les processus du noyau doivent être démarrés par l'utilisateur avant d'obtenir et de communiquer le message. Pour plus d'informations, reportez-vous à la section «Démarrage du noyau» à la page 97.

   Avec le modèle insertion, le noyau obtient le message à partir de la file d'attente des messages. Il exécute l'acheminement si celui-ci est activé. Il poursuit l'acheminement du message en sélectionnant l'adaptateur cible approprié. En général,un adaptateur cible distinct est exigé pour chaque type de message.
7. Le noyau communique le message à l'adaptateur cible approprié. L'adaptateur cible exécute les fonctions qui lui ont été intégrées. L'une des fonctions types est le mappage d'éléments au format des données de l'application de messagerie d'intégration en éléments au *format de l'application cible*.

Les adaptateurs cible peuvent être hébergés soit par un démon d'adaptateur MQSeries Adapter Kernel, soit par WebSphere Application Server. Reportez-vous à la section «Utilisation de MQSeries Adapter Kernel avec WebSphere Business Integrator et WebSphere Application Server» à la page 29 pour obtenir une description de ce dernier programme.

8. L'adaptateur cible envoie les données à l'application cible au format de l'application cible en utilisant une interface propre à l'application développée en dehors de MQSeries Adapter Offering.
9. Lorsque l'adaptateur cible a communiqué son message, celui-ci est validé à partir de la file d'attente des messages. Ceci supprime le message de la file d'attente.
10. Si l'adaptateur source a défini une valeur de contrôle de message pour demander un accusé de réception, le noyau fournit soit un accusé de réception de communication des messages, soit une sortie de l'adaptateur cible à l'adaptateur source en utilisant la méthode `sendResponse`.
11. En cas d'erreur, le noyau met le message original dans la file d'attente des erreurs. Si le noyau ne peut mettre le message d'origine dans la file d'attente des erreurs, aucune validation ne se produit.

Rôle de MQSeries ou d'un autre logiciel de communication en mode message, bus applicatif

Les messages de communication de MQSeries Adapter Offering sont transportés sur des files d'attente de messages. Les files d'attente de messages sont fournies par des logiciels de communication en mode message tels que MQSeries ou le Java Message Service (JMS). Les messages transportés par MQSeries Adapter Offering utilisent les types de files d'attente suivants :

- *Files d'attente de réception*, selon la terminologie de MQSeries Adapter Offering. Celles-ci sont utilisées comme les files d'attente d'entrée principales pour la réception des messages. Il peut y avoir des files d'attente de réception multiples par application cible.
- *Files d'attente des erreurs*, selon la terminologie de MQSeries Adapter Offering. Celles-ci sont utilisées lorsqu'un message obtenu à partir d'une file d'attente de réception ne peut être traité.
- En option, *files d'attente de réponse*. Celles-ci sont utilisées avec la méthode `sendRequestResponse`.

MQSeries Adapter Offering utilise certaines capacités de MQSeries telles que les types de message suivants :

- Datagrammes, utilisés par la méthode`sendMsg`.
- Demande, utilisée par la méthode`sendRequestResponse`.
- Réponse, utilisée par la méthode `sendRequestResponse` et la méthode `sendResponse`.

MQSeries peut, en option, servir d'interface propre à l'application.

Reportez-vous à la section «Annexe B. Configurations validées» à la page 113 pour une liste des configurations validées de MQSeries et MQSeries Adapter Offering. Reportez-vous à la section «Logiciel» à la page 34 pour une liste des versions acceptées par MQSeries et par d'autres logiciels.

Rôle de MQSeries Integrator

MQSeries Integrator peut être déployé avec MQSeries Adapter Kernel en option. Il peut être utilisé pour répondre à plusieurs exigences éventuelles relatives au courtage :

- L'acheminement complexe, à savoir l'acheminement basé sur le contenu de l'en-tête du message ou du corps de message. L'acheminement peut être modifié de manière dynamique lorsque le contenu du corps de message est modifié. Reportez-vous à la section «Acheminement et communication» à la page 14 pour plus d'informations sur l'acheminement complexe et l'acheminement simple.
- La transformation de données, à savoir la modification en un type de document différent.
- La médiation de données, à savoir la modification du contenu du corps de message. Par exemple, si l'application source fournit la valeur `each` dans un champ mais que l'application cible prévoit que la valeur du champ sera `ea` , la médiation de données remplace la valeur fournie par la valeur prévue.

Vous pouvez utiliser MQSeries Integrator pour exécuter la plupart des opérations d'acheminement sur votre site. Vous pouvez également utiliser la fonctionnalité d'acheminement de MQSeries Adapter Kernel dans une moindre mesure.

Reportez-vous à la section «Annexe B. Configurations validées» à la page 113 pour une liste des configurations validées de MQSeries Integrator et MQSeries Adapter Offering. Reportez-vous à la section «Logiciel» à la page 34 pour une liste des versions acceptées par MQSeries Integrator et par d'autres logiciels.

Fonctionnement du noyau

Les éléments suivants sont traités dans cette section :

- «Composants d'exécution du noyau» à la page 10
- «Message et format de message» à la page 12
- «Acheminement et communication» à la page 14
- «Déroulement de l'exécution» à la page 15

Composants d'exécution du noyau

Lorsque les adaptateurs que vous créez, le code personnalisé que vous développez et MQSeries Adapter Kernel sont exécutés simultanément, ils offrent la fonctionnalité de MQSeries Adapter Offering.

Les composants majeurs d'exécution du noyau sont les suivants :

adaptateur source
Logiciel créé pour une application spécifique (généralement à l'aide de MQSeries Adapter Builder) permettant de convertir les données de cette application en un format de messagerie d'intégration (données de corps). Les adaptateurs source sont généralement exécutés sur le même ordinateur que l'application source, soit comme un processus interne à l'application soit comme un processus distinct. Les fichiers, les structures C et les objets Java sont des exemples de données source. XML est un exemple de format de messagerie d'intégration, généralement conforme à une norme de l'industrie telle que OAG ou RosettaNet.

collecteur de messages
Boîtier destiné aux métadonnées utilisé par le noyau pour encapsuler le message d'intégration et les autres données de contrôle utilisées par le noyau. Parmi les exemples de métadonnées figurent les identificateurs d'application (identificateurs logiques) des applications source et cible, la catégorie du message (par exemple, OAG), le type de message (par exemple, "bon de commande") et le message de communication (données de corps) en cours d'envoi ou de réception.

adaptateur natif
Logiciel utilisé pour l'envoi et la réception d'objets collecteur de messages. Lors de l'envoi de messages, l'adaptateur natif permet l'acheminement simple de données et le support d'un ou de plusieurs mécanismes d'entraînement de communications. L'acheminement simple de données est basé sur les métadonnées de l'objet collecteur de messages telles que la catégorie et le type de message. Les messages peuvent être envoyés de manière asynchrone ou synchrone. Si le mécanisme d'entraînement de communication sous-jacent prend en charge la messagerie transactionnelle, les messages peuvent être envoyés sous commande transactionnelle monophase. Le support transactionnel est limité aux capacités du mécanisme d'entraînement utilisé. L'objet collecteur de messages est classé sous le format de message de communication utilisé par le mécanisme d'entraînement. Lorsqu'un message de communication est reçu, l'adaptateur natif annule ce classement et remet le message dans l'objet collecteur de messages.

démon de l'adaptateur
Processus qui instancie les agents de l'adaptateur. Après son démarrage, le démon de l'adaptateur reste actif. Dans chaque application cible, il peut y avoir un démon d'adaptateur pour chaque file d'attente de réception de l'application.

agent de l'adaptateur
Processus qui communique chaque message à l'adaptateur cible approprié. Chaque agent gère un adaptateur natif. Le démon de l'adaptateur crée et démarre les agents.

La présence d'agents multiples permet la *communication de messages en traitement multifilières* à des adaptateurs cible. Chaque agent, ainsi que son adaptateur natif, peut gérer une filière. S'il n'existe qu'un agent, la communication des messages à l'adaptateur cible et donc à l'application cible est à filière unique.

Outre la gestion d'un adaptateur natif, l'agent réalise également les tâches suivantes :

- Il instancie le client trace si la fonction de trace est activée.
- Il instancie la classe de connexion correspondant à chaque application cible.
- Il sélectionne l'adaptateur cible en se basant sur le type et la catégorie de corps du message.
- Il envoie le message à l'adaptateur cible sélectionné.
- S'il ne peut effectuer une validation, il effectue une annulation, place un drapeau sous le démon de cet adaptateur pour tous les autres agents, puis s'arrête ainsi que son adaptateur natif. Cela indique que le message comporte un incident. L'arrêt de tous les agents empêche les autres agents de procéder au retraitement du même message comportant l'incident avec le même résultat.
- Lorsqu'il reconnaît le drapeau placé par un autre agent en vue de l'arrêt, il s'arrête ainsi que son adaptateur natif.

adaptateur cible
Logiciel créé pour une application spécifique (généralement à l'aide de MQSeries Adapter Builder) permettant de convertir les données d'un format de messagerie d'application (données de corps) en types de données exigés par une application cible. L'adaptateur cible appelle les API nécessaires sur l'application cible pour communiquer le message. Les adaptateurs cible fonctionnent sur le même ordinateur que l'application ou le client de l'application.

adaptateur de bean de session de service Java
Type d'adaptateur EJB en langage Java qui est hébergé dans un serveur EJB tel que WebSphere Application Server.

composant de configuration

Données servant à convertir les identificateurs logiques en objets tels que des noms de file d'attente. Les données de configuration peuvent être spécifiées soit dans un fichier, soit dans la structure LDAP du produit WebSphere Business Integrator. Les données contrôlent les aspects suivants de la configuration du noyau :

- Classement et acheminement des messages
- Vérification de l'installation
- Mode de communication
- Fonction de trace

Reportez-vous à la section «Le fichier de configuration» à la page 69 pour une description complète du fichier de configuration. Reportez-vous à la documentation de WebSphere Business Integrator pour savoir comment configurer ce produit afin qu'il fonctionne avec le noyau.

composant de la fonction de trace

Logiciel qui écrit les messages trace. La plupart des composants du noyau utilisent le composant de la fonction de trace. Reportez-vous à la section «Fonction de trace» à la page 29 pour une vue d'ensemble de la fonction de trace et au manuel *Problem Determination Guide* pour obtenir des détails sur cette fonction.

Message et format de message

Dans MQSeries et MQSeries Adapter Offering, un *message* est une collecte de données envoyée par un programme et destinée à un autre programme. Le format du message à un moment donné dépend de l'emplacement du message dans le flot de messages à ce moment précis. MQSeries Adapter Kernel spécifie trois types de messages, comme suit :

- *Message d'intégration*—Message se composant de données provenant d'une application source converties en un autre format, tel que XML, pour envoi à une application cible. Le message d'intégration est inséré dans l'objet collecteur de messages en tant que données de corps du message. XML est une norme destinée à la représentation de données. Lorsque le format est XML, celui-ci est défini par une *Définition de type de document* (DTD). Une DTD correspond à un ou plusieurs fichiers contenant une définition formelle de document—dans ce cas, du corps de message. Bien que cela soit fortement recommandé, il n'est pas nécessaire que le corps de message soit dans un format d'application neutre. Le format du corps de message peut être privé ou autrement spécialisé, mais ce type de format n'est pas recommandé.

Les documents objet de gestion (BOD) peuvent être utilisés par MQSeries Adapter Offering pour définir les corps de messages dans ses messages d'intégration. Un BOD est la représentation d'une opération standard effectuée au sein d'une organisation ou entre des organisations, PAr exemple, l'«ajout d'un bon de commande», l'«indication de la disponibilité du produit» ou l'«ajout d'un bordereau de vente». Les BOD sont définis en XML par le groupe Open Applications Group (OAG). L'utilisation des BOD est recommandée mais n'est pas obligatoire.

- *Objet collecteur de messages*—Objet contenant le message d'intégration et les métadonnées d'en-tête supplémentaires représentant les valeurs de contrôle propres à MQSeries Adapter Kernel. L'adaptateur source crée l'objet collecteur de messages, définit les informations de contrôle appropriées et, en cas de message d'intégration à envoyer, définit les données de corps. Les adaptateurs cible reçoivent les objets collecteur de messages, obtiennent les données de corps et convertissent ces données en données propres à l'application cible. Les adaptateurs source et les adaptateurs cible sont créés à l'aide de MQSeries Adapter Builder.
- *Message de communication*—Toute information propre au transfert de communications plus l'objet collecteur de messages, convertie en un format de messagerie propre au transfert de communications utilisé. Certains transferts de communications supportent plusieurs formats de messagerie. Généralement, les valeurs des métadonnées d'en-tête du noyau associées au message de communication sont considérées comme des données d'application par le transfert de communications. Pour plus d'informations, reportez-vous à la section «Annexe A. Modes de communication» à la page 107. A titre d'exemple,le transfert MQSeries se compose d'un en-tête de message propre à MQSeries et de l'objet collecteur de messages classé. Les formats MQSeries spécifiques comprennent les suivants :
  - L'en-tête de message MQSeries qui est ajouté par MQSeries
  - Si MQSeries Integrator est utilisé, l'en-tête de message propre à la version :
    - L'en-tête de message MQSeries Integrator version 1 si MQSeries Integrator version 1.1 est utilisé
    - L'en-tête de message MQSeries Integrator version 2 si MQSeries Integrator version 2 est utilisé
  - Les métadonnées de l'en-tête propre au noyau représentant les valeurs de contrôle
  - Le message d'intégration (données de corps)

  Reportez-vous à la section «Annexe C. En-têtes de messages» à la page 115 pour une liste des champs appropriés utilisés dans les en-têtes de message de MQSeries Adapter Offering et leurs descriptions.

Acheminement et communication

Le noyau achemine chaque message à partir de l'adaptateur source et le communique à l'adaptateur cible approprié. L'acheminement s'effectue en deux étapes :

1. Le côté source du noyau met le message dans la file d'attente des messages appropriée.
2. Le côté cible du noyau extrait le message de la file d'attente des messages et appelle l'adaptateur cible approprié.

L'acheminement est déterminé par plusieurs facteurs :

- Les files d'attente de messages. Au niveau le plus fondamental, les files d'attente de messages doivent être configurées pour supporter l'acheminement de MQSeries Adapter Offering.
- Les valeurs de contrôle dans le message. Elles comprennent l'identificateur logique source, l'identificateur logique cible, l'identificateur logique de réponse, la catégorie de corps, le type de corps, l'identificateur de transaction, l'identificateur de message, l'accusé de réception demandé et les données d'horodatage. Pour plus d'informations, reportez-vous à la section «Valeurs de contrôle de message» à la page 16. L'identificateur logique cible du message peut se substituer au fichier de configuration du noyau. L'acheminement peut être modifié de façon dynamique au fur et à mesure que ces valeurs de contrôle de message se trouvant dans chaque en-tête de message sont modifiées. Cependant, le contenu des données de corps du message (message d'intégration) ne peut déterminer l'acheminement.
- Les valeurs de contrôle de message du fichier de configuration du noyau. Ce fichier peut indiquer des identificateurs logiques cible, des noms de file d'attente et des adaptateurs cible associés. Déterminez et modifiez la configuration en éditant ce fichier. Reportez-vous à la section «Le fichier de configuration» à la page 69 pour de plus amples informations.
- Eventuellement, MQSeries Integrator, qui peut être utilisé pour le courtage de messages, y compris l'acheminement complexe. L'acheminement peut être modifié de manière dynamique lorsque le contenu du corps de message est modifié. Reportez-vous à la section «Rôle de MQSeries Integrator» à la page 9. Au contraire, par lui-même, MQSeries Adapter Offering ne peut exécuter que l'acheminement simple. L'acheminement simple est basé sur une combinaison de valeurs de contrôle de message dans le message et de valeurs de contrôle de message associées dans le fichier de configuration. Il n'est pas basé sur le contenu du corps de message.

Une demande d'accusé de réception de la communication des messages peut être adressée au noyau. Il s'agit d'un accusé de réception au niveau de l'application.

Déroulement de l'exécution

Cette section décrit de manière détaillée le déroulement de l'exécution —la manière dont le noyau envoie, achemine, trace et communique un message dans un environnement de production type. Reportez-vous à la section figure 2 pour voir le diagramme du déroulement de l'exécution.

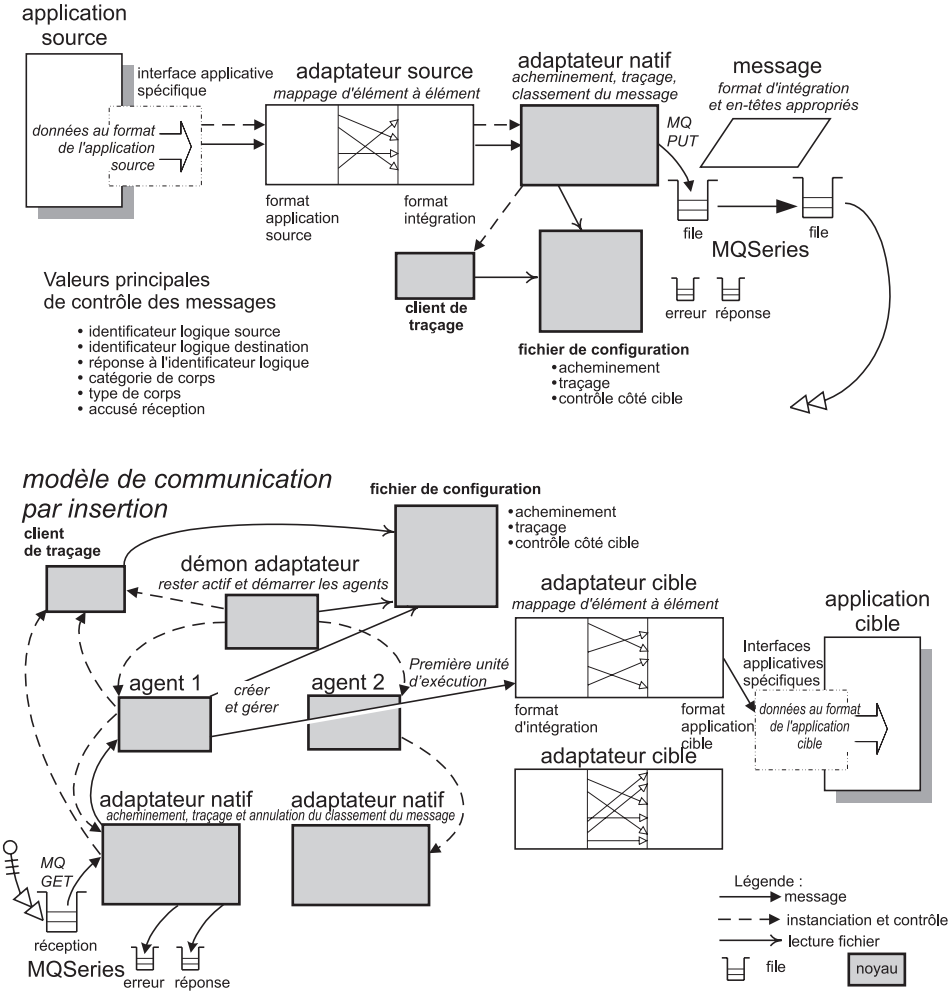


Figure 2. Classer, envoyer, acheminer et tracer un message — vue d'ensemble.

Côté source du noyau

Cette section décrit le déroulement de l'exécution du côté source du noyau, c'est-à-dire le déplacement de données d'une application source à un transfert de communications par l'intermédiaire d'un adaptateur source. La section «Côté cible du noyau» à la page 20 décrit le déplacement des données du transfert de communications à la cible.

1. A l'aide d'une interface propre à l'application, l'adaptateur source acquiert un message de l'application source. Généralement, l'adaptateur source est appelé par l'interface propre à l'application.
2. L'adaptateur source exécute les fonctions qui lui ont été intégrées dans MQSeries Adapter Builder. Généralement, il transforme les données au format d'application source en un format d'intégration d'application neutre (pour le corps de message).

   Dans le cadre de sa fonctionnalité, l'adaptateur source place plusieurs valeurs de contrôle de message dans l'en-tête de MQSeries Adapter Kernel ; il utilise ces valeurs pour envelopper le message. Les cinq premières valeurs de contrôle de message déterminent le classement et l'acheminement, et la dernière valeur détermine l'accusé de réception.

   valeurs de contrôle de message

   identificateur logique source
   Identificateur logique de l'application source. Il est toujours exigé dans le message.

   identificateur logique cible
   Identificateur logique de l'application cible. S'il n'est pas présent dans le message, des valeurs par défaut du fichier de configuration sont utilisées à la place. Dans le fichier de configuration, des identificateurs logiques cible multiples peuvent être utilisés au lieu des valeurs absentes du message.

   Identificateur logique de réponse
   L'identificateur logique de l'application à laquelle des réponses doivent être envoyées dans le cas où une réponse est exigée. Il prend par défaut l'identificateur logique source du message.

   catégorie de corps
   Représente le type d'application du message—par exemple, OAG ou RosettaNet. Il est toujours exigé dans le message.

   type de corps
   Représente le but spécifique du message—par exemple, «l'ajout d'un bordereau de vente» ou «la synchronisation des informations articles». Il est toujours exigé dans le message.

accusé de réception demandé

Détermine si l'application source demande une réponse. La réponse peut prendre l'une des deux formes suivantes :

- Données de réponse provenant de l'application source
- Un message Confirm BOD OAG

Remarque : Le message Confirm BOD est prédéfini par l'OAG. Sa catégorie de corps est `OAG` et son type de corps est `CONFIRM_BOD_003`. Il peut également contenir des données.

Cette réponse est un accusé de réception au niveau de l'application.

Lorsque le noyau utilise la méthode`sendRequestResponse` pour envoyer le message, seule la première réponse reçue par la méthode `sendRequestResponse` est utilisée. Si le message d'origine est envoyé à des destinations multiples et demande une réponse (ce qui n'est pas recommandé), seule la première réponse est renvoyée à l'application source.

La valeur par défaut est sans accusé de réception ; ainsi, aucune réponse n'est demandée ou envoyée.

3. L'adaptateur source initialise l'adaptateur natif et lui transmet les informations suivantes :
   - L'identificateur logique de l'application sous laquelle l'adaptateur source est exécuté.
   - L'objet collecteur de messages, qui contient les valeurs de contrôle et les données de corps du message.
4. L'adaptateur natif vérifie dans le fichier de configuration si le programme trace est activé pour cet identificateur logique source. Si le programme trace est activé, l'adaptateur natif instancie un client trace.
5. Le client trace vérifie dans le fichier de configuration quel niveau de trace utiliser et afin d'obtenir d'autres valeurs. Le client trace utilise le niveau de trace pour opérer un filtrage des messages trace. Reportez-vous à la section «Fonction de trace» à la page 29 pour une vue d'ensemble de la fonction de trace et au manuel *Problem Determination Guide* pour obtenir des informations détaillées sur cette fonction.
6. L'adaptateur natif vérifie si l'identificateur logique cible est dans l'objet collecteur de messages. S'il est présent, il est utilisé.
   - Si l'identificateur logique cible n'est pas présent, l'adaptateur natif effectue une recherche pour savoir si l'identificateur logique cible par défaut est dans le fichier de configuration en se basant sur l'identificateur logique source, la catégorie et le type de corps.

- En se basant sur l'identificateur logique source, l'adaptateur natif recherche, en plusieurs étapes, les valeurs de catégorie et de type de corps dans le fichier de configuration, dans l'ordre suivant :
  a. Valeurs de la catégorie et du type de corps spécifiques.
  b. Valeur de catégorie spécifique et valeur de type par défaut.
  c. Valeur de catégorie par défaut et valeur de type spécifique.
  d. Valeurs de la catégorie et du type de corps par défaut.

  Remarque : Le noyau effectue cette recherche en plusieurs étapes à chaque fois qu'il recherche des valeurs dans le fichier de configuration.

7. Pour chaque identificateur logique cible déterminé au cours de l'étape précédente, l'adaptateur natif recherche le *mode de communication* en se basant sur l'identificateur logique cible, la catégorie et le type de corps. Les modes de communication suivants sont pris en charge :

   MQPP Le noyau transporte les messages à l'aide des services de base MQSeries.

   MQRFH1 Le noyau transporte les messages à l'aide de MQSeries et les messages des courtiers via MQSeries Integrator version 1.1.

   MQRFH2 Le noyau transporte les messages à l'aide de MQSeries et les messages des courtiers via MQSeries Integrator version 2.

   MQBD Le noyau transporte les messages à l'aide des services de base de MQSeries mais envoie et reçoit uniquement les données de corps.

   MQ Le noyau transporte les messages à l'aide de MQSeries.

   JMS Le noyau transporte les messages à l'aide de Java Message Service (JMS).

   FILE Le noyau place des messages dans un fichier et les extrait d'un fichier. Ce mode est prévu à des fins de diagnostic uniquement.

   La structure du message est différente dans chaque mode de communication. Reportez-vous à la section «Message et format de message» à la page 12. Pour plus d'informations sur les modes de communication, reportez-vous à la section «Annexe A. Modes de communication» à la page 107.

Remarque : Si MQSeries Integrator est utilisé, la destination finale à laquelle il envoie le message doit utiliser le même mode de communication que MQSeries Integrator pour recevoir les messages.

8. En fonction du mode de communication, l'adaptateur natif instancie une sous-classe qui lui est intégrée pour gérer le message. La sous-classe est appelée *service de messages logiques*. Chaque mode de communication a une sous-classe de service de messages logiques distincte.

   L'adaptateur natif transmet les identificateurs logiques cible, la catégorie et le type de corps au service de messages logiques.

9. La sous-classe de service de messages logiques trouve les paramètres dont elle a besoin pour envoyer le message. Par exemple, si le mode de communication est MQPP, les paramètres comprennent le format et les noms des files d'attente d'erreurs, de réception et de réponse. En fonction des identificateurs logiques cible, de la catégorie et du type de corps qui lui sont transmis, le service de messages logiques effectue une recherche en plusieurs étapes dans le fichier de configuration :
   a. Valeurs de la catégorie et du type de corps spécifiques.
   b. Valeur de catégorie spécifique et valeur de type par défaut.
   c. Valeur de catégorie par défaut et valeur de type spécifique.
   d. Valeurs de la catégorie et du type de corps par défaut.

A ce stade, le service de messages logiques possède toutes les informations dont il a besoin pour acheminer et classer le message.

10. Le service de messages logiques exécute les tâches suivantes :
    - Classe le message comme il convient suivant le format et le mode de communication. Chaque mode de communication utilise un format par défaut si le format n'est pas autrement spécifié. Par exemple, si le mode de communication est MQRFH2, le service de messages logiques crée des en-têtes appropriées et structure le message en vue du transfert à l'aide de MQSeries et en vue du courtage via MQSeries Integrator version 2.
    - Envoie le message. Par exemple, si le mode de communication est MQRFH2, il met le message dans la file d'attente des messages MQSeries appropriée.

11. Il existe deux méthodes pour envoyer le message :
    - Si l'adaptateur natif utilise la méthode `sendMsg` pour envoyer le message, l'adaptateur natif n'attend pas la réponse.
    - Si l'adaptateur natif utilise la méthode `sendRequestResponse` pour envoyer le message, le service de messages logiques attend la réponse. En utilisant le service de messages logiques, l'adaptateur natif intercepte la file d'attente de réponses pour obtenir le *délai de time-out de réception* défini dans le fichier de configuration.

Le délai de time-out de réception est basé sur l'identificateur d'application source, la catégorie et le type de corps.

- Si un accusé de réception est reçu, l'adaptateur natif renvoie le message.
- Si un accusé de réception n'est pas reçu pendant le délai de time-out de réception, l'adaptateur natif ne renvoie pas de message.

12. MQSeries ou un autre logiciel de communication en mode message, bus applicatif, transporte le message en fonction de sa configuration. MQSeries Integrator peut exécuter des services de courtage, en option. Reportez-vous à la section«Rôle de MQSeries Integrator» à la page 9.
13. Lorsque l'adaptateur source n'a plus besoin de l'adaptateur natif, il le ferme pour libérer les ressources.

Côté cible du noyau

Cette section indique comment utiliser MQSeries Adapter Kernel de façon autonome pour recevoir et traiter des messages côté cible, et fournit une description détaillée de l'utilisation du noyau avec WebSphere Application Server. Reportez-vous à la section «Utilisation de MQSeries Adapter Kernel avec WebSphere Business Integrator et WebSphere Application Server» à la page 29 pour savoir comment utiliser le noyau avec JMS, le composant JMS Listener de WebSphere Business Integrator et WebSphere Application Server du côté cible du noyau. Cette section décrit le modèle de communication par insertion, dans lequel le noyau est responsable du lancement et de la gestion de la communication du message à l'application cible. Reportez-vous à la section «Modèles de communication» à la page 138 pour une brève description des modèles.

Généralités sur l'agent de l'adaptateur

Cette section décrit la structure et le comportement des agents d'adaptateurs MQSeries Adapter Kernel. L'un des postulats de l'architecture MQSeries Adapter Kernel est que les applications cible ne participent pas activement aux flots de données d'intégration avec d'autres applications ; autrement dit, en principe, ces applications ne recherchent pas activement des messages à traiter. Dans ce cas, les données de message doivent être activement insérées (push) dans l'application cible. Les agents d'adaptateur insèrent les données de message dans une application ou un autre service en sélectionnant et en appelant une série de types d'interface de service.

MQSeries Adapter Kernel peut héberger des agents d'adaptateur qui fonctionnent dans un démon autonome (démon de l'adaptateur) ou dans un serveur d'application Enterprise JavaBeans (actuellement, IBM WebSphere Application Server Advanced Edition). Les messages parviennent à l'agent d'adaptateur par différents moyens, selon le type d'environnement cible utilisé. Si un démon d'adaptateur autonome est utilisé, il héberge un ou plusieurs agents d'adaptateur autonomes qui reçoivent des messages par

l'intermédiaire de l'adaptateur natif. Si un serveur EJB est utilisé, le composant JMS Listener reçoit les messages et les transmet à un bean de message agent.

Quel que soit l'environnement cible utilisé, une fois que l'agent de l'adaptateur a reçu le message, il le transmet à l'adaptateur cible approprié. L'adaptateur cible effectue alors les opérations nécessaires pour communiquer le message à l'application cible. Les adaptateurs cible sont créés pour fonctionner avec des applications cible spécifiques. En revanche, le démon de l'adaptateur, le serveur d'application, l'agent d'adaptateur autonome et le bean de message agent ne sont pas spécifiques à une application source ou cible donnée.

L'agent de l'adaptateur gère deux types d'interface d'adaptateur cible : les adaptateurs de commande EAB (Enterprise Access Builder) et les beans de session de service EJB. Chaque type d'adaptateur comprend un gestionnaire qui configure l'environnement approprié, accède aux informations de configuration complémentaires requises pour l'adaptateur et effectue les autres tâches de bas niveau nécessaires pour le fonctionnement de l'adaptateur. Le gestionnaire utilisé dépend du type d'adaptateur répertorié dans le fichier de configuration. Les deux types existants assurent les tâches supplémentaires suivantes :

- Le gestionnaire EAB obtient une classe de connexion, qui fournit des informations de connexion à l'adaptateur cible, et initialise la phase d'exécution de la structure IBM CCF (Common Connector Framework). L'identificateur logique de l'application est transmis à la classe de connexion, qui l'utilise pour obtenir les informations de connexion propres à l'application.
- Le gestionnaire EJB obtient une connexion JNDI (Java Naming and Directory Interface), puis l'interface éloignée du bean de session de service et les autres informations requises pour l'accès au bean de session de service.

Le déroulement élémentaire des processus d'un agent d'adaptateur dans un démon d'adaptateur autonome est le suivant :

1. Au démarrage le démon de l'adaptateur instancie un ou plusieurs agents d'adaptateur autonomes, selon les informations fournies dans le fichier de configuration du noyau. L'identificateur logique et les valeurs facultatives de la catégorie et du type de corps de l'application sont transmis au démon. Les valeurs de la catégorie et du type de corps par défaut sont utilisées pour obtenir des valeurs de configuration supplémentaires.

2. Chaque agent d'adaptateur autonome exécute les tâches suivantes :
   a. L'agent de l'adaptateur instancie un adaptateur natif et commence à recevoir des messages. Chaque message est reçu sous contrôle transactionnel et renvoyé à l'agent de l'adaptateur sous forme d'objet collecteur de messages.
   b. Pour chaque message reçu, l'agent de l'adaptateur extrait le type de commande de l'adaptateur cible afin de traiter le message à partir du fichier de configuration et obtient le gestionnaire approprié pour ce type de commande.
   c. Le gestionnaire tire du fichier de configuration les informations complémentaires dont il a besoin pour instancier l'instance de l'adaptateur cible. Il instancie l'adaptateur cible et lui transmet le message.
   d. Si le traitement aboutit (c'est-à-dire qu'il ne provoque aucune exception, erreur ou donnée incorrecte), le message est validé à partir de la file d'attente des messages entrants. Si le traitement n'aboutit pas, le message est placé dans une file d'attente des erreurs. Si le traitement n'aboutit pas, mais que le message ne peut pas être placé dans une file d'attente des erreurs, le message est annulé et tous les agents arrêtés.

Le déroulement élémentaire des processus d'un agent d'adaptateur dans WebSphere Application Server est le suivant :

1. Un processus JMS Listener fonctionnant avec le serveur EJB de WebSphere Application Server Advanced Edition reçoit un message JMS. Il obtient alors un bean de message agent permettant de traiter le message. L'identificateur logique et les valeurs facultatives de la catégorie et du type de corps de l'application font partie de l'environnement du bean de message agent. Les valeurs de la catégorie et du type de corps par défaut sont utilisées pour obtenir des valeurs de configuration supplémentaires.
2. Chaque bean de message agent exécute les tâches suivantes :
   a. Il instancie un adaptateur natif et lui applique la méthode `receiveMsg`, en lui transmettant le message JMS. L'adaptateur natif convertit le message JMS en objet message et renvoie l'objet collecteur de messages.
   b. Pour chaque objet collecteur de messages reçu, l'agent de l'adaptateur extrait le type de commande de l'adaptateur cible afin de traiter l'objet à partir du fichier de configuration et obtient le gestionnaire approprié pour ce type de commande.
   c. Le gestionnaire tire du fichier de configuration les informations complémentaires dont il a besoin pour instancier l'instance de l'adaptateur cible. Il instancie l'adaptateur cible et lui transmet l'objet collecteur de messages.

d. Si le traitement aboutit (c'est-à-dire qu'il ne provoque aucune exception, erreur ou donnée incorrecte), l'objet collecteur de messages est validé à partir de la file d'attente des messages entrants. Si le traitement n'aboutit pas, l'objet collecteur de messages est placé dans une file d'attente des erreurs. Si le traitement n'aboutit pas, mais que le message ne peut pas être placé dans une file d'attente des erreurs, le message est annulé et tous les agents arrêtés.

Le déroulement de l'exécution côté cible avec un démon d'adaptateur et un agent d'adaptateur autonome est le suivant :

1. Il existe un démon d'adaptateur pour chaque file d'attente de réception de l'application cible. Le démon de l'adaptateur est démarré.

   A son démarrage, un nom servant d'identificateur d'application lui est attribué. Généralement, le nom de chaque démon de l'adaptateur est basé sur l'identificateur logique cible, c'est-à-dire l'identificateur logique de l'application cible. Par exemple, si le démon de l'adaptateur assure le service d'une application cible dont l'identificateur logique cible est `ABC`, le nom du démon de l'adaptateur est `ABCdaemon`.

   Parmi les autres paramètres pouvant être transmis au démon de l'adaptateur au démarrage figurent la catégorie et le type de corps. L'adaptateur natif les utilise ultérieurement pour déterminer le mode de communication et la file d'attente de réception pour les messages entrants.

   Reportez-vous à la section «Démarrage du noyau» à la page 97 pour obtenir des instructions concernant le démarrage du démon de l'adaptateur.
2. Au démarrage, le démon de l'adaptateur effectue une recherche dans le fichier de configuration pour déterminer si le programme trace est activé ou non pour ce nom spécifique du démon de l'adaptateur. Si le programme trace est activé, le démon de l'adaptateur instancie un client trace.

   Reportez-vous à la section *Problem Determination Guide* pour obtenir des détails sur le programme trace.
3. Au démarrage, le démon de l'adaptateur instancie le premier agent et lui transmet le nom du démon de l'adaptateur ainsi que la catégorie et le type de corps du message.
4. Le premier agent effectue une recherche dans le fichier de configuration pour déterminer si le programme trace est activé pour ce nom de démon spécifique de l'adaptateur. Si le programme trace est activé, le premier agent instancie un client trace, et le client trace effectue une recherche dans le fichier de configuration pour déterminer le niveau de trace. Reportez-vous à la section *Problem Determination Guide* pour obtenir une liste des niveaux de trace valides.

5. Le premier agent effectue une recherche dans le fichier de configuration, en fonction de l'identificateur d'application du démon de l'adaptateur, pour trouver les valeurs indiquant le nombre minimum d'agents devant être instanciés et démarrés.

   Le premier agent effectue également une recherche de l'*identificateur d'application dépendante*. L'identificateur d'application dépendante est le nom de l'application dont l'agent assure le service. Il est transmis ultérieurement à l'adaptateur natif.
6. Le démon de l'adaptateur interroge le premier agent sur le nombre minimum d'agents.
7. Le démon de l'adaptateur démarre le premier agent, puis instancie et démarre le nombre minimum d'agents.

   La raison de la présence d'agents multiples est qu'elle permet la communication de messages en traitement multifilières à des adaptateurs cible. Chaque agent, ainsi que son adaptateur natif, peut gérer une filière. S'il n'existe qu'un agent, la communication des messages à l'adaptateur cible et donc à l'application cible est à filière unique.

   Sur les systèmes AIX, deux principes d'organisation sont disponibles pour les filières : organisation basée sur le processus et organisation basée sur le système. Dans l'organisation basée sur le processus (par défaut), toutes les filières utilisateur sont mappées dans un groupe de filières du noyau du système d'exploitation (SE) et sont exécutées sur un groupe de processeurs virtuels. Dans l'organisation basée sur le système, chaque filière utilisateur est mappée dans une filière unique du noyau du système d'exploitation et exécutée sur un processeur virtuel unique. Si vous utilisez des adaptateurs source C appelés à partir des fichiers exécutables sur AIX, vous devez utiliser l'organisation basée sur le système. Pour obtenir des informations sur le paramétrage du principe d'organisation par filières sur AIX, reportez-vous à la section 6 à la page 47.

   Notez que seule l'organisation basée sur le processus est acceptée par les systèmes Windows, HP-UX, Solaris et OS/400.

Les autres agents effectuent également les étapes suivantes réalisées par le premier agent :

8. Chaque agent instancie son adaptateur natif associé. Un adaptateur natif est associé à chaque agent. L'identificateur d'application dépendante, la catégorie et le type de corps sont transmis à l'adaptateur natif. L'adaptateur natif utilise ces trois valeurs pour déterminer le mode de communication et, à l'aide du service de messages logiques, le format et la file d'attente de réception pour les messages entrants. Ce processus est semblable au processus utilisé pour envoyer des messages.
9. L'adaptateur natif extrait le message de communication de la file d'attente de réception sous contrôle de validation et le convertit en objet collecteur de messages. Il supprime tous les en-têtes propres au transfert de communications, à l'exception de l'en-tête du noyau natif.

10. L'adaptateur natif transmet l'objet collecteur de messages à l'agent, qui lit la catégorie, le type de corps et la valeur de l'accusé de réception demandé dans l'en-tête du noyau natif du message.

    En fonction de l'identificateur d'application dépendante, de la catégorie et du type de corps, l'agent effectue une recherche en plusieurs étapes dans le fichier de configuration pour trouver le type de commande cible à appeler, dans l'ordre suivant :

    a. Valeurs de la catégorie et du type de corps spécifiques.
    b. Valeur de catégorie spécifique et valeur de type par défaut.
    c. Valeur de catégorie par défaut et valeur de type spécifique.
    d. Valeurs de la catégorie et du type de corps par défaut.

    En fonction du type de commande cible, l'agent détermine le gestionnaire de type d'adaptateur cible approprié, une classe Java qui traite ce type d'adaptateur particulier. Il instancie cet adaptateur cible particulier.

11. Il existe deux types de gestionnaires de type d'adaptateur : les gestionnaires d'adaptateur cible de commande EAB et les gestionnaires d'adaptateur cible de bean de session de service EJB. Ces différents types de gestionnaires fonctionnent comme suit :

    Remarque : Le gestionnaire d'adaptateur cible de bean de session de service EJB n'est pris en charge que lorsque WebSphere Business Integrator fonctionne avec WebSphere Application Server sur la plate-forme Windows NT.

    - Si un gestionnaire d'adaptateur cible de commande EAB est appelé, il lance l'environnement de structure CCF (Common Connector Framework), définit une classe de connexion avec un nom tiré du fichier de configuration et appelle l'adaptateur cible EAB avec le nom tiré du fichier de configuration.
    - Un gestionnaire d'adaptateur cible de bean de session de service EJB doit communiquer avec WebSphere Business Integrator et WebSphere Application Server pour obtenir les informations de configuration appropriées et appeler le bean de session de service EJB. Reportez-vous à la section «Utilisation de MQSeries Adapter Kernel avec WebSphere Business Integrator et WebSphere Application Server» à la page 29 pour savoir comment utiliser le noyau avec JMS, le composant JMS Listener de WebSphere Business Integrator et WebSphere Application Server du côté cible du noyau.

12. Chaque type d'adaptateur possède une interface et des classes de support nécessaires différentes, comme suit :
    - Une commande d'adaptateur cible EAB a trois méthodes à appeler, qui s'exécutent dans l'ordre suivant :
        a. La méthode *sélectionner l'entrée de message* qui sélectionne le message pour traitement dans l'adaptateur cible
        b. La méthode *exécuter* qui traite le message ayant été placé dans l'adaptateur cible via la méthode sélectionner l'entrée de message, puis attend.
            1) L'adaptateur cible exécute les fonctions qui lui ont été intégrées à l'aide de MQSeries Adapter Builder. Généralement, il transforme les données du message d'intégration au format d'application cible. Il effectue un mappage des éléments un par un.
            2) L'adaptateur cible envoie le message à l'application cible via l'interface propre à l'application.
            3) Suivant sa nature, l'application cible envoie ou non une réponse à l'adaptateur cible.
        c. La méthode *obtenir la sortie du message* qui permet d'obtenir la réponse de l'adaptateur cible. La réponse peut simplement indiquer que l'application cible a reçu le message ; elle peut également contenir des données.
    - Un bean de session de service EJB appelle une méthode, qui exige un objet `TerminalDataContainer`. Les données renvoyées par cette méthode sont considérées comme données de réponse et doivent être un objet de type `TerminalDataContainer`.
13. Si la commande de l'adaptateur cible ne génère pas d'écart ou n'obtient pas de réponse Confirm BOD (qui peut signaler une erreur), l'agent valide le message reçu à partir de la file d'attente de réception via l'adaptateur natif.
14. Si un accusé de réception a été demandé, l'agent appelle la méthode `sendResponse` sur l'adaptateur natif.
    - Si l'adaptateur cible a créé une réponse, il place l'identificateur logique de réponse du message d'origine dans le champ identificateur logique cible du message de réponse.
    - Si l'adaptateur cible n'a pas créé de réponse, l'agent crée un message de réponse Confirm BOD contenant l'état d'avancement.
        – En l'absence d'erreurs, l'état d'avancement est succès.
        – En présence d'erreurs, l'état d'avancement est positionné sur un état d'erreur.

15. La réponse est envoyée.
    a. L'agent envoie le message de réponse, si celui-ci a été créé, à l'adaptateur natif.
    b. L'adaptateur natif met le message de réponse dans la file d'attente de réponses.
    c. L'adaptateur natif envoie le message de réponse suivant le message d'origine reçu :
        - S'il s'agissait d'un message de demande MQSeries, l'adaptateur natif obtient alors les informations de file d'attente correspondant à la réponse au message de demande MQSeries. Ces informations de file d'attente se substituent à l'identificateur logique cible du message.
        - S'il ne s'agissait pas d'un message de demande MQSeries, l'adaptateur natif utilise la méthode `sendMsg` pour envoyer la réponse.
16. En cas d'écart ou de message de réponse Confirm BOD comportant un état d'erreur, l'agent enregistre un message d'écart dans un fichier des écarts appelé `EpicSystemExceptionFile`*nnnnnnnn*`.log` qui réside dans le même répertoire que le démon de l'adaptateur, *nnnnnnnn* représentant le numéro du fichier journal. Par ailleurs, si les classes WebSphere Business Integrator sont installées, elles envoient une erreur au composant WebSphere Business Integrator Solution Management. Pour plus d'informations, reportez-vous à la section «Messages d'erreur» à la page 100.
17. En cas d'écart ou de message de réponse Confirm BOD comportant un état d'erreur, l'agent ordonne à l'adaptateur natif de mettre le message d'origine dans la file d'attente des erreurs. Le nom de la file d'attente des erreurs est obtenu à partir du fichier de configuration en fonction de l'identificateur logique dépendant, de la catégorie et du type de corps du message d'origine.

    En fonction de l'identificateur d'application dépendante, de la catégorie et du type de corps, l'agent effectue une recherche en plusieurs étapes dans le fichier de configuration, dans l'ordre suivant :
    a. Valeurs de la catégorie et du type de corps spécifiques.
    b. Valeur de catégorie spécifique et valeur de type par défaut.
    c. Valeur de catégorie par défaut et valeur de type spécifique.
    d. Valeurs de la catégorie et du type de corps par défaut.
    - Si l'adaptateur natif est capable de mettre le message d'erreur dans la file d'attente des erreurs, ordre est donné à l'adaptateur natif de valider le message à partir de la file d'attente de réception.

- Si l'adaptateur natif est incapable de mettre le message d'erreur dans la file d'attente des erreurs, les opérations suivantes se produisent :
  a. L'agent ordonne à l'adaptateur natif d'annuler, c'est-à-dire de ne pas valider.
  b. L'agent positionne un drapeau ordonnant l'arrêt à tous les agents dépendant du démon de l'adaptateur. Cela signifie que le message comporte un incident. L'arrêt de tous les agents empêche les autres agents de procéder au retraitement du même message comportant l'incident avec le même résultat.
  c. Si une erreur mémoire insuffisante se produit, l'écart est traité de la même manière que tous les autres écarts, à cette exception près que l'agent positionne un drapeau pour lui-même lorsqu'il a terminé le traitement du message courant. Ceci permet aux autres agents de disposer d'une mémoire plus importante.

18. Lorsque l'adaptateur natif notifie l'agent que le travail est terminé, l'agent vérifie deux drapeaux :
    - Pour savoir si l'agent doit s'arrêter. Ceci peut être causé par un état de mémoire insuffisante Java.
    - Pour savoir si tous les agents doivent s'arrêter, la cause étant celle décrite à l'étape précédente.
19. Si l'un ou l'autre drapeau est positionné, l'agent s'arrête. Si aucun drapeau n'est positionné, l'agent traite le message suivant. L'agent demande à l'adaptateur natif de recevoir un message.
20. Si un message de réponse est mis dans la file d'attente de réponses ou si un message d'erreur est mis dans la file d'attente des erreurs, les opérations suivantes se produisent :
    a. MQSeries ou un autre logiciel d'application en mode message, bus applicatif, le renvoie au côté source du noyau.
    b. Si l'adaptateur source a appelé la méthode `sendRequestResponse` de son adaptateur natif, le noyau récupère alors le message dans la file d'attente de réponses et le renvoie à l'adaptateur source. Si l'adaptateur source a appelé la méthode `sendMsg`, le noyau met alors le message dans la file d'attente de réception de l'application source.

Capacités transactionnelles

Une *transaction* est un ensemble d'opérations qui doivent être exécutées en tant qu'unité indivisible de travail. Si toutes les opérations constituant une transaction sont réussies, la transaction est *validée*; c'est-à-dire que toutes les opérations sont exécutées. Si une ou plusieurs opérations constituant une transaction échoue(nt), la transaction est *annulée*; c'est-à-dire qu'aucune des opérations n'est exécutée. A l'aide des capacités transactionnelles du noyau MQSeries Adapter Kernel, un adaptateur source peut exécuter une série d'opérations comme une seule unité, avec l'assurance que toutes les

opérations aboutissent si la transaction est validée ou qu'aucune opération n'est exécutée si la transaction est annulée.

Les capacités transactionnelles peuvent être intégrées aux adaptateurs à l'aide de MQSeries Adapter Builder ou en utilisant les méthodes de `lancement`, d'`annulation` et de `validation` sur la classe `EpicNativeAdapter` de l'API Java du noyau. Si une méthode transactionnelle est appelée dans un contexte invalide (par exemple, appel de la méthode de `validation` sans avoir d'abord appelé la méthode de `lancement`, ou l'appel de `lancement` dans le cadre d'une autre transaction), le noyau ignore l'appel et émet un avertissement au programme trace. Reportez-vous à la section «Chapitre 5. Utilisation des Interfaces de Programmation d'Applications (API) de MQSeries Adapter Kernel» à la page 103 pour obtenir des informations sur l'utilisation de l'API.

Limitations

Les limitations suivantes sont associées aux capacités transactionnelles du noyau :

- Les transactions ne sont pas acceptées avec la méthode `sendRequestResponse`.
- Les transactions imbriquées (c'est-à-dire les transactions appelées à l'intérieur d'autres transactions) ne sont pas acceptées.
- Les transactions ne sont pas acceptées par tous les modes de communication ; reportez-vous à la section «Annexe A. Modes de communication» à la page 107 pour obtenir des détails.

Fonction de trace

Un message trace contient l'état de traitement d'un message à un moment donné dans le noyau. Vous pouvez utiliser les messages trace pour aider à diagnostiquer les incidents au niveau du noyau ou de vos adaptateurs. Le manuel MQSeries Adapter Kernel *Problem Determination Guide* traite de l'utilisation du programme trace avec le noyau.

Utilisation de MQSeries Adapter Kernel avec WebSphere Business Integrator et WebSphere Application Server

Cette section décrit l'utilisation de MQSeries Adapter Kernel avec les produits WebSphere Business Integrator et WebSphere Application Server. Pour plus de détails, reportez-vous à la documentation de WebSphere Business Integrator.

JMS Listener

WebSphere Business Integrator fournit un composant appelé JMS Listener, qui fonctionne avec MQSeries Adapter Kernel et WebSphere Application Server Advanced Edition pour assurer un autre moyen de communication des messages aux applications cible. JMS Listener s'exécute dans le cadre du serveur EJB (Enterprise JavaBeans) de WebSphere Application Server. Cette section présente les fonctionnalités de JMS Listener. Pour plus d'informations,

y compris des détails sur la configuration de WebSphere Business Integrator et de JMS Listener, reportez-vous à la documentation de WebSphere Business Integrator. Reportez-vous à la section «Configuration du noyau» à la page 62 pour savoir comment configurer MQSeries Adapter Kernel afin qu'il reconnaisse JMS Listener en tant que cible. L'utilisation de JMS Listener comme cible équivaut à envoyer un message à un démon d'adaptateur.

Avant de pouvoir utiliser JMS Listener, vous devez déployer un bean de message agent MQSeries Adapter Kernel, ainsi que des adaptateurs de bean de session de service Java ou des adaptateurs EAB pour le côté cible du noyau. Exécutez ces tâches à l'aide de MQSeries Adapter Builder. Dans un environnement WebSphere Business Integrator, le fonctionnement du noyau dans WebSphere Application Server est le même qu'avec un démon d'adaptateur autonome, sauf que c'est JMS Listener qui reçoit le message pour le compte de l'agent de l'adaptateur et appelle l'agent approprié. Dans un environnement MQSeries Adapter Kernel autonome, le démon de l'adaptateur démarre des agents d'adaptateurs, qui, à leur tour, reçoivent les messages directement.

Lorsque MQSeries Adapter Kernel fonctionne avec JMS Listener, les événements se succèdent dans l'ordre suivant :

1. JMS Listener, qui contrôle une file d'attente JMS, reçoit un objet message JMS, soit depuis un client EJB, soit depuis une application non-EJB.
2. JMS Listener instancie un *bean de message agent* et lui transmet l'objet message. Le bean de message agent est une instance d'un *bean de session*, un type de bean d'entreprise qui encapsule des données temporaires associées à un client particulier.
3. Le bean de message agent convertit l'objet message JMS en un objet collecteur de messages MQSeries.
4. En fonction des valeurs d'en-tête du message, le noyau appelle un adaptateur EAB ou EJB. Si l'adaptateur à appeler est de type EAB, le flot de données est le même que dans l'environnement autonome. Si l'adaptateur à appeler est de type EJB, un gestionnaire EJB est appelé pour effectuer les tâches suivantes :
   - Il détermine le bean de session de service (interface locale) à appeler, la méthode appropriée et le type de paramètre d'entrée de cette méthode pour l'objet `TerminalDataContainer`.
   - Il convertit les données applicatives contenues dans l'objet collecteur de messages en structure de données `TerminalDataContainer` appropriée pour le bean de session de service en utilisant une classe `Mapper`. L'objet `TerminalDataContainer` contient les métadonnées de l'objet collecteur de messages plus les objets application. Dans de nombreux cas, l'objet application est la chaîne de données de corps du document XML de l'objet collecteur de messages.

- Il appelle le bean de session de service, en transmettant l'objet `TerminalDataContainer` à la méthode appropriée sur ce bean. Le bean de session de service, qui fait partie de l'adaptateur de service Java, est la cible du message.

5. Si une réponse était demandée, le bean de message agent convertit l'objet de réponse `TerminalDataContainer` en objet collecteur de messages et envoie la réponse à l'aide de l'adaptateur natif.
6. Si une erreur se produit, le bean de message agent place l'objet collecteur de messages dans une file d'attente des erreurs à l'aide de l'adaptateur natif.

Support en langue nationale

MQSeries Adapter Kernel offre un support en langue nationale lorsque des adaptateurs Java sont utilisés. Par contre, ce support en langue nationale n'est pas assuré pour les adaptateurs C.

Chapitre 2. Planification de l'installation du noyau

Ce chapitre indique les exigences préalables relatives aux composants de MQSeries Adapter Kernel.

Pour connaître les derniers détails, consultez le site Web de la famille des produits MQSeries à l'adresse suivante :

www.ibm.com/software/ts/mqseries/

IBM se réserve le droit de mettre à jour les informations indiquées ici. Pour consulter les informations les plus récentes sur les versions des logiciels prises en charge, reportez-vous à l'adresse suivante :

www.ibm.com/software/ts/mqseries/platforms/supported.html

Matériel

MQSeries Adapter Kernel fonctionne sur les matériels suivants :

- Un PC IBM (ou compatible) fonctionnant sous Windows NT 4.0, Service Pack 5 ou ultérieur, ou Windows 2000, Service Pack 1.
- Un ordinateur RS/6000 IBM fonctionnant sous AIX version 4.3.2 ou 4.3.3.
- Un ordinateur HP Series 9000 fonctionnant sous HP-UX version 11.0.
- Un ordinateur Sun SPARC ou UltraSPARC fonctionnant sous Solaris version 8.
- Un ordinateur IBM AS/400 ou iSeries fonctionnant sous OS/400 version 4.4 ou 4.5.

Remarque : L'installation de MQSeries Adapter Kernel sur OS/400 nécessite l'interface d'un système Windows avec l'ordinateur AS/400. Pour plus d'informations, reportez-vous à la section «Exigences préalables à l'installation d'OS/400» à la page 36.

MQSeries Adapter Kernel requiert au moins 25 Mo d'espace disque pour les données et le code du produit.

Vérifier que l'espace disque disponible est suffisant pour contenir les adaptateurs. Leur taille dépend de la taille des structures de données, de la complexité des mappages et du code personnalisé utilisé. Vous trouverez ci-dessous différents exemples de tailles d'adaptateur sur les systèmes Windows. Les adaptateurs de votre site requièrent un espace disque plus ou moins important. Chaque exemple représente la source de l'adaptateur, le code adaptateur compilé, la source API et le code API compilé en Mo ou Ko.

- Adaptateur source pour l'ajout d'un bordereau de vente : 1.89 Mo
- Adaptateur cible pour la synchronisation d'un fichier client : 389 Ko
- Adaptateur cible pour la synchronisation d'un fichier inventaire : 161 Ko
- Adaptateur cible pour la synchronisation d'un article : 249 Ko
- Adaptateur cible pour la synchronisation d'un bordereau de vente : 579 Ko

Prévoyez en outre au moins 20 Mo d'espace de travail pour le noyau et les adaptateurs. L'espace de travail requis peut varier en fonction d'un certain nombre de facteurs, comme le nombre et la taille des files d'attente ainsi que la taille des fichiers trace.

Logiciel

Cette section indique les logiciels pris en charge par MQSeries Adapter Kernel. Les niveaux pris en charge sont indiqués. Reportez-vous à la section «Annexe B. Configurations validées» à la page 113. Notez que les compilateurs sont nécessaires sur les systèmes de développement mais pas sur les systèmes de production. Les compilateurs C indiqués ici ont été testés de manière concluante avec MQSeries Adapter Kernel ; d'autres compilateurs C peuvent fonctionner correctement avec le noyau, mais ils ne sont pas officiellement pris en charge.

Pour les systèmes Windows :

- Microsoft Windows NT version 4.0, Service Pack 5 ou ultérieur, ou Microsoft Windows 2000, Service Pack 1. Pour déterminer la version et le Service Pack de Microsoft Windows, ouvrez l'explorateur Windows, puis cliquez sur **Help > About Windows**.
- Compilateur Microsoft Visual C++ 6.0.
- MQSeries version 5.2 avec SupportPac MA88.
- IBM Java Development Kit (JDK) version 1.2.2 ou 1.3.

  Remarque : Windows NT et Windows 2000 sont actuellement les seules plates-formes sur lesquelles MQSeries Adapter Kernel prend en charge JDK version 1.3.

Pour AIX :

- Système d'exploitation AIX version 4.3.2 ou 4.3.3.
- IBM C Set++ pour AIX version 3.1.3.
- MQSeries version 5.2 avec SupportPac MA88.
- Java Development Kit version 1.2.2. Le JDK 1.3 n'est pas pris en charge.
- Système X Window (X11R5 ou supérieur). Il est nécessaire pour l'installation mais pas en phase d'exécution.

Pour HP-UX :

- Système d'exploitation HP-UX version 11.0.
- Compilateur C HP-UX C/ANSI. Consultez le fichier `readme.txt` pour plus de détails.
- MQSeries version 5.2 avec SupportPac MA88.
- Java Development Kit version 1.2.2. Le JDK 1.3 n'est pas pris en charge.
- Système X Window (X11R5 ou supérieur). Il est nécessaire pour l'installation mais pas en phase d'exécution.

Pour Solaris :

- Environnement d'exploitation Solaris version 8.
- Compilateurs C/C++ Sun Workshop. Consultez le fichier `readme.txt` pour plus de détails.
- MQSeries version 5.2 avec SupportPac MA88.
- Java Development Kit version 1.2.2. Le JDK 1.3 n'est pas pris en charge.
- Système X Window (X11R5 ou supérieur). Il est nécessaire pour l'installation mais pas en phase d'exécution.

Pour OS/400 :

- Système d'exploitation OS/400 version 4.4 ou 4.5, y compris les programmes suivants :
  - Java Toolkit et Java Developer Kit version 1.2.2. Le JDK 1.3 n'est pas pris en charge. Java Toolkit et Java Developer Kit sont livrés sous le numéro de licence de logiciel 5769–JV1. Reportez-vous à la section «Exigences préalables à l'installation d'OS/400» à la page 36 pour des détails supplémentaires sur les versions du Java Developer Kit nécessaire pour l'installation de MQSeries Adapter Kernel sur un système AS/400.
  - L'option Host Servers, qui est livrée sous le numéro de licence de logiciel 5769–SS1, option 12.
  - Qshell Interpreter, qui est livré sous le numéro de licence de logiciel 5769–SS1, option 30.
  - TCP/IP, qui est livré sous le numéro de licence de logiciel 5769–TC1.
  - Environnement ILE C pour AS/400, qui est livré sous le numéro de licence de logiciel 5769–CX2.
- MQSeries version 5.2 avec SupportPac MA88.

Reportez-vous à la section «Exigences préalables à l'installation d'OS/400» à la page 36 pour plus d'informations sur les exigences à respecter pour l'installation de MQSeries Adapter Kernel sur OS/400.

Les produits suivants sont pris en charge par MQSeries Adapter Kernel :

- MQSeries version 5.2 avec SupportPac MA88

  Remarque : Si MQSeries n'est pas utilisé, un autre logiciel de messagerie comme une implémentation du Java Message Service (JMS) doit être utilisé.
- MQSeries Integrator version 1.1
- MQSeries Integrator version 2

Reportez-vous à la section «Annexe B. Configurations validées» à la page 113 pour obtenir la liste des configurations validées MQSeries Adapter Kernel, MQSeries et MQSeries Integrator.

Exigences préalables à l'installation d'OS/400

Cette section décrit les exigences préalables à l'installation de MQSeries Adapter Kernel sur un système AS/400 ou iSeries. Reportez-vous au paragraphe 3 à la page 44 pour obtenir des instructions détaillées sur l'installation de MQSeries Adapter Kernel sur un système AS/400. Etant donné que les terminaux AS/400 ne supportent pas le système graphique Java, un poste de travail graphique comme un système Windows est nécessaire pour exécuter le programme d'installation GUI Java du noyau. L'établissement de l'interface entre le poste de travail et le système AS/400 peut s'effectuer de l'une des manières suivantes :

- Via un AWT distant, dans lequel tous les graphiques sont traités sur le système AS/400 et affichés sur le poste de travail. Ce processus est décrit plus en détails dans la section «Utilisation de l'AWT distant».
- Comme un client associé, dans lequel le poste de travail traite et affiche les graphiques. Ce processus est décrit plus en détails dans la section «Utilisation d'un client associé» à la page 37.

Cette section suppose l'utilisation d'un système Windows comme poste de travail graphique.

Utilisation de l'AWT distant

Lorsque l'AWT distant est utilisé, le traitement graphique Java est effectué sur le système AS/400 et les graphiques sont affichés sur un poste de travail client associé au système AS/400. Cette section décrit les exigences à respecter pour l'installation de MQSeries Adapter Kernel sur un système AS/400 à l'aide de l'AWT distant.

Les programmes suivants doivent être installés avec OS/400 :

- Java Toolkit et Java Developer Kit version 1.2.2. Java Toolkit et Java Developer Kit sont livrés sous le numéro de licence de logiciel 5769–JV1. Les fonctions AWT distantes sur OS/400 sont fournies par le Java Developer Kit.

- TCP/IP, qui est livré sous le numéro de licence de logiciel 5769–TC1. Pour plus d'informations sur TCP/IP, consultez les documents *AS/400 TCP/IP Fastpath Setup Information* et *AS/400 TCP/IP Configuration* , disponibles dans la bibliothèque AS/400 à l'adresse suivante : www.ibm.com/servers/eserver/iseries/library/.

Les exigences relatives au poste de travail sont les suivantes :

- Un PC IBM (ou compatible) fonctionnant sous Windows 95, Windows 98, Windows NT ou Windows 2000.
- Une connexion TCP/IP au système AS/400.
- JDK version 1.2.2 ou supérieure.

Pour configurer et démarrer un AWT distant, effectuez les opérations suivantes :

1. Vérifiez que JDK version 1.2.2 ou supérieure est installé sur le poste de travail.
2. Vérifiez qu'une connexion TCP/IP existe entre le système AS/400 et le poste de travail.
3. Copiez le fichier `RAWTGui.jar` du répertoire `/QIBM/ProdData/Java400/jdk12` sur le système AS/400 dans un répertoire sur le poste de travail.
4. Sur le poste de travail, passez au répertoire dans lequel vous avez copié le fichier `RAWTGui.jar` et démarrez l'AWT distant en entrant la commande suivante :

   ```
   java -jar RAWTGui.jar
   ```

Remarque : Etant donné que le traitement graphique Java absorbe une quantité de ressources importante sur un système AS/400, l'utilisation de l'AWT distant peut prendre bien plus de temps que l'utilisation d'un client associé pour l'installation de MQSeries Adapter Kernel.

Pour plus d'informations sur l'AWT distant, consultez la bibliothèque AS/400 à l'adresse suivante : www.ibm.com/servers/eserver/iseries/library/.

Utilisation d'un client associé

Lorsqu'un client associé est utilisé pour l'installation de MQSeries Adapter Kernel sur un système AS/400, le traitement graphique Java est effectué sur le poste de travail client et non sur le système AS/400. Cette section décrit les exigences à respecter pour l'installation de MQSeries Adapter Kernel sur un système AS/400 à l'aide d'un client associé.

Les programmes suivants doivent être installés avec OS/400 :

- Java Toolkit et Java Developer Kit version 1.2.2. Java Toolkit et Java Developer Kit sont livrés sous le numéro de licence de logiciel 5769–JV1.

- L'option Host Servers, qui est livrée sous le numéro de licence de logiciel 5769–SS1, option 12.
- TCP/IP, qui est livré sous le numéro de licence de logiciel 5769–TC1.

Les exigences relatives au poste de travail sont les suivantes :

- Un PC IBM (ou compatible) fonctionnant sous Windows NT 4.0, Service Pack 5 ou Windows 2000, Service Pack 1.
- Une connexion TCP/IP au système AS/400.
- JDK version 1.2.2 ou supérieure.

Composants du noyau

Après son installation, MQSeries Adapter Kernel réside dans son répertoire racine. Il contient des sous-répertoires qui peuvent à leur tour contenir d'autres répertoires. La racine et ses sous-répertoires sont indiqués, de même qu'un récapitulatif des fichiers se rapportant le plus à l'installation et à la configuration.

Racine

La désignation par défaut est `C:\Program Files\MQAK` sur les systèmes Windows, `/usr/lpp/mqak` sur AIX, `/MQAK/` sur HP-UX, `/opt/MQAK` sur Solaris et `/QIBM/ProdData/mqak` sur OS/400. Elle contient les éléments suivants :

- Tous les autres répertoires MQSeries Adapter Kernel.
- Le fichier `aqmsetenv.bat` (systèmes Windows) ou `aqmsetenv.sh` (UNIX), qui modifie les variables de l'environnement système après l'installation, le cas échéant.
- Le fichier `readme.txt`.
- Le fichier `aqmuninstall.bat` (systèmes Windows) ou `aqmuninstall.sh` (UNIX).

`Corbeille`

Elle contient les éléments suivants :

- Les bibliothèques de classe et les bibliothèques partagées.
- Les adaptateurs fournis comme partie intégrante du noyau, pour vérification uniquement.
- Le fichier `aqmversion.bat` (systèmes Windows) ou `aqmversion.sh` (UNIX et OS/400), un script exécuté pour afficher le numéro de la version du noyau.
- Le fichier `aqmcrtmsg.bat` (systèmes Windows) ou `aqmcrtmsg.sh` (UNIX et OS/400), un script exécuté pour créer un fichier XML utilisé pour valider le fichier de configuration avant qu'il ne soit mis en production.

- Le fichier `aqmsndmsg.bat` (systèmes Windows) ou `aqmsndmsg.sh` (UNIX et OS/400), un script exécuté pour valider le fichier de configuration avant sa mise en production.
- Le fichier `aqmstrad.bat` (systèmes Windows) ou `aqmstrad.sh` (UNIX et OS/400), un script exécuté pour démarrer le démon de l'adaptateur.
- Le fichier `aqmstrtd.bat` (systèmes Windows) ou `aqmstrtd.sh` (UNIX et OS/400), un script exécuté pour démarrer le serveur trace.

Documentation
Contient la documentation du produit, y compris le Centre d'aide et d'information.

Fichiers d'exécution
Contient les fichiers d'exécution du noyau.

Modèles

Contient les modèles d'adaptateurs ainsi que les fichiers de configuration et les fichiers utilitaires associés. Vous pouvez vous en servir pour effectuer des essais et vous initier.

Remarque : Le noyau doit être utilisé avec des adaptateurs créés via l'outil MQSeries Adapter Builder. Il ne doit pas être utilisé via des appels aux API de noyau uniquement à partir d'un code personnalisé. Les modèles d'adaptateurs sont uniquement fournis en vue d'aider l'utilisateur à comprendre le mode de fonctionnement du noyau et à l'établissement des diagnostics.

- Modèles d'adaptateurs
- Le fichier de configuration du noyau `aqmsetup` comportant des valeurs qui supportent les modèles d'adaptateurs. Reportez-vous à la section «Le fichier d'installation» à la page 68 pour obtenir une description de ce fichier.
- Le fichier de configuration du noyau `aqmconfig.xml`, comportant des valeurs qui supportent les modèles d'adaptateurs, y compris les valeurs trace des modèles. Reportez-vous à la section «Le fichier de configuration» à la page 69 pour obtenir une description de ce fichier.

Outils de développement
Contient un ensemble d'outils de développement de logiciels (SDK) comprenant les éléments suivants :

- Les fichiers d'en-tête.
- Les fichiers de bibliothèque utilisés pendant la compilation sous le système Windows.

Désinstaller
Contient des fichiers utilisés pour désinstaller le noyau.

Vérification
Contient les fichiers suivants qui assurent la vérification de l'installation du noyau :

- Le fichier `aqmverifyinstall.bat` (systèmes Windows) ou `aqmverifyinstall.sh` (UNIX et OS/400), un script exécuté pour vérifier l'installation du noyau sur un ordinateur.
- Le fichier `aqmcreateq.bat` (systèmes Windows) ou `aqmcreateq.sh` (UNIX et OS/400), un script qui crée les files d'attente MQSeries pour vérification. Pour plus d'informations, reportez-vous à la section «Création des files d'attente MQSeries» à la page 102.
- Le fichier `aqmconfig.xml`. Reportez-vous à la section «Le fichier de configuration» à la page 69 pour obtenir une description de ce fichier.
- Le fichier `aqmsetup`. Reportez-vous à la section «Le fichier d'installation» à la page 68 pour obtenir une description de ce fichier.
- Le fichier `aqminstalltest.xml`.

Chapitre 3. Installation du noyau

Le présent chapitre décrit les étapes nécessaires pour installer et vérifier and MQSeries Adapter Kernel. L'installation comprend les étapes générales suivantes :

Étape 1. Préparation de l'installation. Pour plus d'informations, reportez-vous à la section «Préparation de l'installation» à la page 42.

Étape 2. Installation du noyau. Pour plus d'informations, reportez-vous à la section «Installation du noyau» à la page 43.

Étape 3. Exécution de diverses opérations après l'installation. Pour plus d'informations, reportez-vous à la section «Exécution des étapes après installation» à la page 46.

Étape 4. Vérification de l'installation. Pour plus d'informations, reportez-vous à la section «Vérification de l'installation» à la page 49.

Ce chapitre aborde également les sujets suivants :

- Installation automatique de MQSeries Adapter Kernel. Pour plus d'informations, reportez-vous au «Installation automatique» à la page 54.
- Mise à niveau de MQSeries Adapter Kernel à partir d'une version antérieure. Pour plus d'informations, reportez-vous à la section «Mise à jour du noyau» à la page 56.
- Retrait d'une installation de MQSeries Adapter Kernel. Pour plus d'informations, reportez-vous à la section «Désinstallation du noyau» à la page 57.

Une fois le noyau installé, effectuez les opérations supplémentaires suivantes pour le préparer en vue de son utilisation :

1. Configurez le noyau. Pour plus d'informations, reportez-vous à la section «Configuration du noyau» à la page 62.
2. Configurez le logiciel de communication en mode message et le logiciel en option. Pour plus d'informations, reportez-vous à la section «Configuration de MQSeries et MQSeries Integrator» à la page 96.
3. Créez vos adaptateurs au moyen de MQSeries Adapter Builder, puis testez-les et déployez-les.
4. Démarrez le noyau. Pour plus d'informations, reportez-vous à la section «Démarrage du noyau» à la page 97.

Préparation de l'installation

MQSeries Adapter Kernel doit être installé par un administrateur ou un superutilisateur. Vous devez être autorisé à créer des fichiers et à y accéder à l'emplacement où vous avez installé MQSeries Adapter Kernel et à l'emplacement où se trouvent les deux fichiers de configuration du noyau. Le répertoire courant doit se trouver dans votre chemin d'accès exécutable. Assurez-vous que tous les ID utilisateur utilisant le noyau ont un droit de lecture, d'écriture et d'exécution.

Vous devez être autorisé à effectuer des opérations MQSeries telles que la création de gestionnaires de file d'attente et la création et l'accès aux files d'attente. Ces opérations sont effectuées de différentes manières selon les plates-formes. Pour plus d'informations, consulter le *MQSeries Administration Guide* de votre plate-forme.

L'identificateur utilisateur qui démarre les processus du noyau doit être dans le groupe mqm. Il existe deux catégories de processus du noyau :

- Démon de l'adaptateur, un pour chaque application cible prise en charge par l'ordinateur
- Serveur trace (en option)

Notez que l'adaptateur source est utilisé dans le processus de l'application source. Tout démon ou serveur contenant l'adaptateur source doit être démarré pour que cet adaptateur s'exécute.

Vous devez installer et configurer le noyau pour utiliser les adaptateurs que vous avez créés. Cependant, il n'est pas nécessaire d'installer le noyau pour installer MQSeries Adapter Builder ou pour l'utiliser pour créer vos adaptateurs.

Procédez comme suit avant de commencer l'installation :

- Lisez le fichier `readme.txt` sur le CD-ROM ou le réseau local. Ce fichier contient éventuellement des informations importantes devenues disponibles après l'achèvement de ce manuel. Il se trouve dans le répertoire principal de l'installation.
- Visitez le site Web à l'adresse suivante : www.ibm.com/software/ts/mqseries/ . Il peut contenir des informations importantes devenues disponibles après l'achèvement de ce manuel, y compris, éventuellement, une nouvelle édition de celui-ci.
- Si vous effectuez une mise à niveau à partir d'une version précédente de MQSeries Adapter Kernel, les instructions sont indiquées dans «Mise à jour du noyau» à la page 56.

- Assurez-vous que toutes les conditions préalables relatives au matériel et aux logiciels sont satisfaites. Pour plus d'informations, reportez-vous à la section «Matériel» à la page 33 et «Logiciel» à la page 34. MQSeries doit être installé et exécuté avant qu'il soit possible de vérifier l'installation de MQSeries Adapter Kernel. Assurez-vous que le support de MQSeries Java est installé et configuré.

Installation du noyau

Pour installer MQSeries Adapter Kernel sur un système Windows (Windows NT ou Windows 2000), sur une plate-forme UNIX (AIX, HP-UX, ou Solaris), ou sur OS/400, effectuez les opérations suivantes, propres à chaque système d'exploitation :

Sur les systèmes Windows :

Étape 1. Démarrez le programme d'installation comme suit :

- Si vous effectuez l'installation à partir d'un réseau local, passez dans le répertoire contenant les fichiers d'installation de MQSeries Adapter Kernel et exécutez le fichier`install.bat`.
- Si vous effectuez l'installation à partir du CD-ROM, insérez le CD-ROM MQSeries Adapter Kernel dans le lecteur de CD-ROM. Si l'exécution automatique est activée, le programme d'installation démarre automatiquement ; si l'exécution automatique n'est pas activée, exécutez le fichier `install.bat` se trouvant dans le répertoire principal du CD-ROM pour démarrer le programme d'installation.

Remarque : Sur les systèmes Windows, il n'est pas nécessaire de copier le fichier `install.bat` à un autre emplacement avant de l'exécuter. Pendant le processus d'installation, le programme vous demande de choisir où MQSeries Adapter Kernel doit être installé.

Étape 2. Suivez les messages présentés par le programme d'installation. Notez que si vous choisissez d'installer MQSeries Adapter Kernel à un emplacement autre que l'emplacement par défaut (sur les systèmes Windows, `C:\Program Files\MQAK`), vous devez indiquer le répertoire d'installation comme un nom de chemin complet et non comme un nom de chemin relatif.

Sur UNIX :

Étape 1. Démarrez le programme d'installation comme suit :

- Si vous effectuez l'installation à partir d'un réseau local, passez dans le répertoire contenant les fichiers d'installation de MQSeries Adapter Kernel et exécutez le script `install.sh`.

- Si vous effectuez l'installation à partir du CD-ROM, insérez le CD-ROM MQSeries Adapter Kernel dans le lecteur de CD-ROM et, si nécessaire, montez le lecteur de CD-ROM conformément à la documentation de votre système d'exploitation. Exécutez le script `install.sh` dans le répertoire principal du CD-ROM.

Étape 2. Suivez les messages présentés par le programme d'installation. Notez que si vous choisissez d'installer MQSeries Adapter Kernel à un emplacement autre que l'emplacement par défaut, vous devez indiquer le répertoire d'installation comme un nom de chemin complet et non comme un nom de chemin relatif. Les répertoires d'installation par défaut sur UNIX sont les suivants :

- AIX : `/usr/lpp/mqak`
- HP-UX : `/MQAK`
- Solaris : `/opt/MQAK`

Sur OS/400 :

Étape 1. Assurez-vous que toutes les conditions préalables indiquées dans les sections «Matériel» à la page 33, «composant logiciel prérequis OS/400» à la page 35, et «Exigences préalables à l'installation d'OS/400» à la page 36 sont satisfaites. Notez que l'installation de MQSeries Adapter Kernel sur OS/400 utilise un programme basé sur InstallShield nécessitant l'utilisation d'un interfaçage de poste de travail avec le système AS/400 ; pour plus de détails, reportez-vous à la section «Exigences préalables à l'installation d'OS/400» à la page 36.

Étape 2. Créez un profil utilisateur appelé MQAKSRV sur le système AS/400 en lançant la commande **CRTUSRPRF** au niveau d'une invite du langage de contrôle (CL).

Étape 3. Selon que vous utilisiez un AWT à distance ou un poste de travail de client associé pour réaliser l'installation, effectuez les opérations suivantes :

- Si vous utilisez un AWT à distance pour réaliser l'installation, veuillez procéder comme suit :
  a. Assurez-vous que l'AWT à distance est configuré et en fonctionnement. Pour plus d'informations, reportez-vous à la section «Utilisation de l'AWT distant» à la page 36.
  b. Assurez-vous que le fichier `installAS400.jar` est accessible au système AS/400. Le fichier doit se trouver soit dans le système de fichiers intégré (IFS), soit dans un périphérique connecté au système AS/400. Si le fichier se trouve dans un périphérique, utilisez la commande Create Link (**CRTLINK**) pour créer une liaison symbolique avec le fichier.

c. Pour améliorer les performances du processus d'installation, exécutez la commande Create Java Program(**CRTJVAPGM**) selon le fichier `installAS400.jar`.

d. Exécutez la commande Run Java (**RUNJVA**) comme suit, où *n.n.n.n* représente l'adresse TCP/IP du poste de travail qui utilise l'AWT à distance :

```
RUNJVA CLASS(run)
CLASSPATH('/installAS400.jar')
PROP((os400.class.path.rawt 1) (RmtAwtServer 'n.n.n.n')
(java.version 1.2))
```

- Si vous utilisez un poste de travail client associé pour réaliser l'installation, veuillez procéder comme suit :

  Étape a. Assurez-vous que les exigences indiquées dans la section «Utilisation d'un client associé» à la page 37 sont satisfaites.

  Étape b. Assurez-vous que l'option Serveurs Hôtes est installée et en fonctionnement sur la machine AS/400. Vous pouvez démarrer l'option Serveurs Hôtes en utilisant la commande Start Host Servers (**STRHOSTSVR**) au niveau d'une invite CL.

  Étape c. Assurez-vous que TCP/IP est installé et en fonctionnement sur la machine AS/400. Vous pouvez démarrer TCP/IP en utilisant la commande Start TCP/IP (**(STRTCP)**) au niveau d'une invite CL.

  Étape d. Sur le poste de travail, ouvrez une invite et passez dans le répertoire `AS400` du support d'installation de MQSeries Adapter Kernel (soit le réseau local, soit le CD-ROM).

  Étape e. Entrez la commande suivante :

  ```
  java -classpath installAS400.jar; run -os400
  ```

Étape 4. Le programme d'installation démarre et affiche**l'écran d'ouverture de session avec AS/400**. Entrez l'adresse TCP/IP de la machine AS/400 dans le champ **System:** et votre ID utilisateur, ainsi que le mot de passe dans les champs correspondants. Ne cochez pas la case **Default User**. Cliquez sur **Next**.

Étape 5. Suivez les messages présentés par le programme d'installation. Selon la vitesse de votre réseau et de vos machines, la procédure d'installation peut prendre jusqu'à une heure. Une barre de progression affichée sur le poste de travail indique l'état de l'installation.

Notez que sur OS/400, MQSeries Adapter Kernel est toujours installé dans le répertoire `/QIBM/ProdData/mqak` à la racine du système de fichiers intégré (IFS).

Étape 6. Configurez les variables d'environnement CLASSPATH, PATH, et QIBM_MULTI_THREADED comme suit :

- Ajoutez le répertoire `/QIBM/ProdData/mqak/bin` à la variable d'environnement CLASSPATH.
- Ajoutez le répertoire `/QIBM/ProdData/mqak/bin` à la variable d'environnement PATH.
- Paramétrez la variable d'environnement QIBM_MULTI_THREADED sur `Y`.

Étape 7. Ajoutez la bibliothèque MQAK à la liste des bibliothèques QSYS.LIB.

L'installation du noyau est terminée. Tel qu'il est installé, le noyau est configuré pour prendre en charge la vérification et non pour prendre en charge la production dans votre site en particulier. Vérifiez l'installation en effectuant les opérations indiquées dans la section «Vérification de l'installation» à la page 49. Après avoir vérifié l'installation, suivez les étapes de la section «Exécution des étapes après installation» pour paramétrer les variables d'environnement et déplacer plusieurs fichiers de configuration pour la prise en charge de la production sur votre site.

Installez le noyau sur d'autres ordinateurs, si besoin.

Exécution des étapes après installation

Une fois le noyau installé, effectuez les opérations suivantes :

Étape 1. Déterminez où placer les fichiers `aqmsetup` et `aqmconfig.xml`, qui servent à configurer le noyau. Pour plus d'informations sur ces fichiers, reportez-vous à la section «Configuration du noyau» à la page 62.

ATTENTION :
Si vous ne créez pas vos propres fichiers de configuration et si vous utilisez les fichiers de configuration fournis dans le répertoire `samples` pour la production, l'installation d'une nouvelle version du noyau écrasera ces fichiers et détruira votre configuration de production.

Étape 2. Créez un répertoire pour les deux fichiers de configuration. Il n'est pas nécessaire qu'ils soient placés dans le même répertoire, mais il est recommandé de le faire pour simplifier les opérations. Si vous les placez hors du répertoire où vous avez installé MQSeries Adapter Kernel, il restera moins de répertoires si le noyau est désinstallé ultérieurement. Le procédé de désinstallation laisse les répertoires qui contiennent tous les fichiers qui ne sont pas les fichiers originaux de MQSeries Adapter Kernel.

Étape 3. Copiez les fichiers aqmsetup et aqmconfig.xml du répertoire samples à l'emplacement désiré. Vous pouvez les placer sur une unité de réseau ou sur un autre emplacement central accessible par plusieurs ordinateurs afin de faciliter leur mise à jour et leur sauvegarde.

Si vous renommez le fichier aqmconfig.xml, le noyau ne fonctionnera pas correctement. Vous pouvez renommer le fichier aqmsetup, sous réserve que vous définissiez une variable d'environnement qui le pointera correctement à l'étape 5.

Étape 4. A l'aide d'un éditeur de texte, éditez le fichier aqmsetup qui pointera sur le répertoire désiré du fichier aqmconfig.xml. Utilisez un nom de chemin complet (et non un nom de chemin relatif) pour l'emplacement du répertoire. N'indiquez pas le nom du fichier lui-même dans le chemin. Vous trouverez un exemple ci-dessous :

```
# Location of configuration file aqmconfig.xml.
AQMCONFIG=C:\Program Files\MQAK\Data\
```

Même si l'emplacement choisi pour le fichier aqmconfig.xml est le répertoire où réside le fichier aqmsetup, vous devez entrer ici le nom de fichier complet. Sauvegardez et fermez le fichier aqmsetup.

Étape 5. Paramétrez la variable d'environnement AQMSETUPFILE de manière à pointer sur l'emplacement du fichier aqmsetup (par exemple, C:\Program Files\MQAK\Data\aqmsetup sur les systèmes Windows, /MQAK/data/aqmsetup sur UNIX, ou /home/*user_name*/aqmsetup sur OS/400). Notez que sur OS/400, le fichier aqmsetup doit toujours être placé dans le répertoire IFS personnel courant de l'utilisateur (c'est-à-dire, /home/*user_name*).

Si le noyau est installé sur une unité de réseau, effectuez cette opération pour chaque ordinateur y ayant accès.

Étape 6. Si vous utilisez AIX et si vous envisagez d'utiliser des adaptateurs source du langage C natif, appelés dans un programme C, paramétrez la variable d'environnement AIXTHREAD_SCOPE à la valeur S. Pour paramétrer cette variable d'environnement dans l'interpréteur de commandes Bourne ou Korn, entrez la commande suivante :

```
export AIXTHREAD_SCOPE=S
```

Pour paramétrer cette variable d'environnement dans l'interpréteur de commandes C, entrez la commande suivante :

```
setenv AIXTHREAD_SCOPE S
```

Pour que la variable AIXTHREAD_SCOPE soit paramétrée automatiquement lorsque vous ouvrez une session avec AIX, ajoutez cette commande dans votre fichier `.profile` (si vous utilisez l'interpréteur de commandes Bourne ou Korn) ou votre fichier `.cshrc` (si vous utilisez l'interpréteur de commandes C).

Reportez-vous à la section 7 à la page 24 pour plus d'informations concernant les méthodes de programmation.

Étape 7. Si nécessaire, paramétrez la variable d'environnement THREADS_FLAG. Vous ne devez paramétrer cette variable que si *toutes* les conditions suivantes sont vraies :

- Solaris est le système d'exploitation utilisé.
- La version 1.2.2 du Java Development Kit (JDK) est la version utilisée.
- MQSeries est utilisé pour transporter des messages.
- Vos adaptateurs source et cible sont écrits en langage C.

Si toutes ces conditions sont vraies, paramétrez la variable d'environnement THREADS_FLAG sur `native`. Pour paramétrer cette variable d'environnement dans l'interpréteur de commandes Bourne ou Korn, entrez la commande suivante :

```
export THREADS_FLAG=native
```

Pour paramétrer cette variable d'environnement dans l'interpréteur de commandes C, entrez la commande suivante :

```
setenv THREADS_FLAG native
```

Pour que la variable THREADS_FLAG soit paramétrée automatiquement lors de la connexion avec Solaris, ajoutez cette commande à votre fichier `.profile` (si vous utilisez l'interpréteur de commandes Bourne ou Korn) ou à votre fichier `.cshrc` (si vous utilisez l'interpréteur de commandes C).

Une fois les étapes après installation terminées, procédez comme suit pour préparer le noyau en vue de son utilisation :

1. Préparez la production. Pour plus d'informations, reportez-vous à la section «Préparation de la mise en service» à la page 61.
2. Editez le fichier de configuration. Pour plus d'informations, reportez-vous à la section «Configuration du noyau» à la page 62.
3. Configurez le logiciel MQSeries et le logiciel en option. Reportez-vous à la section «Configuration de MQSeries et MQSeries Integrator» à la page 96.
4. Pour les systèmes de production, tenez compte des «Recommandations en matière de performances» à la page 96.

5. Démarrez le noyau. Pour plus d'informations, reportez-vous à la section «Démarrage du noyau» à la page 97.
6. Etablissez un plan de maintenance du noyau. Pour plus d'informations, reportez-vous à la section «Maintenance du noyau» à la page 99.

Vérification de l'installation

Après avoir installé le noyau, assurez-vous qu'il a été installé correctement en exécutant un script de vérification. Ce script envoie un message d'essai à partir d'une application source en utilisant un adaptateur source, puis vers MQSeries en utilisant le noyau. Il utilise ensuite le noyau pour recevoir le message en provenance de MQSeries, puis appelle un adaptateur cible. Tous ces procédés sont effectués sur un seul ordinateur.

Dans cette vérification, l'application source est une file d'attente MQSeries appelée TEST1. L'application cible est une autre file d'attente MQSeries appelée TEST2.

La vérification effectue les tâches suivantes :

- Elle s'assure que le noyau, avec l'adaptateur source et l'adaptateur cible fournis, a classé et routé correctement le message d'essai, en utilisant MQSeries comme logiciel de communication en mode message, de bout en bout dans l'ordinateur.
- Elle vérifie les fichiers `aqmconfig.xml` et `aqmsetup` fournis lors de l'installation. Ils déterminent la configuration du noyau. Pour plus d'informations sur ces fichiers, reportez-vous à «Configuration du noyau» à la page 62.

Vous pouvez valider le fichier de configuration avant de le mettre en production. Pour plus d'informations, reportez-vous à la section «Validation du fichier de configuration» à la page 92.

Les scripts de vérification de l'installation qui sont fournis avec MQSeries Adapter Kernel supposent que MQSeries est installé et configuré sur la machine dans laquelle les scripts doivent être exécutés. Si vous utilisez un logiciel de communication en mode message autre que MQSeries, vous pouvez éditer les scripts de vérification de l'installation pour la prise en charge de votre logiciel de communication en mode message comme suit :

1. passez dans le répertoire `verificationde` l'installation du noyau.
2. Ouvrez le fichier `aqmconfig.xml` dans un éditeur de texte et remplacez la ligne `<epicmqppqueuemgr>DEFAULT</epicmqppqueuemgr>` par `<epicmqppqueuemgr>`*queue_manager_name*`</epicmqppqueuemgr>`, où *queue_manager_name* est le nom de votre gestionnaire de file d'attente.

3. Editez le fichier aqmverifyinstall comme suit :
   - Si vous effectuez une vérification d'installation sur un système Windows, ouvrez le fichier aqmverifyinstall.bat dans un éditeur de texte et remplacez la ligne aqmcreateq TEST2 par aqmcreateq TEST2 *queue_manager_name*, où *queue_manager_name* est le nom de votre gestionnaire de file d'attente.
   - Si vous effectuez une vérification d'installation sur UNIX ou OS/400, ouvrez le fichier aqmverifyinstall.sh dans un éditeur de texte et remplacez la ligne aqmcreateq.sh TEST2 par aqmcreateq.sh TEST2 *queue_manager_name*, où *queue_manager_name* est le nom de votre gestionnaire de file d'attente.

Cette vérification utilise certains composants, tels qu'un nom d'adaptateur cible, com.ibm.epic.adapters.eak.test.InstallVerificationTest, qui ne font pas partie du noyau. Ils sont fournis avec le noyau uniquement pour la vérification de l'installation.

Lorsque la vérification est terminée, le démon de l'adaptateur de vérification est arrêté.

Le traçage n'est pas activé pendant la vérification.

Procédure de vérification

Étape 1. La vérification crée et utilise trois files d'attente MQSeries. Si ces files d'attente comportent des messages avant que la vérification ne soit réalisée, celle-ci échoue. Supprimez les messages dans les files d'attente suivantes :
- TEST2AIQ
- TEST2AEQ
- TEST2RPL

Étape 2. Assurez-vous que vous êtes autorisé à installer et vérifier le noyau. Reportez-vous à la section«Préparation de l'installation» à la page 42.

Étape 3. Lancez la vérification comme suit :
- Sur les systèmes Windows, cliquez deux fois sur le fichier aqmverifyinstall.bat dans le répertoire verification. Vous pouvez également ouvrir une fenêtre d'invite, passer au répertoire verification et exécuteraqmverifyinstall.bat.
- Sur UNIX, ouvrez un terminal, passez dans le répertoire verification et exécutez le fichier aqmverifyinstall.sh.
- Sur OS/400, effectuez les opérations suivantes :
  a. Lancez une session **qsh** en entrant la commande **STRQSH**.
  b. Copiez le fichier /QIBM/ProdData/mqak/verification/aqmsetup sur votre répertoire personnel(/home/*user_name*).

c. Passez dans le répertoire /QIBM/ProdData/mqak/verification.
d. Exécutez le fichier aqmverifyinstall.sh.

Le fichier aqmverifyinstall contient des remarques concernant son fonctionnement.

Étape 4. Le message Installation Verification Test completed successfully indique que l'opération s'est déroulée correctement. Fermez la fenêtre de vérification, si nécessaire.

Étape 5. En cas d'incident, examinez la fenêtre de vérification et le fichier de consignation, EpicSystemExceptionFile*nnnnnnnn*.log, afin de déterminer l'erreur.

Étape 6. Se reporter à la section «Problèmes de vérification courants» pour les incidents courants pouvant être rencontrés pendant la vérification et pour les réponses possibles.

Étape 7. Si vous le désirez, effectuez une vérification en option. Pour plus d'informations, reportez-vous au «Vérification en option» à la page 54.

Étape 8. Retournez à la procédure d'installation et configurez le noyau pour la prise en charge de l'exploitation dans votre site particulier. Passez à l'opération 1 à la page 46.

Problèmes de vérification courants

Cette section indique les problèmes courants pouvant être rencontrés pendant la vérification, avec les solutions possibles. Les informations importantes des messages d'erreur sont indiqués en **caractères gras**.

Problème : Le fichier aqmsetup n'a pas été trouvé.

Réponse : S'assurer que la variable d'environnement AQMSETUPFILE est positionnée à l'emplacement du fichier aqmsetup dans le répertoire verification.

Message d'erreur :

```
com.ibm.epic.adapters.eak.nativeadapter.EpicNativeAdapter::main: caught
throwable with message <AQM0002: com.ibm.epic.adapters.eak.common.
AdapterDirectory::getProperties():
Received exception <com.ibm.epic.adapters.eak.common.AdapterException>
Message information: <AQM0002: com.ibm.epic.adapters.eak.common.
AdapterCfg::readConfig(String):
Received exception <java.io.FileNotFoundException> Message information:
<C:\aqmsetup> Additional program information <>.>
Additional program information <Error Reading Configuration File
[File or Keys in file may not exist]>.>
```

Problème : Le fichier aqmconfig.xml n'a pas été trouvé.

Réponse : Editez le fichier aqmsetup dans le répertoire verification et assurez-vous que l'entrée AQMCONFIG= pointe vers le répertoire verification. Utilisez un nom de chemin complet. Assurez-vous également que le fichier aqmconfig.xml se trouve dans le répertoire verification.

Message d'erreur :

```
com.ibm.epic.adapters.eak.common.AdapterException: MessageID <AQM0002>
<AQM0002: com.ibm.epic.adapters.eak.common.AdapterDirectory::
getProperties(): Received exception
<java.io.FileNotFoundException> Message information:
<AQMCONFIG.xml> Additional program information <>.>
```

Problème : La file d'attente sur laquelle le message devait être placé n'existait pas.

Réponse : Utilisez MQSeries afin de vous assurer que la file d'attente indiquée dans le message d'erreur (TEST2AIQ lorsque l'installation est en cours de vérification) existe et peut accepter des messages. Pour plus d'informations, reportez-vous à la section «Création des files d'attente MQSeries» à la page 102.

Message d'erreur :

```
com.ibm.epic.adapters.eak.nativeadapter.EpicNativeAdapter::main: caught
throwable with message
<AQM0107: com.ibm.epic.adapters.eak.nativeadapter.LMSMQbase::
createMQOutputQueue(String):
Received MQException creating queue, QManager name <DEFAULT>
Queue name <TEST2AIQ>:
completion code <2> reason code <2085>.>
```

Problème : L'adaptateur cible n'a pas été trouvé.

Réponse : Assurez-vous que l'adaptateur cible indiqué dans le message existe : com.ibm.epic.adapters.eak.test.InstallVerificationTest. Assurez-vous que la variable d'environnement CLASSPATH inclut le répertoire bin du noyau.

Message d'erreur :

```
com.ibm.epic.adapters.eak.adapterdaemon.EpicAdapterWorker::sendException
(Throwable, String):Thread-2:
Message <<TEST2> <2000.05.18.09.41.43.781> <<Processing Messages.>
<com.ibm.epic.adapters.eak.common.AdapterException: MessageID <AQM0002>
<AQM0002: com.ibm.epic.adapters.eak.adapterdaemon.EpicAdapterWorker:
:instantiateClass(String, Class[], Object[]): Received exception
<java.lang.ClassNotFoundException> Message information:
<com.ibm.epic.adapters.eak.test.InstallVerificationTest>
Additional program information <[Cannot obtain Class for class name
<com.ibm.epic.adapters.eak.test.InstallVerificationTest>]>.>>>>
```

Problème : Aucun adaptateur à charger pour la transmission du message n'a été trouvé. L'identificateur logique de destination n'a pas d'entrée dans le fichier `aqmconfig.xml` pour le type de corps et la classe de corps spécifiés dans le message de la file d'attente.

Réponse : Pendant la vérification, la cause la plus probable de ce message d'erreur est l'existence de messages sur une file d'attente appelée TEST2AIQ avant la vérification. Supprimez tous les messages de la file d'attente TEST2AIQ et essayez de nouveau la vérification. La seule entrée pour un nom de classe de commande pour l'application TEST2 dans le fichier `aqmconfig.xml` du répertoire `verification` concerne un type de corps TESTBOD et une classe de corps OAG.

Message d'erreur :

```
com.ibm.epic.adapters.eak.adapterdaemon.EpicAdapterWorker::sendException
(Throwable, String):Thread-2: Message <<TEST2> <2000.05.18.10.28.43.105>
<<Processing Messages.> <com.ibm.epic.adapters.eak.common.
AdapterException:
MessageID <AQM0401> <AQM0401: com.ibm.epic.adapters.eak.
adapterdaemon.EpicAdapterWorker::processMessage(EpicMessage):
Cannot obtain Command class name to load for a received message.>>>>
```

Problème : Le gestionnaire de file d'attente de vérification n'était pas lancé.

Réponse : Assurez-vous que le gestionnaire de file d'attente MQSeries a été lancé correctement.

Message d'erreur :

```
com.ibm.epic.adapters.eak.common.AdapterException: Message ID <AQM0104>
<AQM0104: com.ibm.epic.adapters.eak.nativeAdapter.queueCollection::
constructor(String,String,boolean,String,String,int):
Received MQException creating QManager connection for
QManager name <QMGRNAME>
MQ Message information: completion code <2> reason code <2059>.>
```

Problème : Une erreur générale de MQSeries s'est produite.

Réponse : Assurez-vous que MQSeries est installé et configuré correctement et qu'il est exécuté sur la machine. Examinez le code anomalie de MQException et utilisez le document *MQSeries Messages* pour déterminer la cause de ce code.

Message d'erreur :

```
Received MQException "ACTION ATTEMPTED." Message information:
completion code <completion_code> reason code <reason_code>
```

Vérification en option

Après avoir vérifié que le noyau a été installé correctement sur le premier ordinateur, vous pouvez effectuer les opérations optionnelles suivantes :

1. Vérifiez que le noyau est installé correctement sur un deuxième ordinateur, en utilisant la même vérification.
2. Vérifiez que vous pouvez envoyer un message d'essai à partir d'un adaptateur source situé sur un ordinateur à un adaptateur cible situé sur un autre ordinateur. Configurez et effectuez manuellement cette vérification. Si vous choisissez d'effectuer cette vérification en modifiant les fichiers de vérification originaux fournis avec le noyau, conservez une copie des fichiers originaux de vérification à des fins de sauvegarde.

Installation automatique

Il est possible d'installer MQSeries Adapter Kernel sur toutes les plates-formes à l'aide d'une procédure d'*installation automatique*. Cette procédure remplace le programme d'installation de MQSeries Adapter Kernel, dans lequel vous devez sélectionner manuellement les options d'installation voulues. L'installation automatique est utile si vous souhaitez installer la configuration par défaut sur plusieurs machines.

Pour installer le noyau automatiquement, effectuez les opérations suivantes spécifiques au système d'exploitation :

Sur les systèmes Windows :

Étape 1. Ouvrez une fenêtre d'invite et passez dans le répertoire contenant les fichiers d'installation de MQSeries Adapter Kernel.

Étape 2. Entrez la commande suivante :

```
java -cp install.jar run -P product.installLocation="empl_install"
-silent
```

où *empl_install* désigne l'emplacement d'installation de votre choix (par exemple, `D:\mqak`).

Sur UNIX :

Étape 1. Sur un terminal, passez dans le répertoire contenant les fichiers d'installation de MQSeries Adapter Kernel. Si vous effectuez l'installation à partir du CD-ROM, insérez le CD-ROM MQSeries Adapter Kernel dans le lecteur de CD-ROM et, si nécessaire, montez le lecteur de CD-ROM conformément à la documentation de votre système d'exploitation.

Étape 2. Entrez la commande suivante :

```
java -cp install.jar run -P product.installLocation="empl_install"
-silent
```

où *empl_install* désigne l'emplacement d'installation de votre choix (par exemple, `/opt/mqak`).

Sur OS/400 :

Si vous n'utilisez pas un poste de travail client associé pour accéder à la machine AS/400, procédez comme suit :

Étape 1. Assurez-vous que le fichier `installAS400.jar` est accessible au système AS/400. Le fichier doit se trouver soit dans le système de fichiers intégré (IFS), soit dans un périphérique connecté au système AS/400. Si le fichier se trouve dans un périphérique, utilisez la commande Create Link (**CRTLINK**) pour créer une liaison symbolique avec le fichier.

Étape 2. Pour améliorer les performances du processus d'installation, exécutez la commande Create Java Program(**CRTJVAPGM**) selon le fichier `installAS400.jar`.

Étape 3. Selon que vous utilisez une invite CL ou une session **qsh**, entrez l'une des commandes suivantes :

- A partir d'une invite CL :

  ```
  RUNJVA CLASS(run)
  CLASSPATH('/installAS400.jar')
  PROP((java.version 1.2)) PARM('-silent')
  ```

- A partir d'une session **qsh** :

  ```
  java -Djava.version=1.2 -classpath installAS400.jar run -silent
  ```

Si vous utilisez un poste de travail client associé pour communiquer avec le système AS/400, procédez comme suit :

Étape 1. Assurez-vous que les exigences indiquées dans la section «Utilisation d'un client associé» à la page 37 sont satisfaites.

Étape 2. Assurez-vous que l'option Serveurs Hôtes est installée et en fonctionnement sur la machine AS/400. Vous pouvez démarrer l'option Serveurs Hôtes en utilisant la commande Start Host Servers (**STRHOSTSVR**) au niveau d'une invite du langage de contrôle (CL).

Étape 3. Assurez-vous que TCP/IP est installé et en fonctionnement sur la machine AS/400. Vous pouvez démarrer TCP/IP en utilisant la commande Start TCP/IP (**(STRTCP)**) au niveau d'une invite CL.

Étape 4. Sur le poste de travail, ouvrez une invite et passez dans le répertoire `AS400` du support d'installation de MQSeries Adapter Kernel (soit le réseau local, soit le CD-ROM).

Étape 5. Entrez la commande suivante :

```
java -cp installAS400.jar run -silent -os400 nom_machine ID_util
mot_de_passe
```

où *nom_machine* est l'adresse TCP/IP du système AS/400, *ID_util* est votre ID utilisateur et *mot_de_passe*, votre mot de passe.

Mise à jour du noyau

Si vous avez installé la version 1.0 de MQSeries Adapter Kernel, avec ou sans la Corrective Service Diskette (CSD) (disquette de maintenance corrective), ou la version 1.1 de MQSeries Adapter Kernel avec un niveau de modification antérieur, effectuez les opérations suivantes avant d'installer la version 1.1 de MQSeries Adapter Kernel au niveau de modification actuel :

Étape 1. Sauvegardez les fichiers `aqmsetup` et `aqmconfig` (`aqmconfig.properties` ou `aqmconfig.xml`) en dehors du répertoire d'installation de MQSeries Adapter Kernel.

Étape 2. Si une CSD de MQSeries Adapter Kernel est installée, désinstallez-la comme suit :

- Sur Windows NT, utilisez l'une des méthodes suivantes :
  - Dans le menu Start de Windows NT, cliquez sur **Programs > MQSeries Adapter Kernel > Remove CSD**.
  - Utilisez l'utilitaire Ajout/Suppression de programmes du panneau de configuration.
  - Exécutez le fichier `aqmuninstallCSD.bat` se trouvant dans le répertoire principal du noyau.
  - Ouvrez une invite, passez dans le répertoire principal du noyau et entrez la commande suivante :

    ```
    java uninstallCSD
    ```
- Sur AIX, passez dans le répertoire principal du noyau et entrez l'une des commandes suivantes :

  ```
  aqmuninstallCSD.sh
  ```

  ```
  java uninstallCSD
  ```

Étape 3. Désinstallez MQSeries Adapter Kernel comme suit :

- Sur Windows NT, utilisez l'une des méthodes suivantes :
  - Dans le menu Start de Windows NT, cliquez sur **Programs > MQSeries Adapter Kernel > Uninstall MQSeries Adapter Kernel**.
  - Utilisez l'utilitaire Ajout/Suppression de programmes du panneau de configuration.
  - Exécutez le fichier `aqmuninstall.bat` se trouvant dans le répertoire principal du noyau.

– Ouvrez une invite, passez dans le répertoire principal du noyau et entrez la commande suivante :

```
java uninstall
```

- Sur AIX, passez dans le répertoire principal du noyau et entrez l'une des commandes suivantes :

```
aqmuninstall.sh
```

```
java uninstall
```

Étape 4. Installez la version 1.1 de MQSeries Adapter Kernel. Pour plus d'informations, reportez-vous à la section «Installation du noyau» à la page 43.

Étape 5. Restaurez les fichiers `aqmsetup` et `aqmconfig` à leurs emplacements précédents dans le répertoire d'installation de MQSeries Adapter Kernel. Si nécessaire, convertissez le fichier `aqmconfig.properties` en fichier `aqmconfig.xml`. Pour plus d'informations sur le fichier`aqmconfig.xml`, reportez-vous à la section«Le fichier de configuration» à la page 69.

Désinstallation du noyau

Il existe plusieurs manières de désinstaller le noyau. Notez que la procédure de désinstallation n'affecte aucun fichier ou répertoire créé après l'installation du noyau. Cela comprend tous les fichiers historiques et les fichiers de données copiés par l'utilisateur.

- Sur les systèmes Windows, utilisez l'une des méthodes suivantes :
  - Dans le menu Start, cliquez sur **Programs > IBM MQSeries Adapter Kernel > Uninstall MQSeries Adapter Kernel**.
  - Utilisez l'utilitaire Ajout/Suppression de programmes du panneau de configuration.
  - Exécutez le fichier `aqmuninstall.bat` se trouvant dans le répertoire principal du noyau.
  - Pour désinstaller le noyau automatiquement (c'est-à-dire, sans que le programme de désinstallation vous demande des détails ou une confirmation), ouvrez une invite, passez dans le répertoire d'installation du noyau et entrez la commande suivante :

```
java -cp uninstall.jar run -silent
```

- Sur UNIX, passez dans le répertoire principal du noyau et entrez la commande suivante :

```
aqmuninstall.sh
```

Pour désinstaller le noyau automatiquement (c'est-à-dire, sans que le programme de désinstallation vous demande des détails ou une confirmation), passez dans le répertoire principal du noyau et entrez la commande suivante :

```
java -cp uninstall.jar run -silent
```

- Sur OS/400, utilisez l'une des méthodes suivantes pour désinstaller le noyau.
  - Si vous utilisez un AWT à distance pour désinstaller le noyau, procédez comme suit :

    Étape 1. Assurez-vous que l'AWT à distance est configuré et en fonctionnement. Pour plus d'informations, reportez-vous à la section «Utilisation de l'AWT distant» à la page 36.

    Étape 2. Pour améliorer l'exécution du processus de désinstallation, exécutez la commande Create Java Program (**CRTJVAPGM**) selon le fichier `/QIBM/ProdData/mqak/uninstall/uninstall.jar`.

    Étape 3. Exécutez la commande Run Java (**RUNJVA**) comme suit, où *n.n.n.n* représente l'adresse TCP/IP du poste de travail qui utilise l'AWT à distance :

    ```
    RUNJVA CLASS(run)
    CLASSPATH('/QIBM/ProdData/mqak/uninstall/uninstall.jar')
    PROP((os400.class.path.rawt 1) (RmtAwtServer 'n.n.n.n')
    (java.version 1.2))
    ```

  - Si vous utilisez un poste de travail client associé pour désinstaller le noyau, procédez comme suit :

    Étape 1. Assurez-vous que les exigences indiquées dans la section «Utilisation d'un client associé» à la page 37 sont satisfaites.

    Étape 2. Assurez-vous que l'option Serveurs Hôtes est installée et en fonctionnement sur la machine AS/400. Vous pouvez démarrer l'option Serveurs Hôtes en utilisant la commande Start Host Servers (**STRHOSTSVR**) au niveau d'une invite du langage de contrôle (CL).

    Étape 3. Assurez-vous que TCP/IP est installé et en fonctionnement sur la machine AS/400. Vous pouvez démarrer TCP/IP en utilisant la commande Start TCP/IP (**(STRTCP)**) au niveau d'une invite CL.

    Étape 4. Copiez les fichiers `uninstall.jar` et `uninstall.dat`à partir du répertoire`/QIBM/ProdData/mqak/uninstall` du système AS/400 vers un répertoire du poste de travail client.

    Étape 5. Entrez la commande suivante :

    ```
    java -classpath uninstall.jar; run -os400
    ```

Pour désinstaller le noyau automatiquement (c'est-à-dire, sans que le programme de désinstallation vous demande des détails ou une confirmation), entrez la commande suivante :

```
java -cp uninstall.jar run -silent -os400 nom_machine ID_util
mot_de_passe
```

où *nom_machine* est l'adresse TCP/IP du système AS/400, *ID_util* est votre ID utilisateur et *mot_de_passe*, votre mot de passe.

– Si vous travaillez directement sur le système AS/400 à partir d'une invite CL ou d'une session **qsh**, entrez l'une des commandes suivantes pour désinstaller le noyau automatiquement (c'est-à-dire, sans que le programme de désinstallation vous demande des détails ou une confirmation) :

  - A partir d'une invite CL :

    ```
    RUNJVA CLASS(run)
    CLASSPATH('/uninstall.jar')
    PROP((java.version 1.2)) PARM('-silent')
    ```

  - A partir d'une session **qsh** :

    ```
    java -Djava.version=1.2 -classpath uninstall.jar run -silent
    ```

Chapitre 4. Utilisation du noyau

Ce chapitre contient les informations suivantes concernant l'utilisation du noyau :

Préparation de la mise en service

Avant de mettre le noyau en service, effectuez les tâches suivantes :

1. Définissez l'ensemble de l'architecture du système, y compris MQSeries Adapter Offering, MQSeries ou autres logiciels de communication en mode message et, en option, MQSeries Integrator, sur la base des exigences et conditions de votre site. L'architecture est généralement unique pour chaque site.
2. Créez les adaptateurs source et cible nécessaires en utilisant MQSeries Adapter Builder, puis testez-les et déployez-les.
3. Développez des interfaces spécifiques à l'application en dehors de MQSeries Adapter Offering afin :
   - d'activer l'adaptateur source pour l'acquisition des données d'application de l'application source
   - d'activer l'application cible pour l'acquisition de données de messages de l'adaptateur cible

   La nature exacte de l'interface spécifique à l'application dépend des caractéristiques de l'application source et de l'application cible. Vous trouverez ci-dessous quelques exemples d'interfaces spécifiques aux applications :
   - Appels et routines utilisateurs API
   - Lectures et écritures de fichiers
   - Déclencheurs de bases de données
   - Files d'attente de messages

4. Configurez le noyau pour la prise en charge de l'exécution : expédition, routage, traçage et transmission de messages. Reportez-vous à la section «Configuration du noyau» pour plus d'informations sur la configuration du noyau.
5. Configurez MQSeries ou d'autres logiciels de communication en mode message et, éventuellement, MQSeries Integrator pour la prise en charge de l'ensemble de l'architecture de votre système. Reportez-vous à la section «Configuration de MQSeries et MQSeries Integrator» à la page 96.
6. Si nécessaire, créez des catégories de connexions Java pour la prise en charge de la transmission des messages. Elles sont spécifiques à chaque application cible. Elles ne sont nécessaires que si l'adaptateur cible nécessite des informations pour l'ouverture d'une session et la connexion à l'application.
7. Testez l'ensemble du système—c'est-à-dire MQSeries Adapter Kernel avec vos adaptateurs source et adaptateurs cible, vos interfaces spécifiques à l'application et votre code personnalisé—avant de mettre le système en service.
8. Déployez le système dans l'environnement de production.
9. Activez le noyau en démarrant un ou plusieurs démons d'adaptateur et, éventuellement les serveurs de traçage. Assurez-vous que l'application source est démarrée. Si l'adaptateur source est exécuté dans le traitement de l'application source, l'adaptateur source est démarré automatiquement avec l'application source ; aucune autre opération n'est nécessaire pour démarrer l'adaptateur source. Tout démon ou serveur contenant l'adaptateur source doit être démarré. Reportez-vous à la section «Démarrage du noyau» à la page 97.

Configuration du noyau

Cette section décrit la configuration du noyau pour une utilisation dans votre environnement. La section «Généralités sur la configuration» à la page 63 offre une présentation conceptuelle de la configuration du noyau. La section «Fichiers impliqués dans le démarrage et la configuration» à la page 68 décrit les divers fichiers qui, ensemble, définissent une configuration MQSeries Adapter Kernel. La section «Le fichier d'installation» à la page 68 décrit le fichier `aqmsetup`, qui définit plusieurs paramètres initiaux du noyau. La section «Le fichier de configuration» à la page 69 décrit le fichier `aqmconfig.xml`, qui fournit des informations de configuration spécifiques au noyau, comme les noms des applications source et cible, des adaptateurs source et cible, des files d'attente et gestionnaires de files d'attente, des modes de communication et des spécifications de journalisation et de traçage.

Généralités sur la configuration

Cette section offre une présentation conceptuelle de la configuration du noyau. Il est important de comprendre le déroulement de l'exécution du noyau avant de configurer celui-ci. Cette section décrit ce déroulement de façon simplifiée. Reportez-vous à la section «Déroulement de l'exécution» à la page 15 pour des informations plus détaillées.

Au niveau le plus élémentaire, la configuration de MQSeries Adapter Kernel est régie par les données qui circulent entre les applications. La configuration doit également ternir compte des facteurs suivants :

- les applications qui reçoivent les données ;
- les adaptateurs requis côté source, ainsi que les adaptateurs cible, les démons d'adaptateurs et les agents requis côté cible ;
- les modes de communications, les formats de classement et les mécanismes d'entraînement utilisés.

Les structures et le format des données diffèrent pour chaque application de la configuration. Par exemple, si une configuration comprend deux applications, A et B, qui envoient chacune des données de commande à l'application C, il est probable que les données de l'application A soient d'un format différent et possèdent des codes de significations différentes des données de l'application B. Pour éviter que l'application C ait à reconnaître et à analyser les deux flots de données différents provenant des deux applications distinctes, les données de chacune sont converties en un *message d'intégration* dans un *format d'intégration neutre*. En principe, ce format correspond à une norme de l'industrie basée sur la spécification XML. Ainsi, le seul format de données que l'application C doit être capable de reconnaître et d'analyser est ce format d'intégration neutre.

La figure figure 3 à la page 64 représente la circulation de données entre les applications A et B et les applications C et D. Elle est suivie d'une explication des divers flots de données.

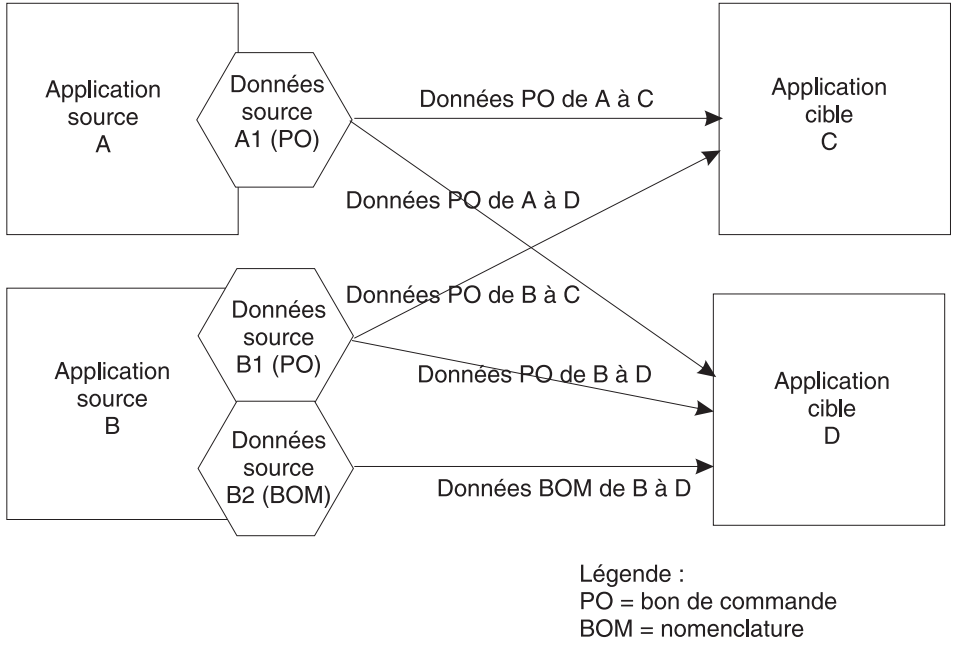


Figure 3. Applications connectées par des flots de données dans une configuration simple

Dans la figure figure 3, les données relatives aux bons de commande provenant des applications A et B circulent en direction des applications C et D, , tandis que les données relatives à la nomenclature provenant de l'application B circulent uniquement vers l'application D. Dans chaque cas, les données sont converties dans un format standard avant d'être envoyées aux applications cible. Les données relatives aux bons de commande provenant des applications A et B sont converties dans un format XML standard représentant ce type de données. Les données relatives à la nomenclature provenant de l'application B sont converties dans un format XML standard représentant ce type de données.

Un transfert de communications comme MQSeries ou une implémentation du service JMS (Java Message Service) sert à envoyer les données à l'application ou aux applications cible. Le message d'intégration est converti dans le *format de classement* requis par le transfert de communications particulier, puis communiqué à ce transfert (par exemple, une file d'attente MQSeries). Chaque application cible peut utiliser un transfert de communications et un format de classement différents pour recevoir les messages. Ainsi, l'application C peut utiliser MQSeries et l'application D, JMS, comme indiqué à la figure figure 4 à la page 65. Dans ce cas, tous les messages d'intégration destinés à l'application C (c'est-à-dire, les données relatives aux bons de commande provenant des applications A et B) sont converties dans un format de classement MQSeries, tandis que tous les messages d'intégration acheminés

vers l'application D (c'est-à-dire, les données relatives aux bons de commande provenant des applications A et B, ainsi que les données relatives à la nomenclature provenant de l'application B) sont converties dans un format de classement JMS. MQSeries Adapter Kernel utilise les mécanismes suivants pour effectuer ces conversions :

- Un *adaptateur source* permet de convertir les données applicatives en messages d'intégration. Ces adaptateurs sont créés dans MQSeries Adapter Builder.
- L'*adaptateur natif* permet de convertir le message d'intégration en *message de communication*. Cet adaptateur utilise le *service de messages logiques* (LMS) pour convertir le message en vue de son acheminement par le transfert de communications ; ce service LMS est propre au transfert de communications utilisé. Le service LMS utilise alors un *programme de formatage* pour classer le message sur le transfert.

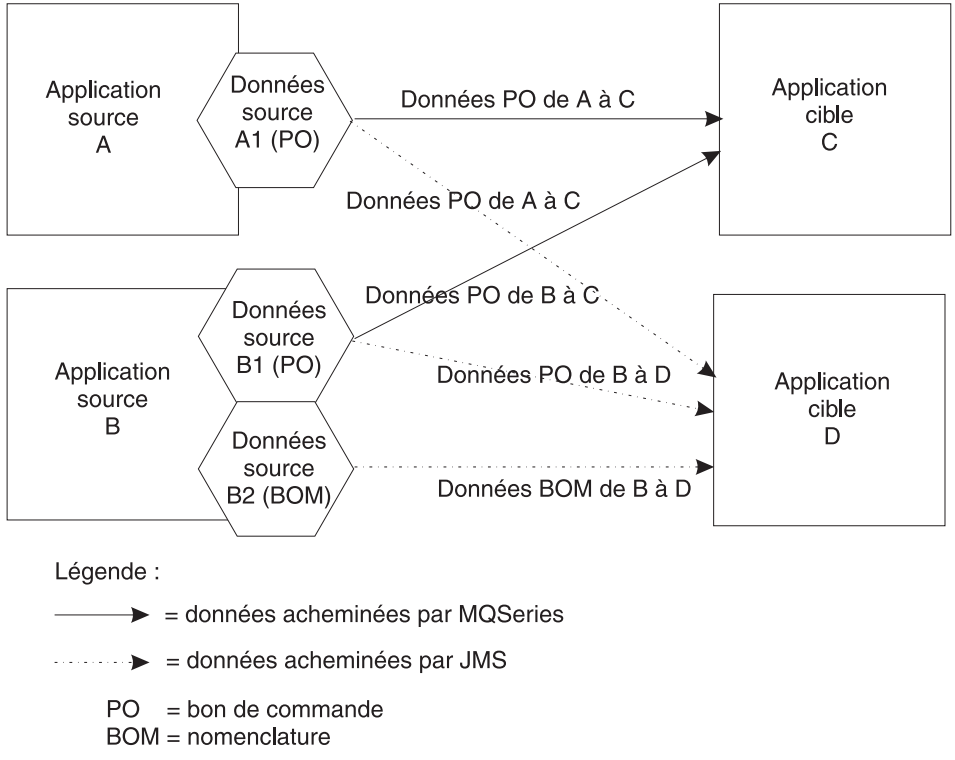


Figure 4. Applications connectées par des transferts de communications différents dans une configuration simple

La circulation des données entre l'application, le message d'intégration et le message de communication est illustrée par les figures figure 5 à la page 66 et figure 6 à la page 66. Lorsqu'un message de communication est reçu sur la

cible, les conversions sont inversées : l'adaptateur natif reconvertit le message de communication en message d'intégration ; puis, si nécessaire, l'adaptateur cible convertit le message d'intégration dans le format de données exigé par l'application cible.

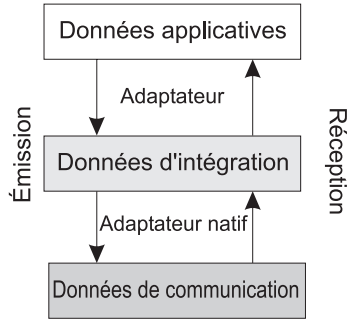


Figure 5. Conversion des données

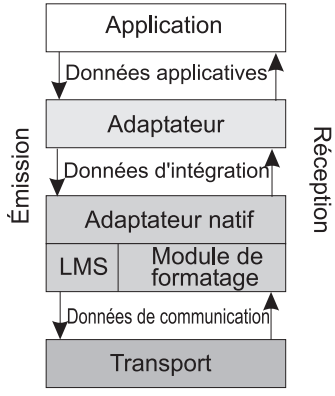


Figure 6. Circulation des données

Chaque étape franchie par les données doit être représentée dans le fichier de configuration du noyau. Il existe trois configurations logiques requises, réparties par identificateur d'application (source ou cible). Un identificateur d'application peut désigner soit une application source, soit une application cible, selon que les messages sont destinés à l'application (cible) ou en sont issus (source). Le fichier de configuration doit inclure les informations suivantes :

- Communications :
  - destination des données ;
  - transfert de communications à utiliser ;
  - méthode de classement des communications (programme de formatage) à utiliser ;

- conditions sous-jacentes requises pour les communications, tels qu'un gestionnaire de files d'attente MQSeries et des noms de files d'attente MQSeries ou un générateur de connexion de files d'attente JMS et des noms de files d'attente JMS.

- Adaptateurs (uniquement côté cible) :
  - adaptateurs requis pour le traitement des données ;
  - type d'adaptateurs utilisés (EAB ou EJB) ;
  - informations supplémentaires propres au type d'adaptateur utilisé ;
  - pour la version autonome de MQSeries Adapter Kernel, informations relatives au démon et à l'agent de l'adaptateur.
- Autres :
  - spécifications de traçage ;
  - spécifications de journalisation.

La circulation des données par rapport aux différentes parties de la configuration est représenté à la figure figure 7.

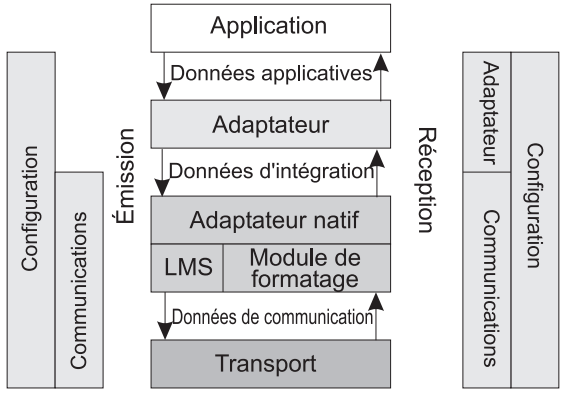


Figure 7. Circulation des données liée à la configuration

Reportez-vous à la section «Syntaxe et organisation du fichier de configuration» à la page 70 pour savoir comment mapper ces conditions de configuration avec des éléments XML dans le fichier `aqmconfig.xml`, qui gère ces aspects de la configuration du noyau. La section «Configurations courantes» à la page 82 répertorie plusieurs configurations courantes.

Fichiers impliqués dans le démarrage et la configuration

La configuration du noyau est déterminée par plusieurs fichiers personnalisables. Au moyen d'un éditeur de texte standard, éditez les fichiers de configuration du noyau pour votre site. Les fichiers suivants sont associés à la configuration du noyau :

- Le fichier `aqmsetenv.bat` (UNIX) ou `aqmsetenv.sh` (systèmes Windows), qui sert à paramétrer les variables d'environnement. Editez ce fichier pour changer les variables d'environnement du système après installation, si nécessaire. Les variables d'environnement paramétrées par ce fichier comprennent PATH, CLASSPATH, et LIBPATH. Ces variables sont paramétrées automatiquement par le programme d'installation sur les systèmes Windows. Pour paramétrer ces variables automatiquement lorsque vous vous connectez à UNIX, ajoutez les valeurs spécifiées dans le fichier `aqmsetenv.sh` à votre fichier `.profile` (si vous utilisez l'interpréteur de commandes Bourne ou Korn) ou à votre fichier `.cshrc` (si vous utilisez l'interpréteur de commandes C).

  Pour plus d'informations sur le paramétrage des variables d'environnement appropriées sur OS/400, reportez-vous à l'étape 6 à la page 46.
- Le fichier `aqmsetup`, qui fournit plusieurs valeurs de paramétrage initiales pour le noyau. Reportez-vous à la section «Le fichier d'installation» pour plus d'informations.
- Le fichier `aqmconfig.xml`, qui configure le noyau. Reportez-vous à la section «Le fichier de configuration» à la page 69 pour plus d'informations. Ce fichier contient la plupart des valeurs qui configurent le noyau.
- Le fichier `aqmcreateq.bat` (systèmes Windows) ou `aqmcreateq.sh` (UNIX et OS/400), qui est un script permettant de créer des files d'attente MQSeries. Pour plus d'informations, reportez-vous à la section «Création des files d'attente MQSeries» à la page 102.

Tous ces fichiers comprennent des commentaires pouvant vous aider à les éditer.

Il est recommandé de sauvegarder ces fichiers. Pour plus d'informations, reportez-vous à la section «Maintenance du noyau» à la page 99.

Le fichier d'installation

Le fichier de configuration, `aqmsetup`, commande plusieurs paramétrages initiaux du noyau, y compris :

- L'emplacement du fichier de configuration. Reportez-vous à la section «Le fichier de configuration» à la page 69.
- L'emplacement des DTD XML, s'il s'agit d'un répertoire autre que le répertoire courant.

- Les variables d'environnement JNI (Java Native Interface) pour l'interface C, pour modifier l'espace mémoire utilisé. Ceci s'applique lorsqu'un module C exécutable démarre un procédé et qu'une machine vituelle Java est instanciée par ce procédé. L'utilisation de la mémoire peut être commandée dans ce cas en supprimant les commentaires et en modifiant les lignes suivantes dans le fichier `aqmsetup` :

  ```
  #AQM_JNI_NATIVESTACKSIZE=1048576
  #AQM_JNI_JAVASTACKSIZE=4194304
  #AQM_JNI_MINHEAPSIZE=16777216
  #AQM_JNI_MAXHEAPSIZE=268435426
  ```

  Toutes les tailles sont indiquées en octets.

Vous trouverez un exemple de fichier `aqmsetup` dans «Annexe E. Modèle de fichier d'installation» à la page 129, ainsi que dans le répertoire `samples` de l'installation de MQSeries Adapter Kernel.

Si nécessaire, éditez le fichier d'installation lors de la première installation de MQSeries Adapter Kernel. Après l'installation, n'éditez le fichier que si le noyau rencontre un problème de mémoire Java insuffisante, comme indiqué dans la liste précédente.

Le fichier de configuration

Cette section traite du fichier `aqmconfig.xml`, qui détermine la configuration du noyau. La «Syntaxe et organisation du fichier de configuration» à la page 70 donne des informations sur la structure du fichier de configuration. Vous trouverez dans la section «Edition du fichier de configuration» à la page 91 des suggestions sur la meilleure façon d'éditer le fichier de configuration.

La configuration de MQSeries Adapter Kernel est déterminée par un fichier XML appelé `aqmconfig.xml`. Un modèle de fichier de configuration se trouve dans «Annexe D. Modèle de fichier de configuration» à la page 123 et est également inclus dans le répertoire `samples` de l'installation de MQSeries Adapter Kernel.

Les valeurs indiquées dans le fichier de configuration commandent les éléments suivants du noyau :

- Identificateurs logiques source
- Identificateurs logiques destination
- Informations sur les démons de l'adaptateur et les agents côté cible
- Clients trace
- Serveurs trace

- Classement et routage des messages, déterminés par les spécifications suivantes :
  - Noms des files d'attente de réception, des files d'attente d'erreur et des files d'attente de réponses
  - Une ou plusieurs destinations par défaut auxquelles les messages doivent être envoyés
  - Nom du gestionnaire de files d'attente MQSeries ou du générateur de connexion de file d'attente JMS qui reçoit ou transmet le message
  - Dépassement du délai imparti pour la réception des messages ou des réponses
  - Classe de l'adaptateur cible du côté cible du noyau qui traite chaque message
  - Informations supplémentaires propres à l'adaptateur cible
  - Nombre minimal d'agents côté cible (en cas d'exécution d'une version autonome de MQSeries Adapter Kernel)
  - Activation et désactivation de la trace et contrôle du niveau de la trace
  - Activation et désactivation du journal de suivi
- Mode de communication

Syntaxe et organisation du fichier de configuration

La configuration de MQSeries Adapter Kernel est basée sur le Lightweight Directory Access Protocol (LDAP), la structure du fichier de configuration crée une image-miroir du LDAP. L'élément XML de niveau supérieur, `Epic`, représente le niveau supérieur du répertoire LDAP et les objets secondaires du LDAP sont représentés par des éléments XML imbriqués dans l'élément de niveau supérieur. Certains éléments XML comportent les attributs requis représentant les informations LDAP. Les valeurs sont ajoutées à la configuration soit comme contenu des éléments, soit comme attributs des éléments. Un exemple d'une valeur de configuration attribuée comme contenu d'un élément est `<epictracelevel>-1</epictracelevel>`, qui attribue la valeur `-1` (tous les messages possibles) à l'élément `epictracelevel`. Un exemple d'une valeur de configuration attribuée comme attribut d'un élément est `<ePICTraceHandler epictracehandler="com.ibm.logging.ConsoleHandler">`, qui attribue la catégorie `com.ibm.logging.ConsoleHandler` à utiliser comme gestionnaire de trace.

Vous trouverez ci-dessous une liste et une description des éléments de niveau supérieur utilisés dans le fichier de configuration. La section «Eléments XML utilisés dans le fichier de configuration» à la page 73 indique et décrit l'ensemble des éléments utilisés dans le fichier de configuration. Reportez-vous à l'exemple de fichier de configuration pour des exemples de l'utilisation des différents éléments dans le contexte.

- `Epic`—L'élément de niveau supérieur nécessaire pour le fichier `aqmconfig.xml`.
- `ePICApplications`—L'enfant nécessaire de l'élément.
- `ePICApplication`—L'enfant nécessaire de l'élément `ePICApplications`. Il indique et définit les applications que le noyau doit servir; un élément complètement défini `ePICApplication` (comprenant des éléments enfants) est nécessaire pour chaque application.
- `AdapterRouting`—Enfant en option de l'élément `ePICApplication`. Il définit le gestionnaire de files d'attente et les informations correspondantes.
- `ePICBodyCategory`—L'enfant nécessaire de l'élémennt `AdapterRouting`. Il définit la catégorie de corps pour les messages devant être routés par le noyau.
- `ePICBodyType`—L'enfant nécessaire de l'élément `ePICBodyCategory`. Il définit le type de corps des messages devant être routés par le noyau. Il contient des définitions pour des éléments tels que les destinations des messages, les modes de communication pour la réception des messages et les modules de formatage des messages.
- `ePICAdapterDaemonExtensions`—Enfant en option de l'élément `ePICApplication` représentant un démon d'adaptateur. Il contient des informations concernant les démons des adaptateurs, y compris des identificateurs d'applications et le nombre des agents d'adaptateurs.
- `ePICTraceExtensions`—Un enfant en option de l'élément `ePICApplication` représentant une application client trace ou un élément de serveur de trace. Il définit des informations liées au traçage.

La figure 8 à la page 72 montre la structure de niveau supérieur du fichier de configuration. Il ne s'agit pas d'un exemple d'exécution d'un fichier de configuration ; il est simplement destiné à démontrer les relations et les interdépendances entre les éléments de niveau supérieur. Pour un exemple complet, reportez-vous à la section «Annexe D. Modèle de fichier de configuration» à la page 123.

```
<?xml version="1.0" encoding="UTF-8"?>
<Epic o="ePIC">
  <ePICApplications o="ePICApplications">
    <!-- The following <ePICApplication> tag configures the kernel to work with
    an application named APP1. -->
    <ePICApplication epicappid="APP1">
      <!-- Tags here specify logging and trace information for the APP1
      application. -->
      <AdapterRouting cn="epicadapterrouting">
        <!-- Tags here specify the queue manager and its attributes. -->
        <ePICBodyCategory epicbodycategory="DEFAULT">
          <ePICBodyType epicbodytype="DEFAULT">
            <!-- Tags here specify the details of transporting and processing messages
             from APP1. -->
          </epicBodyType>
        </ePICBodyCategory>
      </AdapterRouting>
    </ePICApplication>
    <!-- The following <ePICApplication> tag starts an adapter daemon for the
    APP1 application. -->
    <ePICApplication epicappid="APP1Daemon">
      <!-- Specifications for the APP1Daemon adapter daemon, which works with
      the APP1 application. -->
      <ePICAdapterDaemonExtensions cn="epicappextensions">
        <epicdepappid>APP1</epicdepappid>
        <epicminworkers>1</epicminworkers>
      </ePICAdapterDaemonExtensions>
    </ePICApplication>
    <!-- The following <ePICApplication> tag configures the kernel to work with
    an application named APP2. -->
    <ePICApplication epicappid="APP2">
      <!-- Tags here specify logging and trace information for the APP2
      application. -->
      <AdapterRouting cn="epicadapterrouting">
        <!-- Tags here specify the queue manager and its attributes. -->
        <ePICBodyCategory epicbodycategory="DEFAULT">
          <ePICBodyType epicbodytype="DEFAULT">
            <!-- Tags here specify the details of transporting and processing messages
            from APP2. -->
          </epicBodyType>
        </ePICBodyCategory>
      </AdapterRouting>
    </ePICApplication>
    <!-- The following <ePICApplication> tag configures a trace client named
    TraceClient. -->
    <ePICApplication epicappid="TraceClient">
      <ePICTraceExtensions cn="epicappextensions">
        <!-- Tags here specify attributes of the trace client. -->
      </ePICTraceExtensions>
    </ePICApplication>
  </ePICApplications>
</Epic>
```

Figure 8. Structure du fichier de configuration

Vous trouverez ci-dessous une liste et une description de l'ensemble des éléments utilisés dans le fichier de configuration. Si l'un des éléments comporte une valeur par défaut, le noyau utilise cette valeur si un élément de la configuration exige une valeur qui n'est pas spécifiée explicitement.

Eléments XML utilisés dans le fichier de configuration

Epic Elément de niveau supérieur pour le fichier de configuration.

Eléments enfants :

- `context`
- `ePICApplications` (exigé)

Attributs : `o="ePIC"` (exigé)

contexte
Spécifie la racine du contexte du système du fichier JNDI (Java Naming and Directory Interface) (FSContext) lorsque des objets JMS (Java Message Service) sont utilisés. Le répertoire par défaut est le répertoire courant. Nécessaire si JMS est utilisé. Reportez-vous à la section «Utilisation de l'enregistrement en mémoire des objets JMS» à la page 110 pour plus d'informations concernant l'utilisation d'objets JMS avec MQSeries Adapter Kernel.

Eléments enfants : Néant

Attributs : Néant

ePICApplications
Contient des informations concernant les applications prises en charge par le noyau.

Eléments enfants : `ePICApplication` (exigé)

Attributs : `o="ePICApplications"` (exigé)

ePICApplication
Donne des informations concernant une application prise en charge par le noyau.

Eléments enfants :

- `epiclogging`
- `epictrace`
- `epictracelevel`
- `epictraceclientid`
- `epiclogoninfoclassname`
- `AdapterRouting`
- `ePICTraceExtensions`
- `ePICAdapterDaemonExtensions`

Attributs : epicappid="*application_ID*", où *application_ID* est un identificateur d'application valide (exigé)

epiclogging
Détermine s'il y a lieu d'exécuter le journal de suivi. Pour le journal de suivi, le produit WebSphere Business Integrator doit être utilisé. La sélection par défaut est `false`.

Eléments enfants : Néant

Attributs : Néant

epictrace
Détermine s'il y a lieu d'utiliser le traçage. La sélection par défaut est `false`.

Eléments enfants : Néant

Attributs : Néant

epictracelevel
Définit le niveau de traçage à l'aide des constantes spécifiées par la classe`com.ibm.logging.IRecordType`. La valeur par défaut est `0` (pas de message). Reportez-vous au *Problem Determination Guide* pour plus de détails sur le traçage et pour obtenir une liste complète des niveaux de trace valides.

Eléments enfants : Néant

Attributs : Néant

epictraceclientid
Spécifie le nom de l'application client trace. La sélection par défaut est `TraceClient`.

Eléments enfants : Néant

Attributs : Néant

epiclogoninfoclassname
Spécifie le nom de la classe de connexion utilisée pour se connecter à l'application en cas d'utilisation d'un adaptateur EAB. La sélection par défaut est `com.ibm.epic.adapters.eak.adapterdaemon.EpicLogonDefault`.

Eléments enfants : Néant

Attributs : Néant

AdapterRouting
Contient des informations sur les types de messages et le routage des messages.

Eléments enfants :

- epicmqppqueuemgr
- epicuseremotequeuemanagertosend
- epicmqppqueuemgrhostname
- epicmqppqueuemgrportnumber
- epicmqppqueuemgrchannelname
- epicjmsconnectionfactoryname
- ePICBodyCategory (required)

Attributs : cn="epicadapterrouting" (exigé)

epicmqppqueuemgr
Si MQSeries est utilisé comme mécanisme d'entraînement, spécifie le nom du gestionnaire de files d'attente à utiliser. Si celui-ci n'est pas spécifié ou s'il est spécifié sous la forme DEFAULT, c'est le gestionnaire de files d'attente par défaut qui est utilisé.

Eléments enfants : Néant

Attributs : Néant

epicuseremotequeuemanagertosend
Si MQSeries est utilisé comme mécanisme d'entraînement, indique s'il y a lieu d'utiliser un gestionnaires de files d'attente à distance pour envoyer les messages. La sélection par défaut est false.

Eléments enfants : Néant

Attributs : Néant

epicmqppqueuemgrhostname
Si MQSeries est utilisé comme mécanisme d'entraînement, spécifie le nom d'hôte TCP/IP de l'ordinateur sur lequel le gestionnaire de files d'attente réside. Nécessaire uniquement en cas d'utilisation de MQSeries Client.

Eléments enfants : Néant

Attributs : Néant

epicmqppqueuemgrportnumber
Si MQSeries est utilisé comme mécanisme d'entraînement, spécifie le numéro de port du processus serveur du gestionnaire de files d'attente. La sélection par défaut est 1414 (le paramètre par défaut de MQSeries). Nécessaire uniquement en cas d'utilisation de MQSeries Client.

Eléments enfants : Néant

Attributs : Néant

epicmqppqueuemgrchannelname
Si MQSeries est utilisé comme mécanisme d'entraînement, spécifie le nom de canal du serveur du gestionnaire de files d'attente. Nécessaire uniquement en cas d'utilisation de MQSeries Client.

Eléments enfants : Néant

Attributs : Néant

epicjmsconnectionfactoryname
Si JMS est utilisé comme mécanisme d'entraînement, spécifie le nom de fabrication de la Connexion JMS. La valeur doit être spécifiée sous la forme *attribut=objet*, où l'*attribut* est l'attribut LDAP et l'*objet* est l'objet Connexion JMS. L'objet doit être enregistré sous l'élément `AdapterRouting`. Par exemple, pour un objet connexion JMS désigné `QCFTEST1` avec un attribut LDAP de `cn`, la valeur spécifiée par cet élément est `cn=QCFTEST1`.

Eléments enfants : Néant

Attributs : Néant

ePICBodyCategory
Spécifie la catégorie de corps des messages envoyés.

Eléments enfants : `ePICBodyType` (exigé)

Attributs : `epicbodycategory=`*`body_category`*, où *body_category* spécifie la catégorie de corps des messages envoyés (exigé)

ePICBodyType
Spécifie le type de corps des messages envoyés.

Eléments enfants :

- `epiccommandclassname`
- `epiccommandtype`
- `epiccommandejbmapper`
- `epiccommandejbmethod`
- `epiccommandejbmethodparmtype`
- `epiccommandejburl`
- `epiccommandejbinitialcontext`
- `epicdestids`
- `epicreceivemode`
- `epicmessageformatter`
- `epicreceivetimeout`

- epicreceivemqpqueue
- epicerrormqpqueue
- epicreplymqpqueue
- epicjmsreceivequeuename
- epicjmserrorqueuename
- epicreceivefiledir
- epiccommitfiledir
- epicerrorfiledir

Attributs : epicbodytype=*body_type*, où *body_type* spécifie le type de corps des messages envoyés (exigé)

epiccommandclassname
Spécifie le nom d'un adaptateur cible EAB ou d'une commande EJB invoquée pour traiter les messages. Nécessaire en cas d'utilisation d'un démon d'adaptateur ou de WebSphere Application Server pour la réception des messages.

Eléments enfants : Néant

Attributs : Néant

epiccommandtype
Spécifie le type de l'adaptateur cible. Les valeurs qui peuvent être utilisées sont MQAKEAB et MQAKEJB. MQAKEAB spécifie un adaptateur cible EAB MQSeries Adapter Kernel standard ; MQAKEJB spécifie que les beans d'entreprise sont utilisés côté cible du noyau dans WebSphere Application Server. La sélection par défaut est MQAKEAB. La valeur MQAKEJB est requise lorsque l'adaptateur cible est un bean d'entreprise.

Eléments enfants : Néant

Attributs : Néant

epiccommandejbmapper
Spécifie le nom de la classe TDCMapper utilisée pour mapper les données d'entrée. La sélection par défaut est TDCGenericMapper. Nécessaire lorsque l'adaptateur cible est un bean d'entreprise.

Eléments enfants : Néant

Attributs : Néant

epiccommandejbmethod
Spécifie le nom de la méthode à appeler sur un bean d'entreprise. La méthode doit accepter un objet `TerminalDataContainer` en entrée et renvoyer un objet `TerminalDataContainer`. La sélection par défaut est `execute`. Nécessaire lorsque l'adaptateur cible est un bean d'entreprise.

Eléments enfants : Néant

Attributs : Néant

epiccommandejbmethodparmtype
Spécifie le nom de classe de l'objet qui est utilisé comme paramètre pour la méthode appelée sur le bean d'entreprise. La sélection par défaut est le nom de classe de l'objet renvoyé par `TDCMapper`. Nécessaire lorsque l'adaptateur cible est un bean d'entreprise.

Eléments enfants : Néant

Attributs : Néant

epiccommandejburl
Spécifie l'URL (adresse universelle) d'un bean d'entreprise déployé, sous la formr `IIOP://`*`nom_hôte`*`:`*`port`*, où *nom_hôte* correspond au nom d'hôte du serveur EJB et *port* désigne le port sur lequel le serveur de noms est en mode écoute (par défaut, `900`). La sélection par défaut est `IIOP:///`. Nécessaire lorsque l'adaptateur cible est un bean d'entreprise.

Eléments enfants : Néant

Attributs : Néant

epiccommandejbinitialcontext
Spécifie le nom du générateur de contexte initial utilisé pour rechercher le nom principal (home) du bean d'entreprise. La sélection par défaut est `com.ibm.ejs.ns.jndi.CNInitialContextFactory`. Nécessaire lorsque l'adaptateur cible est un bean d'entreprise.

Eléments enfants : Néant

Attributs : Néant

epicdestids
Spécifie les identificateurs d'une ou de plusieurs applications à utiliser comme destinations des messages. Nécessaire si l'application envoie des messages et si l'ID logique de destination est paramétré à `NONE`.

Eléments enfants : Néant

Attributs : Néant

epicreceivemode
Spécifie le mode de communication à utiliser. Reportez-vous à «Annexe A. Modes de communication» à la page 107 pour obtenir une liste et une explication des modes de communication valides. Nécessaire si l'application reçoit des messages.

Eléments enfants : Néant

Attributs : Néant

epicmessageformatter
Spécifie le module de formatage de messages à utiliser en fonction de la valeur indiquée pour `epicreceivemode` et de la méthode d'entraînement employée. Reportez-vous à tableau 10 à la page 109 et tableau 11 à la page 109 pour plus de détails sur les modules de formatage de messages et les méthodes d'entraînement.

Eléments enfants : Néant

Attributs : Néant

epicreceivetimeout
Spécifie, en millisecondes, le laps de temps durant lequel le récepteur reste en attente des messages avant de se mettre en phase de temporisation. La valeur par défaut est `0`. Une valeur de `-1` correspond à l'absence de temporisation.

Eléments enfants : Néant

Attributs : Néant

epicreceivemqppqueue
Spécifie le nom de la file d'attente à partir de laquelle les messages seront reçus. Nécessaire lorsque l'élément `epicreceivemode` spécifie un mode de communication MQSeries. Reportez-vous à «Annexe A. Modes de communication» à la page 107 pour obtenir la liste des modes de communication MQSeries.

Eléments enfants : Néant

Attributs : Néant

epicerrormqppqueue
Spécifie le nom de la file d'attente dans laquelle placer les messages d'erreur. Nécessaire si la fonction de mise en file d'attente des messages d'erreur est utilisée et si l'élément `epicreceivemode` spécifie un mode de communication MQSeries. Reportez-vous à «Annexe A. Modes de communication» à la page 107 pour obtenir la liste des modes de communication MQSeries.

Eléments enfants : Néant

Attributs : Néant

epicreplymqppqueue
Spécifie le nom de la file d'attente à partir de laquelle les messages de réponse seront reçus. Nécessaire si les demandes de réponse sont utilisées et si l'élément epicreceivemode spécifie un mode de communication MQSeries. Reportez-vous à «Annexe A. Modes de communication» à la page 107 pour obtenir la liste des modes de communication MQSeries.

Eléments enfants : Néant

Attributs : Néant

epicjmsreceivequeuename
Spécifie le nom de la file d'attente à partir de laquelle les messages seront reçus. Nécessaire pour le mode de communication JMS. L'objet doit être enregistré sous l'élément ePICBodyType. La valeur doit être spécifiée sous la forme *attribut=objet*, où l'*attribut* est l'attribut LDAP et l'*objet* est le nom de l'objet file d'attente JMS. Par exemple, pour un objet JMS désigné TEST1AIQ avec un attribut LDAP de cn, la valeur spécifiée par cet élément est la suivante : cn=TEST1AIQ.

Eléments enfants : Néant

Attributs : Néant

epicjmserrorqueuename
Spécifie le nom de la file d'attente dans laquelle placer les messages d'erreur. Nécessaire si la fonction de mise en file d'attente des messages d'erreur est utilisée avec le mode de communication JMS. L'objet doit être enregistré sous l'élément ePICBodyType. La valeur doit être spécifiée sous la forme *attribut=objet*, où l'*attribut* est l'attribut LDAP et l'*objet* est le nom de l'objet file d'attente JMS. Par exemple, pour un objet JMS désigné TEST1AEQ avec un attribut LDAP de cn, la valeur spécifiée par cet élément est la suivante : cn=TEST1AEQ.

Eléments enfants : Néant

Attributs : Néant

epicjmsreplyqueuename
Spécifie le nom de la file d'attente à partir de laquelle les messages de réponse seront reçus. Nécessaire si les demandes de réponse sont utilisées avec le mode de communication JMS. L'objet doit être enregistré sous l'élément ePICBodyType. La valeur doit être spécifiée sous la forme *attribut=objet*, où l'*attribut* est l'attribut LDAP et l'*objet* est le nom de l'objet file d'attente JMS. Par exemple, pour un objet JMS désigné TEST1RPL avec un attribut LDAP de cn, la valeur spécifiée par cet élément estcn=TEST1RPL.

Eléments enfants : Néant

Attributs : Néant

epicreceivefiledir
Spécifie le nom du répertoire à partir duquel les messages seront reçus. Nécessaire pour le mode de communication FILE.

Eléments enfants : Néant

Attributs : Néant

epiccommitfiledir
Spécifie le nom du répertoire dans lequel placer en attente les messages reçus jusqu'à leur affectation définitive. Nécessaire pour le mode de communication FILE lors de la réception des messages.

Eléments enfants : Néant

Attributs : Néant

epicerrorfiledir
Spécifie le nom du répertoire dans lequel placer les messages d'erreur. Nécessaire si la fonction de mise en file d'attente des messages d'erreur est utilisée avec le mode de communication FILE.

Eléments enfants : Néant

Attributs : Néant

ePICAdapterDaemonExtensions
Contient des informations sur les extensions de démon d'adaptateur.

Eléments enfants :

- `epicdepappid`
- `epicminworkers`

Attributs : `cn="epicappextensions"` (exigé)

ePICTraceExtensions
Contient des informations sur les extensions de trace. Reportez-vous à *Problem Determination Guide* pour une description complète de cet élément et de ses enfants.

Eléments enfants :

- `epicdepappid`
- `epictracesyncoperation`
- `epictracemessagefile`
- `epictracehandler`
- `ePICTraceHandler`

Attributs : `cn="epicappextensions"` (exigé)

epicdepappid

Spécifie l'identificateur de l'application que le démon de l'adaptateur prend en charge. Il adopte par défaut l'ID de l'application avec lequel le démon de l'adaptateur a été démarré.

Eléments enfants : Néant

Attributs : Néant

epicminworkers

Spécifie le nombre d'agents d'adaptateurs démarrés par le démon de l'adaptateur. La valeur par défaut est 1.

Eléments enfants : Néant

Attributs : Néant

Configurations courantes

Cette section répertorie les valeurs de configuration pour plusieurs scénarios de configuration courants, notamment les valeurs permettant l'envoi et la réception des messages selon divers transferts de communications. A l'envoi d'un message, les valeurs de configuration sont obtenues à la fois du côté source et du côté cible ; à la réception, ces valeurs sont obtenues uniquement du côté cible. La source et la cible sont représentées par leurs identificateurs logiques respectifs. Dans ces exemples, on considère que la source et la cible se trouvent sur deux machines différentes. Si l'identificateur d'application cible n'est pas encore défini, il est déterminé à partir de la valeur de l'élément `epicdestids` dans la configuration de la source.

Remarque : Les scénarios de configuration recensent les valeurs de configuration des éléments qui sont applicables et peuvent être définies. Pour connaître les valeurs par défaut de chaque élément, reportez-vous au paragraphe correspondant de la section «Eléments XML utilisés dans le fichier de configuration» à la page 73.

Configurations courantes MQSeries : Cette section présente les configurations courantes lorsque MQSeries est utilisé comme transfert de communications. L'élément `epicreceivemode` spécifie un mode de communication MQSeries (par exemple, MQPP or MQRFH2). Les scénarios suivants sont cités :

- tableau 2 à la page 83 indique les éléments de configuration qu'il convient de définir pour l'envoi d'un message d'un serveur MQSeries à un autre serveur MQSeries.
- tableau 3 à la page 83 indique les éléments de configuration à définir pour l'envoi d'un message d'un serveur MQSeries utilisant un gestionnaire de files d'attente éloigné, vers un autre serveur MQSeries.

- tableau 4 à la page 84 indique les éléments de configuration à définir pour l'envoi d'un message d'un client MQSeries utilisant un serveur hôte, vers un serveur MQSeries.
- tableau 5 à la page 85 indique les éléments de configuration à définir pour la réception d'un message sur un serveur MQSeries.
- tableau 6 à la page 86 indique les éléments de configuration à définir pour la réception d'un message sur un client MQSeries utilisant un serveur hôte.

Tableau 2. Configuration courante : Envoi d'un message d'un serveur MQSeries à un autre serveur MQSeries

Configuration source	**Configuration cible**
	L'élément `epicreceivemode` spécifie un mode de communication MQSeries.
L'élément `epicmqppqueuemgr` spécifie le nom du gestionnaire de files d'attente. Ce gestionnaire doit exister sur la machine de l'application source.	
	L'élément `epicreceivemqppqueue` spécifie le nom de la file d'attente de réception. Cette file doit être une file éloignée MQSeries sur la machine de l'application cible ou doit faire partie d'une grappe MQSeries.
L'élément `epicreplymqppqueue` spécifie le nom de la file d'attente de réponse. Cette file doit être une file locale MQSeries sur la machine de l'émetteur ou doit faire partie d'une grappe MQSeries. N'est utilisé que pour les demandes et réponses synchrones.	
	L'élément `epicmessageformatter` spécifie le nom du module de formatage à utiliser.
L'élément `epicreceivetimeout` spécifie pendant combien de temps le récepteur attend une réponse avant de se mettre en phase de temporisation.	

Tableau 3. Configuration courante : Envoi d'un message d'un serveur MQSeries à un autre serveur MQSeries via un gestionnaire de files d'attente éloigné

Configuration source	**Configuration cible**
	L'élément `epicreceivemode` spécifie un mode de communication MQSeries.

Tableau 3. Configuration courante : Envoi d'un message d'un serveur MQSeries à un autre serveur MQSeries via un gestionnaire de files d'attente éloigné (suite)

Configuration source	Configuration cible
L'élément `epicmqpqueuemgr` spécifie le nom du gestionnaire de files d'attente. Ce gestionnaire doit exister sur la machine de l'application source.	L'élément `epicmqpqueuemgr` spécifie le nom du gestionnaire de files d'attente. Ce gestionnaire doit exister sur la machine de l'application cible. Son nom doit être spécifié ; il est impossible d'utiliser une valeur par défaut.
L'élément `epicremotequeuemanagertosend` spécifie qu'un gestionnaire de files d'attente à distance est utilisé pour envoyer les messages.	
	L'élément `epicreceivemqppqueue` spécifie le nom de la file d'attente de réception. Cette file doit être une file locale MQSeries sur la machine de l'application cible ou doit faire partie d'une grappe MQSeries.
L'élément `epicreplymqppqueue` spécifie le nom de la file d'attente de réponse. Cette file doit être une file locale MQSeries sur la machine de l'émetteur ou doit faire partie d'une grappe MQSeries. N'est utilisé que pour les demandes et réponses synchrones.	
	L'élément `epicmessageformatter` spécifie le nom du module de formatage à utiliser.
L'élément `epicreceivetimeout` spécifie pendant combien de temps le récepteur attend une réponse avant de se mettre en phase de temporisation.	

Tableau 4. Configuration courante : Envoi d'un message d'un client MQSeries utilisant un serveur hôte, vers un serveur MQSeries.

Configuration source	Configuration cible
	L'élément `epicreceivemode` spécifie un mode de communication MQSeries.
L'élément `epicmqpqueuemgr` spécifie le nom du gestionnaire de files d'attente. Ce gestionnaire doit exister sur la machine hôte du client émetteur.	

Tableau 4. Configuration courante : Envoi d'un message d'un client MQSeries utilisant un serveur hôte, vers un serveur MQSeries. (suite)

Configuration source	Configuration cible
L'élément `epicmqppqueuemgrhostname` spécifie le nom d'hôte de la machine serveur MQSeries.	
L'élément `epicmqppqueuemgrportnumber` spécifie le numéro de port du processus serveur du gestionnaire de files d'attente sur la machine serveur.	
L'élément `epicmqppqueuemgrchannelnumber` spécifie le numéro de canal du serveur du gestionnaire de files d'attente.	
	L'élément `epicreceivemqppqueue` spécifie le nom de la file d'attente de réception. Cette file doit être une file éloignée MQSeries sur la machine de l'application cible ou doit faire partie d'une grappe MQSeries.
L'élément `epicreplymqppqueue` spécifie le nom de la file d'attente de réponse. Cette file doit être une file locale MQSeries sur la machine hôte du client émetteur ou doit faire partie d'une grappe MQSeries. N'est utilisé que pour les demandes et réponses synchrones.	
	L'élément `epicmessageformatter` spécifie le nom du module de formatage à utiliser.
L'élément `epicreceivetimeout` spécifie pendant combien de temps le récepteur attend une réponse avant de se mettre en phase de temporisation.	

Tableau 5. Configuration courante : Serveur MQSeries recevant un message

Configuration source	Configuration cible
Non applicable.	L'élément `epicreceivemode` spécifie un mode de communication MQSeries.
	L'élément `epicmqppqueuemgr` spécifie le nom du gestionnaire de files d'attente. Ce gestionnaire doit exister sur la machine de l'application cible.

Tableau 5. Configuration courante : Serveur MQSeries recevant un message (suite)

Configuration source	Configuration cible
	L'élément `epicreceivemqpqueue` spécifie le nom de la file d'attente de réception. Cette file doit être une file locale MQSeries sur la machine de la cible.
	L'élément `epicerrormqpqueue` spécifie le nom de la file d'attente des erreurs. Cette file doit être une file locale MQSeries sur la machine de la cible ou doit faire partie d'une grappe. Nécessaire uniquement en cas d'utilisation d'un agent d'adaptateur.
	L'élément `epicmessageformatter` spécifie le nom du module de formatage à utiliser.
	L'élément `epicreceivetimeout` spécifie pendant combien de temps le récepteur attend de recevoir un message avant de se mettre en phase de temporisation.

Tableau 6. Configuration courante : Client MQSeries utilisant un serveur hôte recevant un message.

Configuration source	Configuration cible
Non applicable.	L'élément `epicreceivemode` spécifie un mode de communication MQSeries.
	L'élément `epicmqpqueuemgr` spécifie le nom du gestionnaire de files d'attente. Ce gestionnaire doit exister sur la machine hôte client du récepteur.
	L'élément `epicmqpqueuemgrhostname` spécifie le nom d'hôte de la machine serveur MQSeries.
	L'élément `epicmqpqueuemgrportnumber` spécifie le numéro de port du processus serveur du processus de files d'attente sur la machine serveur.
	L'élément `epicmqpqueuemgrchannelnumber` spécifie le numéro de canal du serveur du gestionnaire de files d'attente.

Tableau 6. Configuration courante : Client MQSeries utilisant un serveur hôte recevant un message. (suite)

Configuration source	Configuration cible
	L'élément `epicreceivemqppqueue` spécifie le nom de la file d'attente de réception. Cette file doit être une file locale MQSeries sur la machine hôte client du récepteur.
	L'élément `epicerrormqppqueue` spécifie le nom de la file d'attente des erreurs. Cette file doit être une file locale MQSeries sur la machine hôte client du récepteur ou doit faire partie d'une grappe. Nécessaire uniquement en cas d'utilisation d'un agent d'adaptateur.
	L'élément `epicmessageformatter` spécifie le nom du module de formatage à utiliser.
	L'élément `epicreceivetimeout` spécifie pendant combien de temps le récepteur attend de recevoir un message avant de se mettre en phase de temporisation.

Configuration courantes JMS : Cette section présente les configurations courantes lorsque JMS est utilisé comme transfert de communications. L'élément `epicreceivemode` spécifie JMS.

Si l'implémentation JMS de MQSeries est utilisée, les objets MQSeries appropriés doivent exister. Par exemple, un générateur de connexion de files d'attente JMS doit être associée à un gestionnaire de files d'attente sur un serveur MQSeries et une file d'attente JMS doit être associée à une file MQSeries. Il n'est pas nécessaire que les objets MQSeries soient répertoriés dans la configuration, mais ils doivent exister.

Les scénarios suivants sont cités :

- tableau 7 indique les éléments de configuration à définir pour l'envoi d'un message via JMS.
- tableau 8 à la page 88 indique les éléments de configuration à définir pour la réception d'un message via JMS.

Tableau 7. Configuration courante : Envoi d'un message via JMS

Configuration source	Configuration cible
	L'élément `epicreceivemode` spécifie le mode de communication JMS.

Tableau 7. Configuration courante : Envoi d'un message via JMS (suite)

Configuration source	Configuration cible
L'élément `epicjmsconnectionfactoryname` spécifie le nom du générateur de connexion de file d'attente JMS. L'objet de référence doit exister dans la configuration.	
	L'élément `epicjmsreceivequeuename` spécifie le nom de la file d'attente de réception JMS. L'objet de référence doit exister dans la configuration.
L'élément `epicjmsreplyqueuename` spécifie le nom de la file d'attente de réponse JMS. L'objet de référence doit exister dans la configuration. N'est utilisé que pour les demandes et réponses synchrones.	
	L'élément `epicmessageformatter` spécifie le nom du module de formatage à utiliser.
L'élément `epicreceivetimeout` spécifie pendant combien de temps le récepteur attend une réponse avant de se mettre en phase de temporisation.	

Tableau 8. Configuration courante : Réception d'un message via JMS

Configuration source	Configuration cible
Non applicable.	L'élément `epicreceivemode` spécifie le mode de communication JMS.
	L'élément `epicjmsconnectionfactoryname` spécifie le nom du générateur de connexion de file d'attente JMS. L'objet de référence doit exister dans la configuration.
	L'élément `epicjmsreceivequeuename` spécifie le nom de la file d'attente de réception JMS. L'objet de référence doit exister dans la configuration.
	L'élément `epicjmserrorqueuename` spécifie le nom de la file d'attente des erreurs JMS. L'objet de référence doit exister dans la configuration.
	L'élément `epicmessageformatter` spécifie le nom du module de formatage à utiliser.

Tableau 8. Configuration courante : Réception d'un message via JMS (suite)

Configuration source	Configuration cible
	L'élément `epicreceivetimeout` spécifie pendant combien de temps le récepteur attend de recevoir un message avant de se mettre en phase de temporisation.

Configuration courantes avec adaptateur : Cette section présente les configurations courantes lorsqu'un adaptateur est appelé côté cible. Des valeurs de configuration différentes sont utilisées selon que la cible est un adaptateur cible EAB (Enterprise Access Builder) (spécifié par une valeur `epiccommandtype` égale à `MQAKEAB`) ou un adaptateur cible de bean de session de service EJB (spécifié par une valeur `epiccommandtype` égale à `MQAKEJB`).

Remarque : Les adaptateurs cible de bean de session de service EJB ne sont pris en charge que dans WebSphere Application Server.

Pour une valeur `epiccommandtype` égale à `MQAKEAB`, spécifiez les valeurs des éléments complémentaires suivants :

- `epiclogoninfoclassname`
- `epiccommandclassname`

Pour une valeur `epiccommandtype` égale à `MQAKEJB`, spécifiez les valeurs des éléments complémentaires suivants :

- `epiccommandclassname`
- `epiccommandejbmethod`
- `epiccommandejbmethodparmtype`
- `epiccommandejburl`
- `epiccommandejbinitialcontext`
- `epiccommandejbmapper`

Ajout des informations de l'adaptateur à la configuration

Lorsqu'un nouvel adaptateur est ajouté à la configuration du noyau, plusieurs spécifications au moins doivent être ajoutées au fichier de configuration. Le fichier de configuration `aqmconfig.minimum.xml` est un exemple de fichier de configuration minimum. Ce fichier est représenté dans la section «Annexe D. Modèle de fichier de configuration» à la page 123 et également dans le répertoire `samples` de l'installation de MQSeries Adapter Kernel.

Les spécifications suivantes représentent les informations minimales qui doivent être ajoutées à la configuration lors de l'ajout d'un nouvel adaptateur :

- **Adaptateur source** (envoi des messages) :
  - L'identificateur de l'application sous laquelle l'adaptateur source fonctionne.
  - Le gestionnaire de files d'attente par défaut. Si MQSeries est utilisé comme mécanisme de transfert, est installé et fonctionne sur le même ordinateur que l'adaptateur source, il n'est pas nécessaire d'effectuer une configuration spécifique du gestionnaire de files d'attente.
  - Les identificateurs logiques de destination des messages. Si tous les messages sont adressés à la même destination, utilisez une catégorie de corps DEFAULT et un type de corps DEFAULT.
  - Une file d'attente de réception pour chaque identificateur logique de destination auquel l'adaptateur source envoie les messages.
- **Adaptateur cible** (réception des messages):
  - L'identificateur de l'application sous laquelle l'adaptateur cible fonctionne.
  - Le gestionnaire de files d'attente par défaut. Si MQSeries est utilisé comme mécanisme de transfert, est installé et fonctionne sur le même ordinateur que l'adaptateur source, il n'est pas nécessaire d'effectuer une configuration spécifique du gestionnaire de files d'attente.
  - Le mode de réception pour MQSeries. Il est généralement identique pour tous les messages ; dans ce cas, utilisez une catégorie de corps DEFAULT et un type de corps DEFAULT.
  - La file d'attente de réception pour MQSeries. Si elle est identique pour tous les messages, utilisez une catégorie de corps DEFAULT et un type de corps DEFAULT.
  - La file d'attente des erreurs, au cas où une erreur se produirait lors du traitement du message par l'adaptateur cible. Elle est généralement identique pour tous les messages ; dans ce cas, utilisez une catégorie de corps DEFAULT et un type de corps DEFAULT.
  - Le nom de classe de l'adaptateur cible à invoquer lors de la réception d'un message. Il est spécifique à la catégorie de corps et au type de corps.
  - La valeur de la temporisation de réception. Il est recommandé de définir une valeur appropriée pour empêcher une utilisation intensive de l'unité centrale (UC). Il est généralement identique pour tous les messages ; dans ce cas, utilisez une catégorie de corps DEFAULT et un type de corps DEFAULT.

  Pour les adaptateurs cibles supplémentaires, les mêmes informations peuvent suffire si la même file d'attente de réception est utilisée. Si c'est le

cas, la seule information qui doit être spécifiée en plus est le nom de classe de l'adaptateur cible à invoquer pour la catégorie et le type de corps spécifiques.

- **Spécifications de trace**:
  - L'état d'activation ou de désactivation de la trace.
  - Le niveau de trace.
  - Les spécifications de trace supplémentaires, y compris la destination de trace pour les adaptateurs source et les adaptateurs cible. La trace est affichée par défaut dans la fenêtre d'invite ou le terminal à partir duquel le noyau a été démarré.

Edition du fichier de configuration

Utilisez un éditeur de texte ou un éditeur dédié XML pour éditer le fichier de configuration. Un fichier DTD désigné `aqmconfig.dtd` est fourni dans le répertoire `samples` de l'installation du noyau pour les utilisateurs des éditeurs XML. Un éditeur XML appelé Xeena peut être téléchargé à partir du site Web alphaWorks IBM à l'adresse suivante : www.alphaworks.ibm.com. Les recommandations suivantes s'appliquent à l'édition du fichier de configuration :

- Avant de commencer l'édition du fichier de configuration, collectez toutes les informations pertinentes sur la configuration souhaitée. Il peut s'agir du nom des applications et des files d'attente impliquées dans la configuration, des types de messages échangés, du mode ou des modes de communication utilisés et des informations sur les programmes trace et d'autres extensions.
- Copiez le fichier modèle `aqmconfig.xml` à partir du répertoire `samples`à l'emplacement souhaité. Ne renommez pas la copie du fichier de configuration. Editez la copie.
- Utilisez les commentaires pour identifier les différentes sections du fichier de configuration et pour justifier les valeurs spécifiques utilisées dans votre configuration (par exemple,les identificateurs d'application, les noms des files d'attente des messages et les valeurs de temporisation). En XML, les commentaires débutent par les caractères `<!--` et se terminent par les caractères `-->`. Les commentaires peuvent occuper plusieurs lignes comme dans l'exemple suivant :

  ```
  <!--
       Comment text
  -->
  ```

  Notez que XML ne permet pas d'insérer des commentaires au sein d'autres commentaires.
- Organisez le fichier de configuration selon les identificateurs d'application. Regroupez les entrées concernant chaque application.
- Si vous n'utilisez pas un éditeur XML dédié, utilisez un éditeur de texte qui conserve les fins de ligne et ne coupe pas les lignes lors de la sauvegarde

du fichier. Les éditeurs Notepad sur les systèmes Windows et vi ou Emacs sur UNIX sont des exemples de ce type d'éditeurs de texte.

- Notez bien que XML différencie les majuscules des minuscules ; veillez tout particulièrement à respecter la casse correcte pour tous les noms et attributs de codes (éléments). L'utilisation d'une casse incorrecte dans les codes peut invalider le fichier de configuration. L'utilisation d'un éditeur XML peut permettre d'éviter les erreurs de casse.
- Si vous souhaitez utiliser les valeurs par défaut pour la catégorie de corps et le type de corps et si ces valeurs par défaut ne sont pas encore définies, vous devez configurer la valeur DEFAULT pour chaque valeur dans le fichier de configuration. Si vous n'effectuez pas cette opération, le noyau n'utilise pas les valeurs par défaut.
- Validez le fichier de configuration avant de le mettre en service. Pour plus d'informations, reportez-vous à la section «Validation du fichier de configuration».
- Les modifications apportées au fichier de configuration s'appliqueront dès le lancement du traitement du noyau suivant. Si un traitement est exécuté lors de la modification du fichier de configuration, le traitement doit être interrompu, puis relancé, de manière à ce que les modifications puissent être prises en compte. Procédez avec précautions lors de l'édition d'un fichier de configuration en cours de production.
- Sauvegardez le fichier de configuration à chaque fois que vous l'éditez.

Validation du fichier de configuration

Après l'édition du fichier de configuration et avant sa mise en service, il est recommandé de le valider. Pour cela, effectuez les opérations générales suivantes :

1. Créez un répertoire de validation du fichier de configuration dans lequel valider et configurer le test.
2. Créez un message XML de validation.
3. Configurez les files d'attente de messages de manière à ce qu'elles puissent prendre en charge le test de validation.
4. Configurez puis exécutez un test de validation du fichier de configuration pour l'envoi d'un message et la réception d'un message.
5. Examinez les résultats du test pour déterminer si le fichier de configuration est correct.

L'utilitaire permettant de créer un message XML de validation et le test de validation du fichier de configuration sont tous les deux fournis avec le noyau.

Le test de validation du fichier de configuration invoque la méthode `sendMsg` et envoie un message XML de validation d'un adaptateur natif sur le côté source du noyau à un démon d'adaptateur sur le côté cible du noyau. Un

adaptateur source et un adaptateur cible ne sont pas nécessaires. Toutefois, si un adaptateur cible est en place, vous pouvez également tester l'envoi du message à l'application cible.

La procédure est décrite ci-dessous :

Remarque : Plusieurs scripts sont fournis dans le but de faciliter la procédure de l'utilisateur. Si vous le souhaitez, copiez les scripts puis éditez les copies pour créer vos propres versions. Si vous utilisez OS/400, notez que les versions UNIX des scripts peuvent être exécutées dans une session **qsh**. Vous pouvez ouvrir une session **qsh** en entrant la commande Start QSH (**STRQSH**) au niveau d'une invite Control Language (CL).

Étape 1. Ouvrez une fenêtre d'invite.

Étape 2. Créez un répertoire de validation du fichier de configuration. Copiez le fichier de configuration et le fichier d'installation dans ce répertoire.

Étape 3. Passez au répertoire de validation.

Étape 4. Entrez la commande suivante pour créer le message XML de validation :

- `aqmcrtmsg.bat` (systèmes Windows)
- `aqmcrtmsg.sh` (UNIX et OS/400)

Étape 5. Une liste d'options s'affiche. Sélectionnez une option et appuyez sur Enter. Saisissez une valeur pour chaque option. L'ordre dans lequel les valeurs sont saisies n'a pas d'importance. `set sourcelogicalid`, `set msgtype` et `set bodycategory` sont des exemples d'options. Vous devez entrer des valeurs pour les options 20, 21, 22 et 23. Vous pouvez utiliser les options 24 ou 241 pour fournir les données de corps des messages. D'autres valeurs ne sont pas nécessaires.

Étape 6. Entrez l'option 1 pour créer le fichier XML de validation. Le fichier XML de validation est créé dans le répertoire courant et est désigné `EpicMessage`*`nn`*`.xml`, où *nn* correspond au numéro du fichier XML.

Étape 7. Entrez l'option 0 pour quitter l'utilitaire de validation.

Étape 8. Définissez les files d'attente de messages appropriées pour la prise en charge de la validation.

Étape 9. Définissez la variable d'environnement AQMSETUPFILE de manière à ce qu'elle pointe temporairement sur le fichier d'installation dans le répertoire de validation :

- Au niveau de l'invite sur les systèmes Windows, entrez ce qui suit :

  ```
  set AQMSETUPFILE=E:\fichiers_exécution\aqmsetup
  ```

où `E:\` représente le lecteur correct et *fichiers_exécution*le répertoire de validation.

- Sur UNIX et OS/400, entrez la commande suivante. Cet exemple de commande suppose l'utilisation de l'interpréteur de commandes Korn ; si vous utilisez un autre interpréteur de commandes, modifiez la commande en conséquence.

  ```
  export
  AQMSETUPFILE=répertoire_principal/fichiers_exécution/aqmsetup
  ```

  où *répertoire_principal* est le répertoire d'installation du noyau et *fichiers_exécution* le répertoire de validation. Sur OS/400, le fichier `aqmsetup` doit toujours se trouver dans votre répertoire personnel IFS (`/home/user_name`).

Si nécessaire, éditez le fichier d'installation dans le répertoire de validation de manière à ce qu'il pointe sur le fichier de configuration en cours de validation.

Étape 10. Choisissez parmi les éléments suivants celui que vous souhaitez tester :

- Uniquement le côté source du noyau.
- La possibilité de router le message jusqu'à l'application cible. Pour ce test, un adaptateur cible doit déjà être en place.
- Le traçage.

Testez d'abord le côté source, puis testez le côté cible. Désactivez le démon de l'adaptateur de manière à ne tester que le côté source. Activez le démon de l'adaptateur de manière à tester également le côté cible. Si un adaptateur cible n'est pas déjà en place, vous pouvez toujours tester si le démon d'adaptateur traite le message jusqu'au stadeoù il tente l'invocation de la commande pour l'adaptateur cible approprié. Il est recommandé d'activer le traçage, notamment si un adaptateur cible n'est pas déjà en place.

Étape 11. Exécutez le test de validation. A partir de n'importe quel répertoire, entrez la commande suivante :

- Sur les systèmes Windows :

  ```
  aqmsndmsg.bat -a source_logical_identifier -f XML_message_file
  ```

- Sur UNIX et OS/400 :

  ```
  aqmsndmsg.sh -a source_logical_identifier -f XML_message_file
  ```

  où :

  source_logical_identifier
  indique l'identificateur logique source. Cette valeur doit correspondre à la valeur de l'identificateur logique source entrée pour l'option 20 dans l'opération 5 à la page 93.

XML_message_file
indique le fichier de messages XML.

Remarque : La liste de toutes les options pour ce test peut être affichée en entrant la commande suivante :

Sur les systèmes Windows :

```
aqmsndmsg.bat -?
```

Sur UNIX et OS/400 :

```
aqmsndmsg.sh -?
```

Notez que le `-?` fonctionne uniquement sur l'interpréteur de commandes Korn ; si vous utilisez un autre interpréteur de commandes UNIX (comme l'interpréteur Bourne ou C), utilisez une barre oblique inversée avant le point d'interrogation (c'est-à-dire `-\?`).

Étape 12. Examinez les résultats. Le message de validation contient la catégorie de corps, le type de corps et les données correctes.

- Si vous ne testez que le côté source du noyau (c'est-à-dire si le démon de l'adaptateur n'a pas été démarré), examinez la file d'attente dans laquelle le message a dû être acheminé.
  - Si vous voyez votre message de validation dans cette file d'attente, ces entrées dans le fichier de configuration sont validées.
  - Si vous ne voyez pas votre message de validation dans cette file d'attente, consultez le fichier des écarts. Si la fonction de traçage est activée, vérifiez les messages de trace.
- Si vous testez le côté cible du noyau et si un adaptateur cible est en place, vérifiez l'application cible.
  - Si votre message de validation est reçu par l'application cible, ces entrées dans le fichier de configuration sont validées.
  - Si votre message de validation n'est pas reçu par l'application cible, vérifiez le fichier des écarts. Si la fonction de traçage est activée, vérifiez les messages de trace.
- Si vous testez le côté cible du noyau et si aucun adaptateur cible n'est en place, recherchez le message de validation dans la file d'attente des erreurs et recherchez un message d'erreur dans le fichier des écarts. Si la fonction de traçage est activée, vérifiez les messages de trace.

- Si vous voyez votre message de validation dans la file d'attente des erreurs et un message d'erreur, ces entrées dans le fichier de configuration sont validées.
- Si vous ne voyez pas votre message de validation dans la file d'attente des erreurs, vérifiez le fichier des écarts. Si la fonction de traçage est activée, vérifiez les messages de trace.

Étape 13. Si nécessaire, modifiez le fichier de configuration et validez-le de nouveau.

Configuration de MQSeries et MQSeries Integrator

Configurez MQSeries et le logiciel optionnel comme MQSeries Integrator de manière à ce qu'ils supportent le noyau comme suit :

Dans MQSeries :

- Plusieurs files d'attente sont utilisées pour vérifier l'installation. Si vous utilisez ces files d'attente pour vos environnements de test ou de production, libérez-les pour vérifier l'installation. Reportez-vous à la section «Procédure de vérification» à la page 50 pour connaître les files d'attente utilisées pour vérifier l'installation.
- Définissez les files d'attente pour la prise en charge du transfert des messages selon le programme de routage que vous avez défini.
- Lors de la création de files d'attente, paramétrez la variable d'environnement MAX_QUEUE_DEPTH à la hauteur maximale de file d'attente autorisée.

Dans MQSeries Integrator, configurez les files d'attente d'entrée et de sortie dans les règles (version 1.1) ou dans les flots de messages (version 2) qui correspondent aux files d'attente configurées dans le fichier de configuration.

Recommandations en matière de performances

Les recommandations suivantes concernant les performances s'appliquent exclusivement à MQSeries Adapter Kernel :

- Lors de l'analyse syntaxique des DTD XML, vérifiez que les fichiers DTD résident dans le même répertoire que le processus qui les analyse. Cette précaution permet de réduire les ressources nécessaires au processus pour localiser les DTD.
- Lors de l'envoi et de la réception de longs messages, l'utilisation du type de message RFH2 permet d'obtenir de meilleurs résultats que l'utilisation du type de message XML.

Consultez la documentation MQSeries pour connaître les recommandations générales relatives à l'amélioration des performances.

Démarrage du noyau

Pour démarrer le noyau, démarrez les éléments suivants :

- Le démon de l'adaptateur pour chaque application cible
- Le serveur trace (en option)

Notez que si l'adaptateur source est lancé dans le traitement de l'application source, l'adaptateur source est automatiquement lancé avec l'application source ; aucune autre opération n'est nécessaire pour démarrer l'adaptateur source. Tout démon ou serveur qui contient des adaptateurs source doit être démarré. Les adaptateurs source ne sont pas démarrés directement.

Démarrez chaque démon d'adaptateur et serveur trace en effectuant les opérations suivantes :

Remarque : Plusieurs scripts sont fournis dans le but de faciliter la procédure de l'utilisateur. Si vous le souhaitez, copiez les scripts puis éditez les copies pour créer vos propres versions. Si vous utilisez OS/400, notez que les versions UNIX des scripts peuvent être exécutées dans une session **qsh**. Vous pouvez ouvrir une session **qsh** en entrant la commande Start QSH (**STRQSH**) au niveau d'une invite Control Language (CL).

Étape 1. Démarrez MQSeries ou tout autre logiciel de messagerie ou logiciel optionnel comme MQSeries Integrator.

Étape 2. Lancez les logiciels associés requis par votre site— par exemple, les applications (externes au noyau) permettant la lecture des messages de trace dans les files d'attente.

Étape 3. Ouvrez une fenêtre d'invite. Pour chaque démon d'adaptateur, entrez la commande suivante :

- Sur les systèmes Windows :

  ```
  aqmstrad.bat -a application_identifier [-bc body_category
  -bt body_type] [-noretry]
  ```

- Sur UNIX et OS/400 :

  ```
  aqmstrad.sh -a application_identifier [-bc body_category
  -bt body_type] [-noretry]
  ```

  où

  -a *application_identifier*
  Identifie l'identificateur logique de destination que le démon de l'adaptateur prend en charge.

-bc *body_category*
Spécifie la catégorie de corps que l'agent du démon d'adaptateur utilise pour déterminer le mode de communication et les informations associées pour la réception des messages. En l'absence de toute valeur, le démon de l'adaptateur utilise la valeur DEFAULT.

-bt *body_type*
Spécifie le type de corps que l'agent du démon de l'adaptateur utilise pour déterminer le mode de communication et les informations associées pour la réception des messages. En l'absence de toute valeur, le démon de l'adaptateur utilise la valeur DEFAULT.

-noretry
Spécifie que l'agent cesse automatiquement le traitement lorsqu'il n'y a plus de messages. Si `-noretry` n'est pas spécifié, l'agent recherche en continu les messages dans la file d'attente et le démon de l'adaptateur doit être arrêté manuellement.

Remarque : Si vous devez modifier les paramètres de lancement de Java, éditez le fichier`aqmstrad.bat` (systèmes Windows) ou `aqmstrad.sh` (UNIX et OS/400). Consultez les commentaires contenus dans le fichier pour plus de détails.

Étape 4. Pour chaque serveur de trace, entrez la commande suivante :

- Sur les systèmes Windows :

  ```
  aqmstrtd.bat -how -a source_application_identifier
  ```

- Sur UNIX et OS/400 :

  ```
  aqmstrtd.sh -how -a source_application_identifier
  ```

  où :

  -how
  Indique la manière dont les messages de trace doivent être reçus. Les valeurs qui peuvent être utilisées sont les suivantes :
  - `socket`
  - `ena`, c'est-à-dire l'adaptateur natif

  -a *source_application_identifier*
  Identificateur d'application source. En l'absence de toute valeur, la valeur par défaut `TraceServer` indiquée dans le fichier de configuration file est utilisée.

Consultez le *Problem Determination Guide* pour plus d'informations sur les serveurs trace.

Étape 5. Après le démarrage d'un démon d'adaptateur ou d'un serveur trace, une fenêtre de traitement reste ouverte jusqu'à ce que vous arrêtiez le démon de l'adaptateur. La fenêtre de traitement peut afficher les erreurs. Pour plus d'informations, reportez-vous à la section «Messages d'erreur» à la page 100.

Arrêt du noyau

Pour arrêter le noyau, arrêtez chacun des démons d'adaptateur et des serveurs trace. Plusieurs méthodes permettent de les arrêter :

- Lors du démarrage du démon de l'adaptateur, définissez le paramètre `-noretry`. Reportez-vous à la section «Démarrage du noyau» à la page 97.
- Accédez à l'invite (systèmes Windows) ou au terminal (UNIX) à partir duquel le démon de l'adaptateur ou le serveur trace a été démarré et appuyez sur **Ctrl-C**. Effectuez cette opération pour chaque démon d'adaptateur ou serveur trace.
- Sur les systèmes Windows, vous pouvez utiliser le Gestionnaire de tâches pour arrêter le traitement.
- Sur UNIX, vous pouvez utiliser la commande **ps** pour déterminer le numéro du traitement, puis utiliser la commande **kill** pour arrêter le traitement.

Maintenance du noyau

Etablissez un plan de maintenance du noyau. Il est conseillé de sauvegarder périodiquement les éléments suivants :

- La configuration indiquée dans les fichiers suivants :
  - `aqmconfig.xml`
  - `aqmsetup`
- Les adaptateurs que vous avez créés ainsi que les fichiers associés

Il n'est pas nécessaire d'effectuer une sauvegarde ni une suppression périodique du contenu du fichier trace et des autres fichiers utilisés par le noyau pour la prise en charge de son propre traitement. Sauvegardez ces fichiers si nécessaire. Si les messages de trace sont acheminés vers un seul fichier au lieu de plusieurs fichiers, le fichier de trace unique peut devenir très gros. Si le niveau de traçage est défini de manière à capturer un niveau de détail très élevé (par exemple, tous les messages de trace ou messages d'information), il peut être nécessaire de supprimer les fichiers trace périodiquement.

Etablissement du diagnostic en cas de problème

Vous pouvez utiliser les messages d'erreur, les messages de trace et les files d'attente des erreurs MQSeries pour l'établissement du diagnostic en cas de problème. MQSeries Adapter Kernel génère des messages d'erreur et, si la fonction trace est activée, des messages de trace. Reportez-vous au *Problem Determination Guide* pour plus d'informations sur les méthodes d'établissement de diagnostic dans un environnement MQSeries Adapter Kernel.

Pour comprendre les messages d'erreur et les messages trace, vous devez comprendre comment fonctionne le noyau. Il utilise une file d'attente des erreurs pour traiter certaines erreurs. Pour plus d'informations, reportez-vous à la section «Fonctionnement du noyau» à la page 9.

Vous pouvez identifier le message à l'origine des messages d'erreur et des messages trace grâce à la combinaison de l'identificateur de message unique et de l'identificateur de transaction unique.

Aucun identificateur ne vous permet d'identifier de manière certaine le même message à la fois dans la file d'attente des erreurs et dans le noyau. Toutefois, vous pouvez associer manuellement un message figurant dans la file d'attente des erreurs et le message d'erreur, le message de trace correspondant ou les deux. Vous pouvez comparer un ou plusieurs éléments suivants entre eux :

- L'indication d'horodatage approximative
- La file d'attente pour l'identificateur logique source
- La file d'attente pour l'identificateur logique de destination
- La catégorie de corps
- Le type de corps
- L'identificateur de message unique
- L'identificateur de transaction unique

S'ils concordent, vous avez probablement associé le message de la file d'attente des erreurs au message d'erreur ou message trace correspondant.

Numéro de version

Exécutez `aqmversion.bat` (systèmes Windows) ou `aqmversion.sh` (UNIX et OS/400) dans le répertoire `bin` pour afficher le numéro de version du noyau.

Messages d'erreur

Le noyau génère les types de messages d'erreur suivants :

- L'adaptateur natif sur le côté source du noyau transmet les erreurs à l'adaptateur source. Consultez la documentation MQSeries Adapter Builder pour comprendre comment l'adaptateur source gère ces erreurs.

- L'adaptateur natif sur le côté cible du noyau transmet les erreurs à l'agent qui gère l'adaptateur natif.
- L'agent écrit les erreurs dans le fichier `EpicSystemExceptionFile`*`nnnnnnnn`*`.log`, qui réside dans le même répertoire que l'agent.
- Le démon de l'adaptateur écrit les messages d'erreur dans un fichier d'erreurs désigné `EpicSystemExceptionFile`*`nnnnnnnn`*`.log` qui réside dans le même répertoire que le démon de l'adaptateur. Etant donné que le démon de l'adaptateur et ses agents résident dans le même répertoire, ils enregistrent tous dans le même fichier d'erreurs. Le démon de l'adaptateur écrit des messages d'erreur dans la console (c'est-à-dire la ligne de commande ou le terminal utilisé pour le lancer, s'il a été lancé à partir d'une fenêtre).

Les messages d'erreur de trace du noyau sont différents des messages d'erreur de MQSeries. Vous trouverez ci-dessous un exemple de message d'erreur généré par le noyau :

```
2000.10.26 19:38:20.929 com.ibm.epic.adapters.eak.nativeadapter.LMSMQ
Thread Name=main receiveRequest(ENAService)  ePIC TEST2
TYPE_ERROR_EXC AQM5004: Received exception <com.ibm.epic.adapters.eak.common.
AdapterException> Message information: <AQM0114: com.ibm.epic.adapters.eak.
nativeadapter.MQNMRFH2Formatter::convertMessage(MQMessage): Expecting a message
with an MQHRF2 format and received a message with format <MQSTR   >.>
for <unmarshall Message()> having invalid data <(null)>
```

Les valeurs figurant dans un message d'erreur dépendent de la nature du message, qui peut renfermer les informations suivantes :

- L'indication d'horodatage
- L'identificateur logique source
- L'identificateur logique de destination
- La catégorie de corps
- Le type de corps
- L'identificateur de message unique
- L'identificateur de transaction unique
- Les informations d'erreur

Reportez-vous à la section «Problèmes de vérification courants» à la page 51 pour connaître les problèmes que vous pouvez rencontrer lors de la vérification de l'installation ainsi que les différentes solutions possibles à ces problèmes.

Messages de trace

Le noyau peut être configuré pour générer des messages de trace. Pour plus d'informations sur le traçage, reportez-vous au *Problem Determination Guide*.

Utilitaires

Création des files d'attente MQSeries

Vous pouvez utiliser des fichiers de commande ou des scripts d'interpréteur de commandes pour générer automatiquement la création des files d'attente MQSeries. Exécutez `aqmcreateq.bat` (systèmes Windows) ou `aqmcreateq.sh` (UNIX et OS/400), en utilisant le nom de l'application comme paramètre. Ces fichiers créent les files d'attente pour chaque application :

- La file d'attente de réception, désignée *application_name*AIQ.
- La file d'attente des erreurs, désignée *application_name*AEQ.
- La file d'attente de réponses, désignée *application_ name*RPL.

Chapitre 5. Utilisation des Interfaces de Programmation d'Applications (API) de MQSeries Adapter Kernel

Le noyau comprend des API qui sont utilisées pour des fonctions telles que l'expédition et la réception de messages, la création, l'analyse syntaxique XML et la gestion de la configuration du noyau. Ces API sont utilisées pour les adaptateurs créés à l'aide de MQSeries Adapter Builder. Le Centre d'aide et d'information de MQSeries Adapter Kernel comprend une documentation en ligne relative aux API au format Javadoc HTML.

Le noyau doit être utilisé avec des adaptateurs créés par l'utilisateur via l'outil MQSeries Adapter Builder. Le noyau ne doit pas être utilisé via des appels aux API de noyau uniquement à partir du code personnalisé. La documentation en ligne relative aux API est uniquement fournie en vue d'aider l'utilisateur à comprendre le mode de fonctionnement du noyau et à établir des diagnostics.

La documentation en ligne relative aux API du noyau se trouve dans le répertoire de la documentation.

Chapitre 6. Localisation des informations supplémentaires

Plusieurs sources d'informations peuvent être utiles pour l'utilisation de MQSeries Adapter Offering. Pour obtenir des informations supplémentaires sur MQSeries Adapter Kernel, reportez-vous au document *Problem Determination Guide*, accessible depuis le Centre d'aide et d'information MQSeries Adapter Kernel installé avec le produit. Le *Problem Determination Guide* fournit des informations sur la résolution des problèmes spécifiques susceptibles de se poser lors de l'utilisation du noyau. Pour obtenir des informations sur MQSeries Adapter Builder, consulter le Centre d'aide et d'information et le système d'aide en ligne de ce produit.

Disponibles sur l'Internet

L'adresse du site Web de la famille de produits MQSeries est www.ibm.com/software/ts/mqseries/. En suivant les liens proposés dans ce site Web, vous pouvez :

- Obtenir les dernières informations sur la famille de produits MQSeries, y compris MQSeries Adapter Offering.
- Accéder aux manuels MQSeries en format HTML et PDF, comprenant éventuellement une édition plus récente de ces manuels. Le lien direct avec la page de la bibliothèque MQSeries est www.ibm.com/software/ts/mqseries/library/manualsa/.
- Télécharger MQSeries SupportPacs.

Pour obtenir des informations sur l'utilisation de MQSeries sur OS/400, consultez la bibliothèque OS/400 à l'adresse www.ibm.com/servers/eserver/iseries/library/. Consultez également les manuels spécifiques OS/400– accessibles depuis le site Web de la bibliothèque MQSeries à l'adresse www.ibm.com/software/ts/mqseries/library/manualsa/.

Références

Les sites de référence suivants traitent des sujets étudiés dans ce document :

- Le site Web Open Applications Group à l'adresse suivante :www.openapplications.org/
- La recommandation 1.0 W3C relative au XML(Extensible Markup Language) à l'adresse suivante : www.w3.org/TR/1998/Rec-xml-19980210

Il ne s'agit pas de sites Web IBM.

Annexe A. Modes de communication

Cette annexe fournit des informations sur les modes de communication pris en charge par MQSeries Adapter Kernel et sur les classes Java utilisées pour leur prise en charge. Quelques-uns des modes de communication sont fournis comme modes grande diffusion avec des modules de formatage par défaut. Consultez tableau 10 à la page 109 pour savoir quels modules de formatage par défaut sont utilisés avec les modes grande diffusion.

Les modes de communication suivants sont pris en charge :

MQPP Le noyau transporte les messages à l'aide des services de base MQSeries. Il s'agit d'un mode grande diffusion.

MQRFH1 Le noyau transporte les messages en utilisant les messages brokers et MQSeries via MQSeries Integrator version 1.1. Il s'agit d'un mode grande diffusion.

MQRFH2 Le noyau transporte les messages en utilisant les messages brokers et MQSeries via MQSeries Integrator version 2. Il s'agit d'un mode grande diffusion.

MQBD Le noyau transporte les messages à l'aide des services de base MQSeries, mais n'envoie et ne reçoit que les données de corps. Il s'agit d'un mode grande diffusion. Les caractéristiques suivantes s'appliquent uniquement à ce mode :

- Il ne peut envoyer que les données de corps, pas les valeurs d'en-tête des messages.
- Il peut recevoir les messages qui ne contiennent que des données de corps. Il utilise les valeurs d'en-tête des messages par défaut suivantes pour les messages reçus :
  - `SourceLogicalApplicationID`—La valeur dans l'objet `ENAService` utilisée dans l'appel de méthode de réception.
  - `BodyCategory`—La valeur dans l'objet `ENAService` utilisée dans l'appel de méthode de réception.
  - `BodyType`—La valeur dans l'objet `ENAService` utilisée dans l'appel de méthode de réception.
  - `Acknowledgment`—Si le message MQ est une REQUETE (REQUEST) MQSeries, la valeur`Acknowledgment` est alors paramétrée à 1.
  - `BodyData`—Les données de messages reçues de MQSeries.

Toutes les autres valeurs d'en-tête utilisent les valeurs par défaut normales.

MQ Le noyau transporte les messages à l'aide des services de base MQSeries.

JMS Le noyau transporte les messages à l'aide de Java Message Service (JMS). Reportez-vous à «Utilisation de l'enregistrement en mémoire des objets JMS» à la page 110 pour obtenir des informations sur l'utilisation des objets JMS avec MQSeries Adapter Kernel.

FILE Le noyau place les messages dans un fichier et les extrait d'un fichier. Ce mode sert à l'établissement de diagnostics uniquement.

tableau 9 indique les modes de communication et les classes Java qui les prennent en charge. Toutes les classes Java sont issues du package Java `com.ibm.epic.adapters.eak.nativeadapter`. Notez que toute classe Java qui prend en charge le service de message logique (LMS) peut être désignée comme mode de communication ; dans ce cas, la classe elle-même est utilisée pour la prise en charge de la communication.

Tableau 9. Modes de communication et classes Java assurant leur prise en charge

Mode de communication	Classe Java	Notes
MQPP	`LMSMQBindingMQPP`	Nécessite l'installation de MQSeries
MQRFH1	`LMSMQBindingMQRFH1`	Nécessite l'installation de MQSeries
MQRFH2	`LMSMQBindingMQRFH2`	Nécessite l'installation de MQSeries
MQBD	`LMSMQMQBD`	Nécessite l'installation de MQSeries
MQ	`LMSMQBinding`	Nécessite l'installation de MQSeries
JMS	`LMSJMS`	Nécessite l'installation de JMS
FILE	`LMSFile`	Aucune

tableau 10 à la page 109 indique les modes de communication et leurs interfaces avec les modules de formatage associés. tableau 11 à la page 109 met en corrélation les interfaces des modules de formatage, les noms de classe des modules de formatage et leurs utilisations. Tous les modules de formatage sont issus du package Java `com.ibm.epic.adapters.eak.nativeadapter`. Notez que toute classe de module de formatage peut être désignée pour le mode de

communication ; dans ce cas, le module de formatage spécifié est utilisé comme module de formatage.

Tableau 10. Modes de communication et interfaces des modules de formatage

Mode de communication	Interface du module de formatage	Module de formatage par défaut
MQPP	MQFormatterInterface	MQNMXMLFormatter
MQRFH1	MQFormatterInterface	MQNMRFH1Formatter
MQRFH2	MQFormatterInterface	MQNMRFH2Formatter
MQBD	MQFormatterInterface	MQNMBDFormatter
MQ	MQFormatterInterface	MQNMXMLFormatter
JMS	JMSFormatterInterface	JMSNMRFH2Formatter
FILE	StringFormatterInterface	NMXMLFormatter

Tableau 11. Interfaces du module de formatage, noms de classe du module de formatage et objets

Interface du module de formatage	Nom de classe du module de formatage	Objet
MQFormatterInterface	MQNMXMLFormatter	EpicMessage en XML
	MQNMRFH1Formatter	EpicMessage en RFH1
	MQNMRFH2Formatter	EpicMessage en RFH2
	MQNMDBFormatter	Données de corps uniquement
JMSFormatterInterface	JMSNMXMLFormatter	EpicMessage en XML
	JMSNMRFH2Formatter	EpicMessage en RFH2
	JMSNMBDFormatter	Données de corps uniquement
StringFormatterInterface	NMXMLFormatter	EpicMessage en XML

tableau 12 indique les classes de LMS acceptées et leur degré de support transactionnel. Reportez-vous à «Capacités transactionnelles» à la page 28 pour obtenir des informations sur l'utilisation des transactions avec MQSeries Adapter Kernel.

Tableau 12. Classes LMS et support transactionnel

Classe LMS	Support transactionnel
LMSMQBindingMQPP	Monophase
LMSMQBindingMQRFH1	Monophase

Tableau 12. Classes LMS et support transactionnel (suite)

Classe LMS	Support transactionnel
LMSMQBindingMQRFH2	Monophase
LMSMQMQBD	Monophase
LMSMQBinding	Monophase
LMSJMS	Monophase
LMSFILE	Pas de support

Utilisation de l'enregistrement en mémoire des objets JMS

Les noms des objets JMS sont enregistrés en mémoire grâce à la mise en oeuvre du fichier FSContext de JNDI, qui fait partie de MQSeries JMS SupportPac. Le contexte (arborescence des répertoires) utilisé par le noyau pour FSContext respecte la hiérarchie LDAP en utilisant l'attribut distinctif avec la valeur associée pour le nom du répertoire. Par exemple, pour la hiérarchie LDAP `o=ePIC, o=ePICApplications, epicappid=TEST1`, l'arborescence des répertoires est la suivante : `o-ePIC/o-ePICApplications/epicappid-TEST1`.

Pour créer le contexte et les objets, utiliser l'outil JMS Admin livré avec les outils d'installation JMS. Les opérations de base définissent un contexte, puis modifient le contexte. La modification du contexte permet à l'utilisateur d'accéder au contexte. Créez les objets JMS à l'emplacement approprié. Des exemples de commandes permettant de créer la structure de contexte et les objets JMS figurent ci-après. Dans cet exemple, le nom de l'application est `TEST1`.

```
#
# aqmjmscreatesample.scp 1.00 09Mar01
# Used for MQSeries Adapter Kernel
# Sample AQM JMS Configuration.
#
# Copyright (c) 2001 International Business Machines. All Rights Reserved.
#
# This configuration file is as an example only.
#
# IBM MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF THIS
# SAMPLE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
# IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR
# PURPOSE, OR NON-INFRINGEMENT.
#
# CopyrightVersion 1.0
#
#
# This is a script to use with the JMS administration (JMSAdmin) tool
# which comes with MQSeries Support pac MA88.
# This tool requires the JMSAdmin.config to be set to use either
```

```
# FSCONTEXT (file) or LDAP. This script is setup to work with
# FSCONTEXT, but will work with LDAP with the following changes:
# - Change the "-" signs to "=".  Example: define ctx(o-ePIC)
#   becomes define ctx(o=ePIC)
# - In LDAP the contexts have to already be defined using the
#   LDAP administration tool.  For example you do not need
#   to "define ctx(o=ePIC) but only change into it with the
#   "change ctx(o-ePIC)" command.
# - There are some notes in the following script which highlight
#   differences when using LDAP.
#
#
# Example usage: MQSeries root\java\bin\jmsadmin.bat < aqmjmscreatesample.scp
#
# Some helpful commands:
# "display ctx" will display the context's of the context you are
# currently in.
# "=UP" means return to the parent context.  Example: change ctx(=UP)
# "=INIT" means return to root context. In this example one directory level
# above o-ePIC.  Example: change ctx(=INIT)
# "define xxx" is for creating either a context or object.
# "change xxx" is for changing/moving into the context.
#
# Always required.
define ctx(o-ePIC)
change ctx(o-ePIC)
# Always required.
define ctx(o-ePICApplications)
change ctx(o-ePICApplications)
# Application id is TEST1, requires a context.
define ctx(epicappid-TEST1)
change ctx(epicappid-TEST1)
# Always required.
define ctx(cn-epicadapterrouting)
change ctx(cn-epicadapterrouting)
# This will hold the JMS QueueConnectionFactory object.
# Note: These two steps are not required for LDAP.
define ctx(cn-QCFTEST1)
change ctx(cn-QCFTEST1)
# Create the JMS QueueConnectionFactory object whose name is QCFTEST1
# Using MQSeries in server (bindings) mode.
define qcf(QCFTEST1) qmgr(yourQManagerName) tran(BIND)
change ctx(=UP)
# BodyCategory is DEFAULT
define ctx(epicbodycategory-DEFAULT)
change ctx(epicbodycategory-DEFAULT)
# BodyType is DEFAULT
define ctx(epicbodytype-DEFAULT)
change ctx(epicbodytype-DEFAULT)
# This will hold the JMS Queue object whose name is TEST1AIQ.
# Note: These two steps are not required for LDAP.
define ctx(cn-TEST1AIQ)
change ctx(cn-TEST1AIQ)
# Create the JMS Queue object whose name is TEST1AIQ
# q(JMS Q Object Name) queue(MQSeries Queue name)
```

```
define q(TEST1AIQ) queue(TEST1AIQ)
# Can move up and define other contexts and JMS objects.
# Quit the administration tool.
end
```

Annexe B. Configurations validées

Il existe de nombreuses configurations et combinaisons possibles de MQSeries, MQSeries Adapter Offering et MQSeries Integrator. Chacun de ces membres de la famille de produits MQSeries est riche en fonctions et configurations. En outre, vous pouvez combiner des fonctionnalités de MQSeries, MQSeries Adapter Offering et MQSeries Integrator. Certaines fonctionnalités appartenant à un membre de la famille de produits MQSeries risquent de chevaucher partiellement des fonctionnalités offertes par d'autres membres de la famille. Vous devez déterminer comment utiliser et combiner les différentes fonctionnalités d'acheminement et de communication des messages de MQSeries, MQSeries Adapter Offering et MQSeries Integrator.

Les configurations suivantes de MQSeries, MQSeries Adapter Offering et MQSeries Integrator ont été validées à la date de la publication. Veuillez consulter le site Web MQSeries pour les dernières configurations validées.

MQSeries Adapter Kernel :

- Envoi d'un message avec demande d'accusé de réception et sans demande d'accusé de réception.
- Utilisation du mode de communication MQSeries ou JMS. Pour connaître les modes de communication valides, reportez-vous à la section «Annexe A. Modes de communication» à la page 107.
- Acheminement et communication des messages :
  - Envoi d'un message d'un adaptateur source à un adaptateur cible
  - Envoi d'un message d'un adaptateur source à des adaptateurs cible multiples
  - Communication des messages multifilières, c'est-à-dire à l'aide d'agents multiples
  - Avec identificateur logique cible défini à NONE dans le message, afin que le fichier de configuration du noyau serve à déterminer l'identificateur logique cible en fonction de la catégorie et du type de corps ainsi que de l'identificateur logique source.
  - Modèle de communication par insertion
  - Avec traçage activé

Remarque : Consultez «Annexe C. En-têtes de messages» à la page 115. Elle contient les zones d'en-tête de messages de MQSeries Adapter Kernel que le noyau remplit et traite.

- Avec les conditions préalables indiquées dans les sections «Matériel» à la page 33 et «Logiciel» à la page 34.
- Utilisation du fichier de configuration, non LDAP, pour définir la configuration.

MQSeries :

- Non utilisation des grappes MQSeries.

  Remarque : Consultez «Annexe C. En-têtes de messages» à la page 115. Elle contient les zones d'en-tête de messages de MQSeries Adapter Kernel que le noyau remplit et traite.

MQSeries Integrator :

- MQSeries Adapter Kernel et MQSeries peuvent acheminer et communiquer le message à MQSeries Integrator. Consultez les informations relatives à MQSeries Integrator pour déterminer ses capacités concernant le courtage de ces messages.
- Envoi de messages du côté source du noyau, via MQSeries et MQSeries Integrator version 2, et acheminement direct au côté cible du noyau. Au sein de MQSeries Integrator, le flux de messages est configuré pour l'acheminement statique. Tous les messages arrivant au noeud MQInput du flux sont directement acheminés vers une file d'attente MQOutput spécifique.

  Remarque : Consultez «Annexe C. En-têtes de messages» à la page 115. Elle contient les zones d'en-tête de messages de MQSeries Adapter Kernel que le noyau remplit et traite.

Annexe C. En-têtes de messages

MQSeries Adapter Offering utilise plusieurs en-têtes de messages. Reportez-vous à «Message et format de message» à la page 12 pour savoir quels en-têtes sont utilisés et dans quelles circonstances.

Cette annexe indique et décrit les zones d'en-tête de message.

En-tête descripteur de message MQSeries Adapter Kernel

Valeurs d'en-tête utilisées par MQSeries Adapter Kernel. Ces valeurs sont placées dans des objets collecteurs de messages. La colonne **Propagation dans les réponses ?** indique si la valeur est propagée ou non vers l'application source dans un message de réponse lorsque cette application demande une réponse. Certaines valeurs ne sont utilisées qu'avec WebSphere Business Integrator.

Tableau 13. En-tête MQSeries Adapter Kernel

Nom de l'en-tête	Propagation dans les réponses ?	Signification ou utilisation
UniqueID	Non	Identificateur unique pour chaque message.
TransactionID	Oui	Identificateur de transaction partagé par chaque message et sa réponse, le cas échéant. Equivalent à un PublicProcessID Extricity ou à un ApplicationID DataInterchange (DI).
MessageType	Non	Utilisé pour les messages de passerelle et de journal/de trace/d'erreur.
SourceLogicalID	Non	Identificateur logique de l'application source. Equivalent aux noms réservés dans DI, Partner Agreement Manager (PAM), et Business Flow Manager (BFM).

Tableau 13. En-tête MQSeries Adapter Kernel (suite)

Nom de l'en-tête	Propagation dans les réponses ?	Signification ou utilisation
DestinationLogicalID	Non	Identificateur logique de l'application cible. Pour DI et PAM, la valeur par défaut est `none`, mais peut être remplacée.
RespondToLogicalID	Oui	Identificateur logique auquel un message de réponse doit être envoyé. Copié dans DestinationLogicalID pour DI et dans SourceLogicalID pour PAM.
CorrelationID	Non	Usage réservé.
GroupStatus	Non	Usage réservé.
ProcessingCategory	Non	Equivalent à un PAM Public Process Identifier (identificateur de processus public) ou à un DI Command Process Identifier (identificateur de processus de commande).
QosPolicy	Non	Usage réservé.
DeliveryCategory	Non	Equivalent à un RequestorProfileID DI.
AckRequested	Non	Détermine si l'application source nécessite ou non un message de réponse.
PublicationTopic	Non	Usage réservé.
SessionID	Non	Equivalent à un BatchID DI.
EncryptionStatus	Non	Détermine le type de chiffrement et de signature du corps.
TimeStampCreated	Non	Date et heure à laquelle le message a été créé.
TimeStampExpired	Non	Date et heure après lesquelles le message perd sa signification. La valeur `-1` qu'il n'expire jamais.
Size	Non	Usage réservé.

Tableau 13. En-tête MQSeries Adapter Kernel (suite)

Nom de l'en-tête	Propagation dans les réponses ?	Signification ou utilisation
BodyType	Non	Représente le but spécifique du message.
BodyCategory	Non	Représente le type d'application du message.
BodySecondaryType	Non	Usage réservé.
UserArea	Non	Zone générale pour les données utilisateur.
RelatedSubjectID	Non	Utilisé pour la corrélation interprocessus.
ExternalID	Non	Identificateur du propriétaire actuel (par exemple, un utilisateur ou un partenaire d'échanges) en dehors de l'environnement d'application.
InternalID	Non	Identificateur du propriétaire actuel (par exemple, un utilisateur ou un partenaire d'échanges) à l'intérieur de l'environnement d'application.
BodySignature	Non	Usage réservé.
TransportCorrelationID	Oui	Usage réservé.

En-tête descripteur de message MQSeries

Le contenu des zones est déterminé par MQSeries. MQSeries Adapter Offering place les messages dans les files d'attente selon les indications des valeurs de contrôle des messages. Pour plus d'informations, reportez-vous au «Valeurs de contrôle de message» à la page 16.

Tableau 14. En-tête MQSeries

Section ou zone	Signification ou utilisation
Revision	Fixe.
UniqueID	Chaque message possède un identificateur unique.
TransactionID	Un message et sa réponse partagent le même identificateur de transaction.

Tableau 14. En-tête MQSeries (suite)

MessageType	Usage réservé.
SourceLogicalID	Identificateur logique de l'application source.
DestinationLogicalID	Identificateur logique de l'application cible.
RespondToLogicalID	Identificateur logique auquel le message de réponse doit être envoyé.
CorrelationID	Usage réservé.
GroupStatus	Usage réservé.
ProcessingCategory	Usage réservé.
QosPolicy	Usage réservé.
DeliveryCategory	Usage réservé.
AckRequested	Détermine si l'application source nécessite une réponse.
PublicationTopic	Usage réservé.
SessionID	Usage réservé.
EncryptionStatus	Usage réservé.
TimeStampCreated	Date et heure à laquelle le message a été créé.
TimeStampExpired	Date et heure après lesquelles le message perd sa signification.
Size	Usage réservé.
BodyCategory	Représente le type d'application du message, par exemple OAG ou RosettaNet.
BodyType	Représente l'objet spécifique du message, par exemple ajout d'un bordereau de ventes ou synchronisation de l'inventaire.
BodySecondaryType	Réservé.
UserArea	Zone générale pour les données utilisateur.
BodyData	Données de corps du message.

MQSeries sans MQSeries Integrator

Les valeurs d'en-tête du noyau et les données de corps sont placées dans un document XML. Vous trouverez ci-dessous un exemple du DTD qui décrit le document XML :

```
<!ELEMENT EPICHEADER (HEADER, EPICBODY,USERAREA*)>
<!ELEMENT HEADER (#PCDATA)>
<!ATTLIST HEADER Revision CDATA #FIXED "001">
<!ATTLIST HEADER UniqueID CDATA #REQUIRED>
<!ATTLIST HEADER TransactionID CDATA #REQUIRED>
```

```
<!ATTLIST HEADER MessageType CDATA #REQUIRED>
<!ATTLIST HEADER SourceLogicalID CDATA #REQUIRED>
<!ATTLIST HEADER DestinationLogicalID CDATA #REQUIRED>
<!ATTLIST HEADER RespondToLogicalID CDATA #IMPLIED>
<!ATTLIST HEADER CorrelationID CDATA #IMPLIED>
<!ATTLIST HEADER GroupStatus CDATA #IMPLIED>
<!ATTLIST HEADER ProcessingCategory CDATA #IMPLIED>
<!ATTLIST HEADER QosPolicy CDATA #IMPLIED>
<!ATTLIST HEADER DeliveryCategory CDATA #IMPLIED>
<!ATTLIST HEADER AckRequested CDATA #IMPLIED>
<!ATTLIST HEADER PublicationTopic CDATA #IMPLIED>
<!ATTLIST HEADER SessionID CDATA #IMPLIED>
<!ATTLIST HEADER EncryptionStatus CDATA #IMPLIED>
<!ATTLIST HEADER TimeStampCreated CDATA #REQUIRED>
<!ATTLIST HEADER TimeStampExpired CDATA #REQUIRED>
<!ATTLIST HEADER Size CDATA #IMPLIED>
<!ELEMENT EPICBODY (#PCDATA)> <!-- The data will be escaped -->
<!ATTLIST EPICBODY Size CDATA #IMPLIED>
<!ATTLIST EPICBODY BodyType CDATA #REQUIRED>
<!ATTLIST EPICBODY BodyCategory CDATA #REQUIRED>
<!ATTLIST EPICBODY BodySecondaryType CDATA #IMPLIED>
<!ELEMENT USERAREA (#PCDATA) >
```

En-tête MQSeries Integrator version 1

L'en-tête MQSeries Integrator version 1, RFH1, se compose des éléments suivants :

1. une partie fixe,
2. un en-tête Neon,
3. une section données, qui contient l'en-tête du noyau et les données de corps du message.

Tableau 15. En-tête MQSeries Integrator version 1 — RFH1

Section ou zone	Signification ou utilisation
Partie fixe	Utilisée selon les instructions de MQSeries Integrator version 1.1.
En-tête Neon	Respecte le format d'en-tête Neon.
OPT_APP_GRP	Valeur SourceLogicalId. Issue de l'en-tête du noyau.
OPT_MSG_TYPE	BodyCategory+BodyType. Issu de l'en-tête du noyau. Exemple: si le paramètre BodyCategory est OAG et le paramètre BodyType est SyncItem, la valeur sera alors OAG+SyncItem.

Tableau 15. En-tête MQSeries Integrator version 1 — RFH1 (suite)

Section données	Se compose des valeurs d'en-tête du noyau suivies des données de corps du message.
En-tête du noyau	L'en-tête du noyau figure à l'intérieur des codes<EPICHEADER>*en-tête*</EPICHEADER>. Les valeurs d'en-tête du noyau sont en syntaxe XML. Seuls les attributs accompagnés de valeurs sont présents. Les données réelles ne figurent pas sur des lignes séparées. Exemple du format d'une valeur : <MessageType>*valeur*</MessageType>.
MessageType	Usage réservé.
SourceLogicalID	Identificateur logique de l'application source.
DestinationLogicalID	Identificateur logique de l'application cible.
RespondToLogicalID	Identificateur logique auquel le message de réponse doit être envoyé.
TimeStampCreated	Date et heure à laquelle le message a été créé.
TimeStampExpired	Date et heure après lesquelles le message perd sa signification.
TransactionID	Un message et sa réponse partagent le même identificateur de transaction.
UniqueID	Chaque message possède un identificateur unique.
AckRequested	Détermine si l'application source nécessite une réponse.
ProcessingCategory	Réservé.
BodyCategory	Représente le type d'application du message, par exemple OAG ou RosettaNet.
BodyType	Représente l'objet spécifique du message, par exemple ajout d'un bordereau de ventes ou synchronisation de l'inventaire.
BodySecondaryType	Réservé.
UserArea	Données d'application spécifiques à l'intégration utilisateur.
MsgHeaderVersion	Version de l'en-tête du noyau (réservé).
CorrelationID	Spécifique à l'intégration utilisateur.
GroupStatus	Spécifique à l'intégration utilisateur.
QosPolicy	Réservé.
DeliveryCategory	Réservé.
PublicationTopic	Réservé.

Tableau 15. En-tête MQSeries Integrator version 1 — RFH1 (suite)

SessionID	Réservé.
EncryptionStatus	Réservé.
Données de corps du message	Données de corps du message.

En-tête MQSeries Integrator version 2

L'en-tête MQSeries Integrator version 2, RFH2, se compose des éléments suivants :

1. une partie fixe,
2. <dossier mcd>— libellé du message,
3. <dossier usr>— propriétés définies (par l'utilisateur) de l'application,
4. une section données, qui contient l'en-tête du noyau et les données de corps du message.

Tableau 16. En-tête MQSeries Integrator version 2— RFH2

Section ou zone	**Signification ou utilisation**
Partie fixe	Utilisée selon les instructions de MQSeries Integrator version 2.
<mcd>	XML si le message est en XML. Respectez les règles MQSeries Integrator version 2.
set	Non utilisée par le noyau.
type	Non utilisée par le noyau.
format	XML si le message est en XML. Respectez les règles MQSeries Integrator version 2.
<dossier usr> — propriétés définies (par l'utilisateur) de l'application	Se compose des valeurs d'en-tête du noyau.
En-tête du noyau	Seuls les attributs accompagnés de valeurs sont présents. Les données réelles ne figurent pas sur des lignes séparées.
SourceLogicalID	Identificateur logique de l'application source.
DestinationLogicalID	Identificateur logique de l'application cible.
MessageType	Usage réservé.
RespondToLogicalID	Identificateur logique auquel le message de réponse doit être envoyé.
TimeStampCreated	Date et heure à laquelle le message a été créé.
TimeStampExpired	Date et heure après lesquelles le message perd sa signification.

Tableau 16. En-tête MQSeries Integrator version 2— RFH2 (suite)

TransactionID	Un message et sa réponse partagent le même identificateur de transaction.
UniqueID	Chaque message possède un identificateur unique.
ProcessingCategory	Réservé.
BodyCategory	Représente le type d'application du message, par exemple OAG ou RosettaNet.
BodyType	Représente l'objet spécifique du message, par exemple ajout d'un bordereau de ventes ou synchronisation de l'inventaire.
BodySecondaryType	Réservé.
AckRequested	Détermine si l'application source nécessite une réponse.
UserArea	Données d'application spécifiques à l'intégration utilisateur.
MsgHeaderVersion	Version de l'en-tête du noyau (réservé).
CorrelationID	Spécifique à l'intégration utilisateur.
GroupStatus	Spécifique à l'intégration utilisateur.
QosPolicy	Réservé.
DeliveryCategory	Réservé.
PublicationTopic	Réservé.
SessionID	Réservé.
EncryptionStatus	Réservé.
Section données	Données de corps du message.

Annexe D. Modèle de fichier de configuration

Cette section indique la versiondu fichier aqmconfig.xml qui était courante à la date de cette publication. «Modèle de fichier de configuration minimum» à la page 127 indique la version du fichier aqmconfig.minimum.xml qui était courante à la date de cette publication. Voir la version la plus récente des fichiers aqmconfig.xml et aqmconfig.minimum.xml dans le répertoire samples d'installation du noyau ; il se peut que les exemples indiqués ici soient dépassés.

Le «Le fichier de configuration» à la page 69 contient des informations concernant l'interprétation et l'édition du ficher de configuration.

Plusieurs identificateurs d'applications sont inclus dans cet exemple de fichier de configuration. Un ensemble d'entrées est indiqué sous chaque identificateur d'application. Le modèle de fichier de configuration contient les identificateurs d'applications suivants :

- TEST1
- TEST1Daemon
- TEST2
- TEST3
- TraceClient
- TraceServer

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- aqmconfig.xml 1.01 09Mar01                                          -->
<!-- Used for MQSeries Adapter Kernel -->
<!-- Modèle de configuration AQM.           -->
<!--  -->
<!-- Copyright (c) 2001 International Business Machines. All Rights Reserved. -->
<!--  -->
<!-- This configuration file is as an example only. -->
<!--  -->
<!-- IBM MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF THIS -->
<!-- SAMPLE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE -->
<!-- IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR -->
<!-- PURPOSE, OR NON-INFRINGEMENT. -->
<!--  -->
<!-- CopyrightVersion 1.0 -->
<!--  -->

   <Epic o="ePIC">
      <!-- If getObject is called this indicates the top level directory -->
      <!-- where the JNDI file system context will retrieve objects from. -->
      <!-- This defaults to the current directory if this key is not present. -->
      <!-- All applications share this context root. -->
      <context>file:///epic/configContext</context>
      <!-- Example using a drive letter 'c' -->
      <!--
      <context>file://c:/E/runtimefiles</context>
      -->
      <ePICApplications o="ePICApplications">
         <!-- The following is for sample Test Application ID: TEST1 with a  -->
         <!-- sample AdapterDaemon named TEST1Daemon  -->
         <ePICApplication epicappid="TEST1">
     <!--  Audit Logging on/off.  Requires WebSphere Business Integrator product. -->
     <!--  If no entry defaults to false. -->
```

```
       <epiclogging>false</epiclogging>
<!--  Tracing on/off.  If no entry defaults to false. -->
       <epictrace>false</epictrace>
<!-- Trace levels - Uses the jlog com.ibm.logging.IRecordType constants, -->
<!-- common constants: -->
<!-- 0=TYPE_NONE (No messages), 1=TYPE_INFO, 512=TYPE_ERROR_EXC (Exceptions), -->
<!-- 513=TYPE_INFO | TYPE_ERROR_EXC, -1=TYPE_ALL (All possible messages). -->
<!-- No entry defaults to TYPE_NONE -->
       <epictracelevel>-1</epictracelevel>
<!-- Name of the Trace application id.  Will be used for -->
<!-- trace configuration information.  Defaults to TraceClient -->
       <epictraceclientid>TraceClient</epictraceclientid>
<!-- When processing messages into the application. -->
<!-- LogonInfo class name used for connecting to an application. -->
  <!-- Will be used by the AdapterDaemon.  If no entry will default -->
<!-- to com.ibm.epic.adapters.eak.adapterdaemon.EpicLogonDefault. -->
<epiclogoninfoclassname>com.ibm.epic.adapters.eak.adapterdaemon.EpicLogonDefault
 </epiclogoninfoclassname>
       <AdapterRouting cn="epicadapterrouting">
          <!-- MQSeries Q Manager for this application use, no entry -->
          <!-- uses the default Q Manager.  A value of DEFAULT means -->
          <!-- use the default Q Manager. -->
          <epicmqppqueuemgr>DEFAULT</epicmqppqueuemgr>
          <!-- Use the remote Q Manager for sending messages.  Remote queue -->
          <!-- definitions are not required.  true - use remote Q Manager, -->
          <!-- false - do not use remote Q Manager.  No entry defaults to false -->
          <epicuseremotequeuemanagertosend>false</epicuseremotequeuemanagertosend>
          <!-- MQSeries Client hostname for where the MQSeries server -->
          <!-- resides for TEST1.  Required if using MQSeries Client -->
          <!--
          <epicmqppqueuemgrhostname>localhost</epicmqppqueuemgrhostname>
   -->
          <!-- MQSeries Client port to use for where the MQSeries server -->
          <!-- resides for TEST1.  No entry defaults to MQSeries default -->
          <!--
          <epicmqppqueuemgrportnumber>1414</epicmqppqueuemgrportnumber>
   -->
          <!-- MQSeries Client channel name to use for the MQSeries server, required -->
          <!--
          <epicmqppqueuemgrchannelname>xyz</epicmqppqueuemgrchannelname>
   -->
          <!-- JMS example for TEST1.  Refers to the JMS Connection factory name. -->
          <!-- Requires the attribute describing the object plus the attributes value. -->
          <!-- For JMS the attribute is 'cn'. -->
          <!--
          <epicjmsconnectionfactoryname>cn=QCFTEST1</epicjmsconnectionfactoryname>
   -->
          <ePICBodyCategory epicbodycategory="DEFAULT">
             <ePICBodyType epicbodytype="DEFAULT">
<!-- Contains the Command selection criteria when processing -->
<!-- a message into an application.  Will be used by the -->
 <!-- AdapterDaemon - Command to invoke. -->
                <epiccommandclassname>com.ibm.epic.adapters.eak.samples.SampleCAdapterWrapper
                </epiccommandclassname>
<!-- Represents the type of command the "epiccommandclassname" -->
<!-- represents.  MQAKEAB is the EAB style interface.  MQAKEJB -->
<!-- is an EJB Service Session Bean.  No entry defaults to MQAKEAB -->
                <epiccommandtype>MQAKEAB</epiccommandtype>
<!-- If the "epiccommandtype" is "MQAKEJB" this entry is the -->
<!-- method name to invoke.  No entry defaults to "execute". -->
                <epiccommandejbmethod>execute</epiccommandejbmethod>
<!-- If the "epiccommandtype" is "MQAKEJB" this entry is the -->
<!-- parameter type for the method specified by "epiccommandejbmethod". -->
<!-- This will be the same datatype returned by the -->
<!-- TerminalDataContainerMapper.  No entry defaults -->
<!-- to "com.ibm.mqao.mqak.ejbclient.TDCGenericMapper". -->
                <epiccommandejbmethodparmtype>com.ibm.mqao.mqak.ejbclient.TDCGenericMapper
                </epiccommandejbmethodparmtype>
<!-- If the "epiccommandtype" is "MQAKEJB" this entry is the -->
<!-- URL where the EJB specified in "epiccommandclassname" -->
<!-- has been deployed in the form "IIOP://hostname:900/". -->
<!-- No entry defaults to "IIOP:///". -->
                <epiccommandejburl>IIOP:///</epiccommandejburl>
<!-- If the "epiccommandtype" is "MQAKEJB" this entry is the -->
<!-- name of the Initial Context Factory used to to lookup the -->
<!-- home name for the EJB specified in "epiccommandclassname". -->
<!-- No entry defaults to "com.ibm.ejs.ns.jndi.CNInitialContextFactory". -->
                <epiccommandejbinitialcontext>com.ibm.ejs.ns.jndi.CNInitialContextFactory
                </epiccommandejbinitialcontext>
<!-- If the "epiccommandtype" is "MQAKEJB" this entry is the -->
<!-- name of the Mapper the Worker uses for creating the  -->
<!-- "epiccommandejbmethodparmtype" object passed in to the  -->
<!-- "epiccommandejbmethod" in to the "epiccommandclassname". -->
<!-- No entry defaults to "com.ibm.mqao.mqak.ejbclient.TDCGenericMapper". -->
                <epiccommandejbmapper>com.ibm.mqao.mqak.ejbclient.TDCGenericMapper
```

```
                </epiccommandejbmapper>
<!-- Default destinations to send messages to. -->
        <!-- Single destination. -->
                <epicdestids>TEST2</epicdestids>
        <!-- Multiple destinations. -->
                <!--
                <epicdestids>
                     <Value>TEST2</Value>
                     <Value>TEST3</Value>
                </epicdestids>
                -->
                <!-- Receive transport communications mode this application -->
                <!-- wants for receiving messages. -->
                <!-- For MQSeries normal mode use MQPP. -->
                <!-- For MQSeries using an RFH1 header format use MQRFH1, -->
                <!-- when using MQSeries Integrator V1 -->
                <!-- For MQSeries using an RFH2 header format use MQRFH2, -->
                <!-- when using MQSeries Integrator V2 -->
                <!-- For file normal mode use FILE. -->
                <epicreceivemode>MQPP</epicreceivemode>
                <!-- How to format the message for the receive mode. -->
                <!-- Entry is the class name of the formatter which -->
                <!-- must be for the receive mode -->
                <!-- Receive modes MQPP, MQRFH1, MQRFH2, FILE have -->
                <!-- default receive modes -->
                <epicmessageformatter>com.ibm.epic.adapters.eak.nativeadapter.MQNMBDFormatter
                </epicmessageformatter>
 <!-- JMS formatter for mode for MQSeries provider implementation -->
 <!--
                <epicmessageformatter>com.ibm.epic.adapters.eak.nativeadapter.JMSNMRFH2Formatter
                </epicmessageformatter>
 -->
 <!-- Receive Time out in milliseconds ie. 1000 = 1 second, -->
 <!-- -1 means never ending.  No entry defaults to 0 -->
 <!-- milliseconds.  Used when receiving messages. -->
                <epicreceivetimeout>30000</epicreceivetimeout>
 <!-- MQSeries queue for this application to receive messages -->
 <!-- from for receive modes MQPP, MQRFH1, MQRFH2 -->
                <epicreceivemqppqueue>TEST1AIQ</epicreceivemqppqueue>
 <!-- MQSeries queue required by the AdapterWorker when -->
 <!-- errors encountered processing a message -->
 <!-- for receive modes MQPP, MQRFH1, MQRFH2 -->
                <epicerrormqppqueue>TEST1AEQ</epicerrormqppqueue>
 <!-- MQSeries reply queue required for synchronous request/replies -->
 <!-- for receive modes MQPP, MQRFH1, MQRFH2 -->
                <epicreplymqppqueue>TEST1RPL</epicreplymqppqueue>
 <!-- JMS receive mode, refers to the JMS queue object name for -->
 <!-- this application to receive messages from. -->
                <!-- Requires the attribute describing the object plus the attribute's value. -->
                <!-- For JMS the attribute is 'cn'. -->
                <epicjmsreceivequeuename>cn=TEST1AIQ</epicjmsreceivequeuename>
 <!-- JMS receive mode, refers to the JMS queue object name for -->
 <!-- errors required by the AdapterWorker when errors -->
 <!-- encountered processing a message. -->
                <!-- Requires the attribute describing the object plus the attribute's value. -->
                <!-- For JMS the attribute is 'cn'. -->
                <epicjmserrorqueuename>cn=TEST1AEQ</epicjmserrorqueuename>
 <!-- JMS receive mode, refers to the JMS queue object name for -->
 <!-- the reply queue, required for synchronous request/replies -->
                <!-- Requires the attribute describing the object plus the attribute's value. -->
                <!-- For JMS the attribute is 'cn'. -->
                <epicjmsreplyqueuename>cn=TEST1RPL</epicjmsreplyqueuename>
 <!-- In FILE receive mode, directory for this application to receive messages from -->
                <epicreceivefiledir>./TEST1AID</epicreceivefiledir>
 <!-- In FILE receive mode, interim directory for this application to -->
 <!-- hold received messages until committed. -->
                <epiccommitfiledir>./TEST1ACD</epiccommitfiledir>
 <!-- In FILE receive mode, directory for this application to put error messages -->
 <!-- File receive mode, directory required by the AdapterWorker when -->
 <!-- errors encountered processing a message -->
                <epicerrorfiledir>./TEST1AED</epicerrorfiledir>
            </ePICBodyType>
         </ePICBodyCategory>
      </AdapterRouting>
   </ePICApplication>
   <!-- The following is for sample AdapterDaemon 'TEST1Daemon' -->
   <!-- for the 'TEST1' application  -->
   <ePICApplication epicappid="TEST1Daemon">
      <epictrace>false</epictrace>
      <epictracelevel>-1</epictracelevel>
      <ePICAdapterDaemonExtensions cn="epicappextensions">
  <!-- Dependency appid, if no entry then will default -->
  <!-- to the application id of the daemon.     -->
         <epicdepappid>TEST1</epicdepappid>
  <!-- Minimum number of workers the AdapterDaemon will start. -->
```

```
      <!-- No entry defaults to 1. -->
            <epicminworkers>1</epicminworkers>
         </ePICAdapterDaemonExtensions>
       </ePICApplication>
       <!-- The following is for Test Application ID: TEST2 -->
       <!-- Refer to TEST1 for explanations and possible additional entries. -->
       <ePICApplication epicappid="TEST2">
          <epictrace>true</epictrace>
          <epictracelevel>512</epictracelevel>
          <AdapterRouting cn="epicadapterrouting">
             <epicmqpppqueuemgr>DEFAULT</epicmqpppqueuemgr>
             <ePICBodyCategory epicbodycategory="DEFAULT">
                <ePICBodyType epicbodytype="DEFAULT">
                   <epiccommandclassname>com.ibm.epic.adapters.eak.test.InstallVerificationTest
                   </epiccommandclassname>
                   <epicreceivemode>MQPP</epicreceivemode>
                   <epicreceivemqpppqueue>TEST2AIQ</epicreceivemqpppqueue>
                   <epicerrormqpppqueue>TEST2AEQ</epicerrormqpppqueue>
                   <epicreplymqpppqueue>TEST2RPL</epicreplymqpppqueue>
                </ePICBodyType>
             </ePICBodyCategory>
          </AdapterRouting>
       </ePICApplication>
       <!-- The following is for Test Application ID: TEST3 -->
       <!-- Refer to TEST1 for explanations and possible additional entries. -->
       <ePICApplication epicappid="TEST3">
          <AdapterRouting cn="epicadapterrouting">
             <epicmqpppqueuemgr>DEFAULT</epicmqpppqueuemgr>
             <ePICBodyCategory epicbodycategory="DEFAULT">
                <ePICBodyType epicbodytype="DEFAULT">
                   <epicdestids>TEST1</epicdestids>
                   <epicreceivemode>MQPP</epicreceivemode>
                   <epicreceivemqpppqueue>TEST3AIQ</epicreceivemqpppqueue>
                </ePICBodyType>
             </ePICBodyCategory>
          </AdapterRouting>
       </ePICApplication>
<!-- The following is for sample Trace Client Application ID: TraceClient -->
<!-- Contains the TraceClient configuration information for doing tracing. -->
<!-- This is the application id value in the 'epictraceclientid' element -->
<!-- configured for the application wanting to do tracing -->
        <ePICApplication epicappid="TraceClient">
          <ePICTraceExtensions cn="epicappextensions">
       <!-- Dependency Trace Server application id used for SocketHandler -->
       <!-- and ENAHandler (uses MQSeries), defaults to TraceServer -->
             <epicdepappid>TraceServer</epicdepappid>
             <!-- Write messages synchronously (true) or asynchronously (false), -->
             <!-- defaults to false (write messages asynchronously).  This is    -->
             <!-- used when giving the messages to the handlers. -->
             <epictracesyncoperation>false</epictracesyncoperation>
       <!-- Default Trace message file to use if none passed in to the -->
          <!-- writeTrace method call. Defaults to this file if not indicated -->
             <epictracemessagefile>com.ibm.epic.trace.client.TraceMessage</epictracemessagefile>
       <!-- Handlers to load.  Handlers do the actual processing of the -->
       <!-- Trace message.  If the default trace client id 'TraceClient' -->
       <!-- is used then the handler defaults to the -->
       <!-- com.ibm.logging.ConsoleHandler.  If the default trace client -->
       <!-- id 'TraceClient' is not used, the handler has to be specified. -->
       <!-- A Single Trace Handler -->
             <epictracehandler>com.ibm.logging.ConsoleHandler</epictracehandler>
       <!-- Multiple Trace Handlers -->
       <!--
             <epictracehandler>
                <Value>com.ibm.logging.ConsoleHandler</Value>
                <Value>com.ibm.logging.SocketHandler</Value>
             </epictracehandler>
       -->
             <!-- Handler definitions.  Available definitions depend on the -->
             <!-- handler.  Formatters are used for formatting the trace message.-->
             <ePICTraceHandler epictracehandler="com.ibm.logging.ConsoleHandler">
                <!-- ConsoleHandler formatter to use, defaults to this formatter if none provided. -->
                <epictraceformatter>com.ibm.epic.trace.client.EpicTraceFormatter</epictraceformatter>
             </ePICTraceHandler>
             <ePICTraceHandler epictracehandler="com.ibm.logging.FileHandler">
                <!-- FileHandler formatter to use, defaults to this formatter if none provided. -->
                <epictraceformatter>com.ibm.epic.trace.client.EpicTraceFormatter</epictraceformatter>
                <!-- Trace filename to use, defaults to trc.log in the current directory. -->
                <epictracefilename>trc.log</epictracefilename>
             </ePICTraceHandler>
             <ePICTraceHandler epictracehandler="com.ibm.epic.trace.client.ENAHandler">
                <!-- ENAHandler formatter to use, defaults to this formatter if none provided. -->
                <epictraceformatter>com.ibm.epic.trace.client.EpicXMLFormatter</epictraceformatter>
             </ePICTraceHandler>
             <ePICTraceHandler epictracehandler="com.ibm.logging.SocketHandler">
                <!-- SocketHandler formatter to use, defaults to this formatter if none provided. -->
```

```
                <epictraceformatter>com.ibm.epic.trace.client.EpicXMLFormatter</epictraceformatter>
             </ePICTraceHandler>
          </ePICTraceExtensions>
       </ePICApplication>
<!-- The following is for sample Trace Server Application ID: TraceServer -->
<!-- Contains the TraceServer configuration information. -->
<!-- This is the application id pointed to by the trace client -->
<!-- epicdepappid value.  Definitions are similar to TraceClient example. -->
       <ePICApplication epicappid="TraceServer">
          <AdapterRouting cn="epicadapterrouting">
             <epicmqppqueuemgr>DEFAULT</epicmqppqueuemgr>
             <ePICBodyCategory epicbodycategory="DEFAULT">
                <ePICBodyType epicbodytype="DEFAULT">
                   <epicreceivemode>MQPP</epicreceivemode>
                   <epicreceivemqppqueue>TraceServerAIQ</epicreceivemqppqueue>
                </ePICBodyType>
             </ePICBodyCategory>
          </AdapterRouting>
          <ePICTraceExtensions cn="epicappextensions">
      <!-- Write messages synchronously/asynchronously (true/false (default)). -->
             <epictracesyncoperation>false</epictracesyncoperation>
      <!-- Trace message file.  Defaults to this file if not indicated -->
             <epictracemessagefile>com.ibm.epic.trace.server.TraceServerMessage</epictracemessagefile>
             <!-- Handlers to load, for multiple handlers see TraceClient example. -->
             <!-- If the default trace server id 'TraceServer' is used then the handler -->
             <!-- defaults to the com.ibm.logging.MultiFileHandler. -->
             <!-- Note: Do not use SocketHandler or ENAHandler for the trace server. -->
             <epictracehandler>com.ibm.logging.MultiFileHandler</epictracehandler>
      <!-- Handler definitions for com.ibm.logging.SocketHandler -->
      <!-- Formatter to use, defaults to this formatter if none provided.-->
             <ePICTraceHandler epictracehandler="com.ibm.logging.SocketHandler">
 <!-- Entries when using socket handler from the TraceClient and -->
 <!-- starting the Trace Server in socket receive mode. -->
 <!-- SocketHandler host machine, defaults to localhost -->
                <epictracesocketserverhost>localhost</epictracesocketserverhost>
         <!-- SocketHandler port number, defaults to 8181 -->
                <epictraceportnumber>8181</epictraceportnumber>
             </ePICTraceHandler>
      <!-- Formatter to use, defaults to this formatter if none provided.-->
             <ePICTraceHandler epictracehandler="com.ibm.logging.ConsoleHandler">
         <!-- ConsoleHandler formatter to use, defaults to this formatter if none provided.-->
                <epictraceformatter>com.ibm.epic.trace.client.ReFormatter</epictraceformatter>
             </ePICTraceHandler>
             <ePICTraceHandler epictracehandler="com.ibm.logging.MultiFileHandler">
 <!-- MultiFileHandler formatter to use, defaults to this formatter if none provided. -->
                <epictraceformatter>com.ibm.epic.trace.client.ReFormatter</epictraceformatter>
 <!-- MultiFileHandler trace base filename to use, defaults to trc.log in the -->
 <!-- current directory.  The actual filename will be for this -->
 <!-- example trcx.log, where x is a numeric number starting at -->
 <!-- 0 and going up to the number of trace files specified. -->
                <epictracefilename>trc.log</epictracefilename>
 <!-- MultiFileHandler number of trace files, defaults to 3 -->
                <epictracefilenumber>3</epictracefilenumber>
 <!-- MultiFileHandler file size in number of bytes, defaults to -->
                <epictracefilesize>1000000</epictracefilesize>
             </ePICTraceHandler>
          </ePICTraceExtensions>
       </ePICApplication>
    </ePICApplications>
```

Modèle de fichier de configuration minimum

Vous trouverez dans cette section un exemple de fichier de configuration minimum à utiliser avec MQSeries Adapter Kernel. Reportez-vous à la section «Ajout des informations de l'adaptateur à la configuration» à la page 89 pour des informations sur le fichier de configuration minimum.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- aqmconfig.minimum.xml 1.00 00/11/07                                    -->
<!-- Used for MQSeries Adapter Kernel -->
<!-- Sample AQM Configuration showing a minimum configuration for the       -->
<!-- following conditions:                                                   -->
<!-- 1) Going from applicationid TEST1 to TEST2.  TEST1 is not receiving  -->
<!--    messages. -->
<!-- 2) TEST2 has no special application requirements.  -->
<!-- 3) TEST2 is using 1 worker.                         -->
<!-- 4) Using MQSeries with the default QManager installed on each machine.    -->
<!--    and using default format.                                           -->
<!-- 5) No specific body category and body type being used.                 -->
<!-- 6) Using default tracing to the console.   -->
```

```
<!--  -->
<!--  -->
<!--  -->
<!-- Copyright (c) 2000 International Business Machines. All Rights Reserved. -->
<!--  -->
<!-- This configuration file is as an example only. -->
<!--  -->
<!-- IBM MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF THIS -->
<!-- SAMPLE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE -->
<!-- IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR -->
<!-- PURPOSE, OR NON-INFRINGEMENT. -->
<!--  -->
<!-- CopyrightVersion 1.0 -->
<!--     -->
   <Epic o="ePIC">
      <ePICApplications o="ePICApplications">
         <!-- The following is for sample Test Application ID: TEST1  -->
         <ePICApplication epicappid="TEST1">
     <!--  Tracing on/off.  If no entry defaults to false. -->
            <epictrace>false</epictrace>
     <!-- Trace levels - 512=TYPE_ERROR_EXC (Exceptions),-1=TYPE_ALL (All possible messages). -->
            <epictracelevel>0</epictracelevel>
            <AdapterRouting cn="epicadapterrouting">
               <epicmqpqueuemgr>DEFAULT</epicmqpqueuemgr>
               <ePICBodyCategory epicbodycategory="DEFAULT">
                  <ePICBodyType epicbodytype="DEFAULT">
     <!-- Default destinations to send messages to. -->
                     <epicdestids>TEST2</epicdestids>
                  </ePICBodyType>
               </ePICBodyCategory>
            </AdapterRouting>
         </ePICApplication>
         <!-- The following is for Test Application ID: TEST2 -->
         <ePICApplication epicappid="TEST2">
            <epictrace>false</epictrace>
            <epictracelevel>512</epictracelevel>
            <AdapterRouting cn="epicadapterrouting">
               <epicmqpqueuemgr>DEFAULT</epicmqpqueuemgr>
               <ePICBodyCategory epicbodycategory="DEFAULT">
                  <ePICBodyType epicbodytype="DEFAULT">
      <!-- AdapterDaemon - Command to invoke. -->
                     <epiccommandclassname>com.ibm.epic.adapters.eak.samples.SampleCAdapterWrapper
                     </epiccommandclassname>
                     <epicreceivemode>MQ</epicreceivemode>
      <!-- Receive Time out in milliseconds ie. 1000 = 1 second, -->
      <!-- -1 means never ending.  No entry defaults to 0. -->
      <!-- milliseconds.  Used when receiving messages. -->
                     <epicreceivetimeout>30000</epicreceivetimeout>
                     <epicreceivemqppqueue>TEST2AIQ</epicreceivemqppqueue>
                     <epicerrormqppqueue>TEST2AEQ</epicerrormqppqueue>
                     <epicreplymqppqueue>TEST2RPL</epicreplymqppqueue>
                  </ePICBodyType>
               </ePICBodyCategory>
            </AdapterRouting>
         </ePICApplication>
      </ePICApplications>
   </Epic>
```

Annexe E. Modèle de fichier d'installation

Le fichier suivant est un exemple de fichier aqmsetup qui définit plusieurs valeurs de configuration initiale du noyau, dont plusieurs variables d'environnement. Reportez-vous à la section «Le fichier d'installation» à la page 68 pour de plus amples informations sur ce fichier. Le fichier aqmsetup est situé dans le répertoire d'installation racine du noyau samples.

```
#
# aqmsetup 1.01 01/03/27
# Sample AQM Adapter runtime parameter configuration file entries.
#
# Copyright (c) 2001 International Business Machines. All Rights Reserved.
#
# This configuration file is as an example only.
#
# IBM MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF THIS
# SAMPLE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
# IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR
# PURPOSE, OR NON-INFRINGEMENT.
#
# CopyrightVersion 1.0
#
#
# Pound (#) signs are comments.
#
##################################################################
#
# Use Websphere Business Integrator(WSI) product 5724-A78 LDAP
# Directory Services or configuration file. No entry defaults to
# true (use configuration file). To use the WSI directory service
# set the value to false.  Refer to the WSI documentation for
# specifics on using the directory service.
#AdapterDirectoryUseFileFlag=true
# When using Websphere Business Integrator(WSI) product 5724-A78 LDAP
# Directory Services this additional entry is required. Refer to the
# WSI documentation for specifics on using the directory service.
#DirectoryServices=ChangeToDestDir/samples/DirectoryServices.properties
# Location of configuration file aqmconfig.xml when not using
# the Websphere Business Integrator(WSI) product 5724-A78 LDAP
# Directory Services.
# No entry defaults to current directory.
#AQMConfig=ChangeToDestDir/samples
#
##################################################################
##################################################################
# XML DTD Catalogs and Directories - where to locate DTD's if not
# in the current directory.
# Format: XML_DTD_DIRECTORY_x=ddd where x is a numeric suffix to
# be incremented for each key and ddd is the directory.
# The numeric suffix's must start with 1 and be contiguous.
```

```
#####################################################################
XML_DTD_DIRECTORY_1=ChangeToDestDir/runtimefiles/oag
#XML_DTD_DIRECTORY_2=ChangeToDestDir/runtimefiles
#
#####################################################################
# Java JNI Environment Variables for C Interface for increasing
# the amount of memory used.  This applies to when a C module
# is instantiating a JVM.  When a C Interface is being called
# from within JAVA the JVM is already established.
#####################################################################
# The stack memory is used for holding local function, function
# parameters, local variable references.
# Native stack is used for non-Java calls from within Java such
# as to C code.  Stack size in bytes to use.
# Default is 128 kilobytes on NT.
#AQM_JNI_NATIVESTACKSIZE=1048576
# Java stack is for Java method calls and local variables.
# Stack size in bytes to use.
# Default is 400 kilobytes on NT.
#AQM_JNI_JAVASTACKSIZE=4194304
# The heap memory is used for storing instantiated Java objects
# Minimum heap size in bytes to start with.
# Default is 1 megabyte on NT.
#AQM_JNI_MINHEAPSIZE=16777216
# Maximum heap size in bytes which can be used.
# Default is 16 megabytes on NT.
#AQM_JNI_MAXHEAPSIZE=268435426
#
#####################################################################
#  Designate end of configuration file
#####################################################################
*ENDCFG
```

Remarques

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

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

IBM EMEA Director of Licensing
IBM Europe Middle-East Africa
Tour Descartes
92066 Paris-La Défense Cedex 50
France

Pour le Canada, veuillez adresser votre courrier à : IBM Director of Commercial Relations IBM Canada Ltd 3600 Steeles Avenue East Markham, Ontario L3R 9Z7 Canada

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, Japon

Le paragraphe suivant ne s'applique ni au Royaume-Uni, ni dans aucun pays dans lequel il serait contraire aux lois locales. LE PRESENT DOCUMENT EST LIVRE EN L'ETAT. IBM DECLINE TOUTE RESPONSABILITE, EXPLICITE OU IMPLICITE, RELATIVE AUX INFORMATIONS QUI Y SONT CONTENUES, Y COMPRIS EN CE QUI CONCERNE LES GARANTIES DE VALEUR MARCHANDE OU

D'ADAPTATION A VOS BESOINS. Certaines juridictions n'autorisent pas l'exclusion des garanties implicites, auquel cas l'exclusion ci-dessus ne vous sera pas applicable.

Le présent document peut contenir des inexactitudes ou des coquilles. Il est mis à jour périodiquement. Chaque nouvelle édition inclut les mises à jour. IBM peut modifier sans préavis les programmes et les logiciels qu'il décrit.

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 United Kingdom Laboratories,
Mail Point 151,
Hursley Park,
Winchester,
Hampshire,
Royaume-Uni
SO21 2JN.

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.

Marques

Les termes qui suivent sont des marques d'International Business Machines Corporation dans certains pays :

AIX
AS/400
IBM
MQSeries
OS/400
RISC System/6000
RS/6000
WebSphere

Lotus et LotusScript sont des marques de Lotus Development Corporation aux Etats-Unis et/ou dans d'autres pays.

Java et les marques et logos Java sont des marques de Sun Microsystems, Inc. dans certains pays.

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

D'autres sociétés sont propriétaires des autres marques, noms de produits ou de services qui pourraient apparaître dans ce document.

Glossaire

Le glossaire contient les termes *clés* ainsi que leur signification dans la documentation MQSeries Adapter Kernel.

Si un concept ou un terme particulier apparaît dans une seule section, il se peut qu'il ne figure pas dans le glossaire. Il est toutefois possible de le trouver dans l'«Index» à la page 141.

Ce glossaire ne contient pas de termes relatifs à d'autres produits IBM comme MQSeries.

adaptateur : Il s'agit du résultat obtenu avec MQSeries Adapter Builder. En général, l'utilisateur crée chaque adaptateur de manière à ce qu'il soit spécifique à *un seul type de message* envoyé depuis ou vers une application. Ainsi, les adaptateurs eux-mêmes ne font pas partie de MQSeries Adapter Offering. Un adaptateur se compose d'un code source C ou Java qui est compilé dans une bibliothèque partagée. Lorsque les adaptateurs et MQSeries Adapter Kernel fonctionnent simultanément, ils assurent la fonction d'exécution de MQSeries Adapter Offering. Selon la façon dont il a été créé par l'utilisateur dans MQSeries Adapter Builder, l'adaptateur peut contenir un grande diversité de fonctions comme le flux de commande, le flot de données, la navigation séquentielle, le branchement conditionnel y compris la décision et l'itération, la saisie des données, l'enregistrement en mémoire du contexte de données, la transformation des données élémentaires, le contrôle transactionnel, les opérations logiques et le code personnalisé. Vous pouvez réutiliser les adaptateurs que vous avez créés.

Reportez-vous aux sections «Type de message» à la page 139, «Application source» à la page 136 et «Application cible» à la page 136.

Adaptateur cible : Adaptateur qui exécute les tâches suivantes :

- Reçoit un message (du noyau et de MQSeries ou d'un autre logiciel de messagerie) envoyé par un adaptateur source.
- Traite le message d'intégration selon la manière dont l'adaptateur a été créé.
- Transforme le message d'intégration en un message formaté spécifique à l'application que l'application cible peut recevoir.
- Envoie le message à l'application cible en utilisant une interface spécifique de l'application.
- Avertit l'agent une fois que l'envoi du message à l'application cible est terminé, pour permettre à l'agent d'envoyer un accusé de réception.

Si l'application cible peut recevoir le message d'intégration, il se peut que l'adaptateur cible ne soit pas nécessaire.

Il existe un adaptateur cible pour chaque type de message. Une application cible peut généralement accepter plusieurs types de messages ; dans la plupart des cas, par conséquent, une application cible est prise en charge par plusieurs adaptateurs cibles. Reportez-vous à la section «Adaptateur».

Adaptateur de service Java : Type d'adaptateur en langage Java qui, dans un environnement JMS Listener, assure les fonctions d'un démon d'adaptateur, d'un agent et d'un adaptateur cible.

Adaptateur natif : Logiciel utilisé pour l'envoi et la réception d'objets collecteur de messages.

Adaptateur natif MQSeries Adapter Kernel : Synonyme d'adaptateur natif.

Adaptateur source : Adaptateur qui exécute les tâches suivantes :

- Accepte ou acquiert les données structurées d'une application source (généralement en utilisant une interface spécifique de l'application développée en dehors de l'adaptateur).
- Traite les données structurées selon la manière dont l'adaptateur a été modélisé.
- Transforme les données structurées en un format de message d'intégration.
- En utilisant le noyau, place le message dans une file d'attente de messages, pour le fournir à un ou plusieurs adaptateurs cibles et de là à l'application cible.

Pour chaque type de message, il y a un adaptateur source. En général, une application source peut envoyer plusieurs types de messages ; par conséquent, dans la plupart des cas, une application source est prise en charge par plusieurs adaptateurs source.

Reportez-vous à la section «Adaptateur» à la page 135.

Agent : Logiciel faisant partie du noyau. L'agent n'est utilisé que dans le modèle de sortie par insertion. Le démon de l'adaptateur démarre et crée les agents. Chaque agent gère un adaptateur natif. L'agent délivre chaque message à l'adaptateur cible approprié.

Application cible : Programme nécessaire pour la réception via un réseau informatique des données d'un programme (désigné application source) résidant généralement sur un autre ordinateur.

Application source : Programme nécessaire pour l'envoi des données via un réseau informatique à un programme (désigné application cible) résidant généralement sur un autre ordinateur.

bean de message agent : Bean d'entreprise qui joue le rôle d'un agent lorsque WebSphere Application Server est utilisé du côté cible du noyau.

BOD : Document objet de gestion. Représentation d'un processus de gestion standard qui s'applique au sein d'une entreprise ou entre plusieurs entreprises. Par exemple, l'ajout d'un bon de commande, l'indication de la disponibilité du produit ou l'ajout d'un bordereau de vente. Les BOD sont définis par l'OAG qui utilise XML. Reportez-vous aux sections «OAG» à la page 139 et «XML» à la page 139.

Les BOD peuvent être utilisés par MQSeries Adapter Offering pour définir les corps de messages dans ses messages d'intégration.

Catégorie de corps : Donnée contenue dans un message qui représente le type d'application du message, par exemple OAG ou RosettaNet. Elle appartient à la série des valeurs de contrôle du message. Pour plus d'informations, reportez-vous à la section «Valeurs de contrôle de message» à la page 139.

La catégorie de corps aide également à déterminer le type du message. Pour plus d'informations, reportez-vous à la section «Type de message» à la page 139.

Classe de connexion : Il s'agit d'une classe Java spécifique à chaque application cible, qui peut être utilisée pour faciliter l'acheminement du message vers l'application cible. La classe de connexion n'est nécessaire que lorsque l'adaptateur cible doit se connecter à l'application cible avant l'acheminement du message. Chaque classe de connexion est écrite par l'utilisateur. L'agent instancie la classe de connexion. La classe de connexion recherche dans le fichier de configuration les valeurs dont l'adaptateur cible a besoin pour prendre en charge l'interface spécifique de l'application avec l'application cible. En général, ces valeurs sont des paramètres de connexion. Ainsi, les valeurs sont disponibles pour l'adaptateur cible.

Une classe de connexion factice n'ayant aucun effet est fournie avec le noyau.

Client trace : Composant du noyau qui écrit les messages trace.

Côté cible du noyau : Fonctions du noyau dont l'exécution est lancée lors de l'extraction du

message de la file d'attente de messages et s'achève lorsque le message est envoyé à l'adaptateur cible.

Côté source du noyau : Partie des fonctions du noyau dont l'exécution est lancée lors de la réception du message provenant de l'adaptateur source et qui s'achève lorsque le message est placé dans une file d'attente de messages.

Démon d'adaptateur : Logiciel exécutable qui fait partie du noyau. Le démon de l'adaptateur n'est utilisé que dans le modèle de communication par insertion. Il sert à instancier les agents. Après son démarrage, le démon de l'adaptateur reste actif. Pour chaque application cible, il peut y avoir un ou plusieurs démons d'adaptateur.

Dans certains cas, le démon de l'adaptateur joue le rôle d'une application cible. Il exécute la fonction nécessaire en utilisant, par exemple, un adaptateur cible pour envoyer un message électronique ou pour écrire ou effectuer un enregistrement dans un fichier.

DTD : Définition de type de document. En langage XML, il s'agit généralement d'un fichier (ou de plusieurs fichiers utilisés ensemble) qui contient (contiennent) une définition formelle d'un type de document particulier. Elle spécifie les noms qui peuvent être utilisés pour les éléments inclus dans la DTD, les emplacements autorisés pour ces éléments au sein de la DTD et la manière dont les éléments s'accordent. Dans MQSeries Adapter Offering, vous pouvez utiliser les DTD pour définir les corps de messages. Reportez-vous aux sections «XML» à la page 139 et «Message d'intégration» à la page 138.

Fichier aqmconfig.xml : Pour plus d'informations, reportez-vous à la section «Fichier de configuration».

Fichier aqmsetup : Pour plus d'informations, reportez-vous à la section «Fichier de configuration».

Fichier de configuration : Fichier qui contient plusieurs configurations initiales du noyau. Le nom par défaut du fichier est `aqmsetup`.

Fichier de configuration : Il s'agit du fichier `aqmconfig.xml`, qui contient la plupart des valeurs de configuration du noyau. Pour plus d'informations, reportez-vous à la section «Le fichier de configuration» à la page 69.

File d'attente de réception : Il s'agit, dans la terminologie de MQSeries Adapter Offering, d'une file d'attente de messages utilisée comme file d'attente d'entrée principale pour la réception des messages. Il peut y avoir plusieurs files d'attente de réception par application cible, mais une seule file d'attente de réception pour chaque combinaison d'identificateur d'application, catégorie de corps et type de corps.

File d'attente des erreurs : Dans la terminologie de MQSeries Adapter Offering, il s'agit d'une file d'attente de messages qui est utilisée lorsqu'un message obtenu d'une file d'attente de réception ne peut pas être traité.

File d'attente des réponses : File d'attente de messages utilisée pour la réception des réponses. Elle est utilisée avec la méthode`sendRequestResponse` du noyau.

Format d'application neutre : Pour plus d'informations, reportez-vous à la section «Message d'intégration» à la page 138.

Identificateur d'application dépendante : Nom de l'application dont l'agent assure le service. L'agent obtient l'identificateur de l'application dépendante à partir du fichier de configuration basé sur le nom du démon de l'adaptateur.

Identificateur logique cible : Valeur représentant l'application cible associée à un adaptateur cible. Reportez-vous aux sections «Identificateur logique cible» et «Identificateur logique d'application».

Identificateur logique d'application : Identificateur qui représente l'application à laquelle un adaptateur (un adaptateur source ou un adaptateur cible) est associé. Reportez-vous aux sections «Identificateur logique source» à la page 138 et «Identificateur logique cible».

Identificateur logique de destination : Valeur qui représente l'application cible. Elle est utilisée, de même que les autres valeurs de contrôle de message, par le noyau pour l'acheminement et le classement des messages. Pour plus d'informations, reportez-vous à la section «Valeurs de contrôle de message» à la page 139.

Identificateur logique de réponse : Identificateur logique de l'application auquel les réponses doivent être envoyées lorsqu'une réponse est demandée. L'identificateur par défaut est l'identificateur logique source dans le message.

Identificateur logique source : Valeur qui représente l'application source. Elle est utilisée, de même que les autres valeurs de contrôle de message, par le noyau pour l'acheminement et le classement des messages. Reportez-vous aux sections «Valeurs de contrôle de message» à la page 139, «Identificateur logique d'application» à la page 137 et «Identificateur logique cible» à la page 137.

Interface spécifique de l'application : Interface externe à MQSeries Adapter Offering développée dans l'un des buts suivants :

- Pour permettre à l'adaptateur source d'acquérir un message en provenance de l'application source.
- Pour permettre à l'application cible d'acquérir un message en provenance de l'adaptateur cible.

JMS Listener : Composant fourni par le produit WebSphere Business Integrator qui permet une intégration étroite entre MQSeries Adapter Kernel et WebSphere Application Server Advanced Edition.

Message : Dans MQSeries, MQSeries Adapter Offering compris, série de données envoyées par un programme et destinées à un autre programme.

Message de communication : Toute information spécifique à l'acheminement des communications plus l'objet collecteur de message, convertie en un format de messagerie spécifique au type d'acheminement de communications utilisé.

Message d'intégration : Message composé de données d'application dans un format d'application neutre pour l'intégration. Il peut s'agir par exemple d'un document XML que l'adaptateur source transforme du format de l'application source au format XML.

Messages trace : Messages qui contiennent l'état de traitement d'un message à un moment donné dans le noyau. Vous pouvez utiliser les messages trace pour faciliter l'établissement des diagnostics concernant les problèmes affectant le noyau ou vos adaptateurs.

Reportez-vous à la section «Traçage» à la page 139.

Mode de communication : Il s'agit du mode utilisé par le noyau pour transporter le message et exécuter les services de courtage.

Modèle de communication par extraction : Pour plus d'informations, reportez-vous à la section «Modèles de communication».

Modèle de communication par insertion : Pour plus d'informations, reportez-vous à la section «Modèles de communication».

Modèles de communication : L'interface du noyau avec l'application cible est assurée par deux modèles. Ces deux modèles sont les suivants :

par insertion
C'est le noyau qui assure le lancement et la gestion de l'acheminement du message vers l'application cible. Ce modèle ne nécessite généralement pas la modification de l'application cible pour la prise en charge de MQSeries Adapter Offering.

par extraction
C'est l'application cible qui assure la gestion de l'acheminement du message. Ce modèle nécessite la modification de l'application cible pour la prise en charge de MQSeries Adapter Offering. L'application cible doit gérer l'interface du noyau avec l'application cible.

MQSeries Adapter Builder : Logiciel permettant à l'utilisateur de créer un adaptateur pour pratiquement toutes les applications à l'aide d'une interface graphique utilisateur (GUI).

MQSeries Adapter Kernel : Ensemble d'API, de plusieurs programmes exécutables, en C et Java, et de plusieurs fichiers de configuration. Le noyau fonctionne avec les adaptateurs et les prend en charge. Reportez-vous à la section «Adaptateur» à la page 135. Outre la prise en charge directe des adaptateurs, le noyau exécute les fonctions associées dont, parmi les plus importantes, l'acheminement des messages et les services d'infrastructure comme la création de messages, le traçage et l'interfaçage avec MQSeries ou un autre logiciel de messagerie.

MQSeries Adapter Offering : Ensemble de produits d'intégration d'application, composé de MQSeries Adapter Builder et de MQSeries Adapter Kernel.

Noyau : Synonyme de MQSeries Adapter Kernel.

OAG : Open Applications Group. Consortium industriel non lucratif composé de nombreux acteurs importants dans le domaine de l'interopérabilité des composants de logiciels de gestion. L'OAG définit les Documents Objets de Gestion (BOD).

Objet collecteur de messages : Contenant pour les métadonnées utilisées par le noyau pour encapsuler un message d'intégration et d'autres données de contrôle.

Service de messages logiques : Composant utilisé par l'adaptateur natif pour convertir les messages en vue de leur acheminement par le transfert de communications.

Traçage : Série de processus utilisés par le noyau pour écrire des messages trace. Pour plus d'informations, reportez-vous à la section «Messages trace» à la page 138.

Transaction : Série d'opérations qui doivent être exécutées en tant qu'unité indivisible de travail. Si toutes les opérations comprenant une transaction sont réussies, la transaction est engagée, c'est-à-dire que toutes les opérations sont exécutées. Si une ou plusieurs opérations contenant une transaction échoue(nt), la transaction est annulée, c'est-à-dire qu'aucune des opérations n'est exécutée.

Type de corps : Donnée contenue dans un message qui représente l'objet spécifique du message, par exemple, ajout du bordereau de ventes ou synchronisation de l'inventaire. Elle appartient à la série des valeurs de contrôle du message. Pour plus d'informations, reportez-vous à la section «Valeurs de contrôle de message».

Le type de corps aide également à déterminer le type de message. Reportez-vous à la section «Type de message».

Type de message : Message caractérisé par une combinaison unique de catégorie de corps et de type de corps. Reportez-vous aux sections «Catégorie de corps» à la page 136 et «Type de corps».

Valeurs de contrôle de message : Terme collectif désignant un ensemble de valeurs dans les messages (corps et en-têtes) et dans le fichier de configuration que le noyau utilise pour contrôler le classement et l'acheminement des messages et que chaque adaptateur utilise pour contrôler, en partie, la manière dont il exécute sa fonction.

WebSphere Application Server Advanced Edition : Progiciel IBM qui permet l'utilisation de la spécification EJB (Enterprise JavaBeans) de Sun Microsystems. WebSphere Application Server Advanced Edition comprend un serveur EJB, dans lequel les beans d'entreprise peuvent être exécutés. Ces beans regroupent la logique applicative et les données utilisées et partagées par les clients EJB. Il en existe deux types : les beans de session, qui encapsulent des tâches et des objets propres à un client et de courte durée et les beans d'entité, qui encapsulent des données rémanentes. Un type de bean de session appelé bean de message agent peut être utilisé du côté cible de MQSeries Adapter Kernel.

XML : Extensible Markup Language. Norme W3C relative à la représentation des données.

Index

H

I

J

L

M

N

O

P

R

S

T

U

V

W

X

GC11-1809-01