Notes de version de Sun Java System Message Queue 4.1

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Référence : 820–3190–10
Septembre 2007

071001@18741

Table des matières

◆ ◆ ◆ CHAPITRE 1

Notes de version de Sun Java System Message Queue 4.1

Version 4.1

Numéro de référence 820-3190-10

Ces notes de version contiennent des informations importantes, disponibles au moment de la commercialisation de Sun Java™ System Message Queue 4.1. Vous y trouverez des renseignements sur les nouvelles fonctionnalités, les améliorations, les restrictions et problèmes connus, etc. Lisez ce document avant d'utiliser Message Queue. Ces notes de version contiennent également des informations sur la version 4.0 de Message Queue ; reportez-vous à la section “À propos de Message Queue 4.0” à la page 17 pour découvrir les nouvelles fonctionnalités de cette version.

Vous trouverez la dernière version de ces notes de version sur le site Web de documentation de Sun Java System Message Queue. Consultez ce site Web avant d'installer et de configurer votre logiciel, puis consultez-le régulièrement pour vous procurer la documentation concernant le produit et les notes de version les plus récentes.

Ces notes de version se composent des sections suivantes :

Des URL de sites tiers, qui renvoient à des informations complémentaires connexes, sont référencés dans ce document.

Sun ne peut être tenu responsable de la disponibilité des sites Web des tiers qui sont mentionnés dans le présent document. Sun ne garantit pas le contenu, la publicité, les produits et autres documents disponibles sur ces sites ou dans ces ressources, ou accessibles par leur intermédiaire, et ne saurait en être tenu pour responsable. Sun ne pourra en aucun cas être tenu responsable, directement ou indirectement, de tous dommages ou pertes, réels ou invoqués, causés par ou liés à l'utilisation des contenus, biens ou services disponibles dans ou par l' intermédiaire de ces sites ou ressources.

Historique de révision des notes de version

Le tableau suivant répertorie les dates correspondant à toutes les versions 4.x du produit Message Queue et décrit les principales modifications associées à chaque version.

TABLEAU 1–1 Historique des révisions

Date	Description des modifications
Mai 2006	Version initiale du présent document pour la version 4.0 de Message Queue.
Janvier 2007	Version initiale du présent document pour la version Bêta 4.1 de Message Queue. Ajout d'une description concernant la prise en charge de JAAS.
Avril 2007	Deuxième version du présent document pour la version Bêta 4.1 de Message Queue. Ajout de la fonctionnalité de haute disponibilité.
Septembre 2007	Troisième version du présent document destinée aux clients. Ajout d'une description concernant la structure de contrôle Java Enterprise System, les ports C fixes, les résolutions de bogue et autres fonctionnalités.

À propos de Message Queue 4.1

Sun Java System Message Queue est un service de messagerie complet offrant des fonctionnalités de messagerie asynchrones et fiables conformes à la spécification de messagerie Java (JMS) 1.1. En outre, Message Queue propose des fonctionnalités supplémentaires par rapport à la spécification JMS pour répondre aux besoins des déploiements d'entreprise à grande échelle.

La version 4.1 de Message Queue offre la prise en charge des éléments suivants : haute disponibilité, JAAS (Java Authentication and Authorization Service), ports C fixes et structure de contrôle Java Enterprise System. Elle propose également quelques améliorations, ainsi que des résolutions de bogue. Cette section comprend les rubriques suivantes :

Pour obtenir des informations sur les nouvelles fonctionnalités de Message Queue 4.0, reportez-vous à la section “À propos de Message Queue 4.0” à la page 17.

Nouveautés de la version 4.1

Message Queue 4.1 propose des clusters de courtier haute disponibilité (données et services), la prise en charge de JAAS et d'autres fonctionnalités mineures. Cette section décrit ces nouvelles fonctionnalités et fournit des références supplémentaires pour votre information.

Haute disponibilité

Message Queue 4.1 est équipé de clusters haute disponibilité fournissant la disponibilité des données et des services. Si un client perd sa connexion à un courtier haute disponibilité, celui-ci est automatiquement reconnecté à un autre courtier dans un cluster. Le courtier offrant la nouvelle connexion récupère les données et l'état du courtier en échec et fournit un service continu aux clients de ce courtier. Vous pouvez exécuter des courtiers haute disponibilité sur une connexion sécurisée.

Ceux-ci requièrent l'utilisation d'une base de données hautement disponible (HADB). Si vous ne disposez pas de ce type de base de données ou si vous ne jugez pas la disponibilité des données primordiale, vous pouvez continuer à utiliser des clusters conventionnels, offrant une reconnexion automatique et la disponibilité des services.

La configuration des courtiers haute disponibilité est simple : il vous suffit de spécifier les propriétés suivantes pour chaque courtier du cluster.

- *Propriétés d'appartenance au cluster :* spécifient le courtier comme partie d'un cluster haute disponibilité, l'ID du cluster et l'ID du courtier.
- *Propriétés de la base de données hautement disponible (HADB)* : spécifient le modèle pour les messages persistants (JDBC), le nom du fournisseur de HADB et les propriétés de configuration spécifiques au fournisseur pour la base de données.
- *Propriétés de détection d'erreurs et de reprise* : spécifient le mode de détection et de réparation des erreurs du courtier.

Pour utiliser cette fonctionnalité, procédez comme suit :

1. Installez une base de données hautement disponible.
2. Installez le fichier .jar du pilote JDBC.
3. Créez le schéma de base de données pour le magasin persistant hautement disponible.
4. Définissez les propriétés associées à la fonction de haute disponibilité pour chaque courtier du cluster.
5. Démarrez chaque courtier du cluster.

Si vous souhaitez consulter une discussion conceptuelle sur la haute disponibilité et obtenir une comparaison de celle-ci par rapport aux clusters conventionnels, reportez-vous au Chapitre 4, “Broker Clusters” du *Sun Java System Message Queue 4.1 Technical Overview*. Pour obtenir des informations procédurales et référentielles sur la haute disponibilité, reportez-vous au Chapitre 8, “Broker Clusters” du *Sun Java System Message Queue 4.1 Administration Guide* et à la section “Cluster Configuration Properties” du *Sun Java System Message Queue 4.1 Administration Guide*.

Si vous utilisiez une base de données HADB avec Message Queue version 4.0 et que vous souhaitez utiliser un cluster haute disponiblité, vous pouvez utiliser l'utilitaire `dbmgr` pour procéder à une mise à niveau vers un magasin HADB partagé. Consultez la section “Clusters de courtier” à la page 46 pour plus d'informations.

Prise en charge de JAAS

En plus des mécanismes d'authentification intégrés basés sur les fichiers et le LDAP, Message Queue prend également en charge le JAAS (Java Authentication and Authorization Service), qui permet de connecter divers services au courtier pour authentifier les clients Message Queue. Cette section présente les informations mises à disposition par le courtier à un service d'authentification compatible JAAS et définit la procédure de configuration du courtier pour utiliser ce type de service.

En revanche, l'API JAAS n'est pas décrite dans le présent document. Consultez les sources suivantes si vous avez besoin d'informations détaillées.

- Pour obtenir une description complète de l'API JAAS, reportez-vous au manuel *Java Authentication and Authorization Service (JAAS) Reference Guide*.

  `http://java.sun.com/j2se/1.5.0/docs/guide/security/jaas/JAASRefGuide.html`

- Pour obtenir des informations sur l'écriture d'un module de connexion, reportez-vous au manuel *Java Authentication and Authorization Service (JAAS) LoginModule Developer's Guide*.

  `http://java.sun.com/j2se/1.5.0/docs/guide/security/jaas/JAASLMDevGuide.html`

L'API JAAS est une API noyau dans J2SE et fait donc partie intégrante de l'environnement d'exécution de Message Queue. JAAS définit une couche d'abstraction entre une application et un mécanisme d'authentification, permettant ainsi de connecter le mécanisme souhaité sans modifier le code d'application. En ce qui concerne le service Message Queue, la couche

d'abstraction se trouve entre le courtier (application) et un fournisseur d'authentification. En définissant certaines propriétés du courtier, vous avez la possibilité de connecter n'importe quel service d'authentification compatible JAAS et de mettre à niveau ce dernier sans interruption ou modification du code du courtier.

Vous pouvez utiliser des clients JMX pour gérer le courtier si vous utilisez un service d'authentification JAAS ; en revanche, vous devez configurer manuellement la prise en charge de JAAS (en définissant les propriétés du courtier correspondantes) avant de démarrer le courtier. Vous ne pouvez pas utiliser l'API JMX pour modifier ces propriétés.

Éléments du JAAS

Figure 1–1 : présente les éléments de base du JAAS : un client JAAS, un service d'authentification compatible JAAS et un fichier de configuration JAAS.

- Le client JAAS est une application visant à effectuer une authentification à l'aide d'un service d'authentification compatible JAAS. Il communique avec ce service à l'aide d'un module de connexion et est chargé de fournir un gestionnaire de rappels pouvant être appelé par le module de connexion afin d'obtenir le nom d'utilisateur, le mot de passe et toute autre information pertinente.
- Le service d'authentification compatible JAAS représente un ou plusieurs modules de connexion et une logique exécutant l'authentification requise. Le module de connexion peut inclure la logique d'authentification ou peut utiliser un protocole privé ou une API pour communiquer avec un module fournissant cette logique.
- Le fichier de configuration JAAS est un fichier texte utilisé par le client JAAS pour localiser le ou les modules de connexion requis pour communiquer avec le service compatible JAAS.

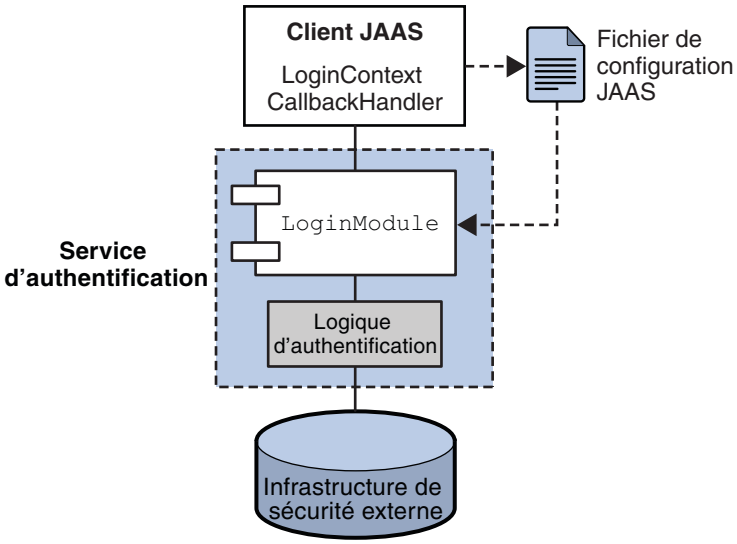

FIGURE 1–1 Éléments du JAAS

La section suivante décrit de quelle manière le service Message Queue utilise ces éléments pour fournir une authentification compatible JAAS.

JAAS et Message Queue

La figure suivante montre de quelle manière JAAS est utilisé par le courtier Message Queue. Elle illustre une implémentation plus complexe du modèle JAAS représenté sur la figure précédente.

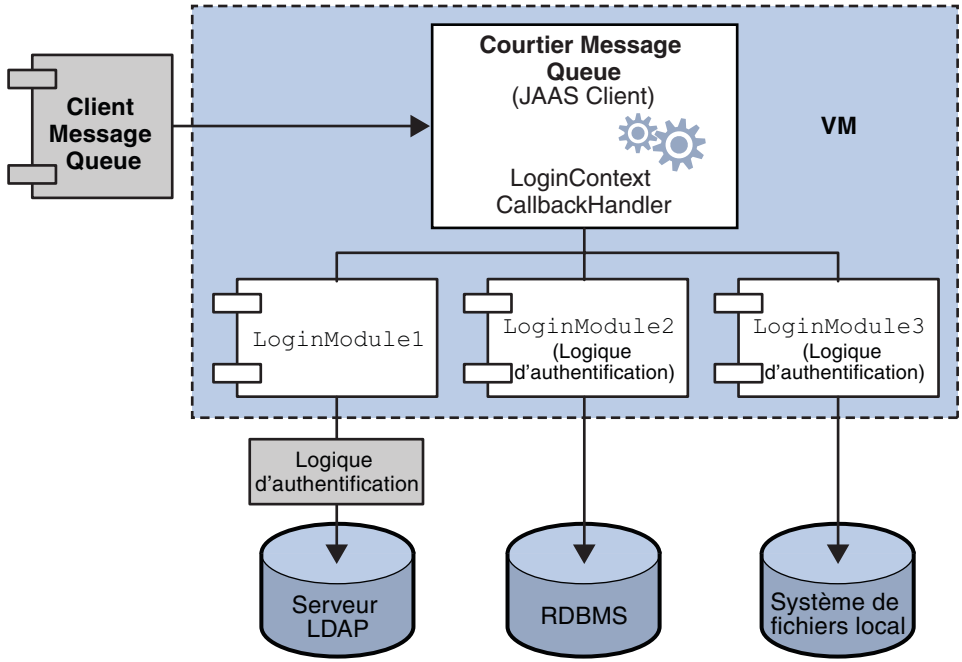

FIGURE 1–2 Comment Message Queue utilise JAAS

Comme le montrait la figure simplifiée, la couche du service d'authentification est séparée du courtier. Le service d'authentification comprend un ou plusieurs modules de connexion (LoginModule), ainsi que des modules d'authentification supplémentaires si nécessaire. Les modules de connexion s'exécutent sur la même machine virtuelle Java que le courtier. Le courtier Message Queue apparaît au module de connexion sous forme de `LogInContext` et communique avec celui-ci au moyen d'un `CallBackHandler` , faisant partie du code d'exécution du courtier.

Le service d'authentification fournit également un fichier de configuration JAAS comprenant des entrées vers les modules de connexion. Le fichier de configuration spécifie l'ordre d'utilisation des modules ainsi que certaines conditions d'utilisation. Lorsque le courtier démarre, JAAS localise le fichier de configuration à l'aide de la propriété système Java `java.security.auth.login.config` ou du fichier de propriétés de sécurité Java. Il sélectionne ensuite une entrée dans le fichier de configuration JAAS, selon la valeur de la propriété du courtier `imq.user_repository.jaas.name` . Cette entrée spécifie quels modules de connexion seront utilisés pour l'authentification. Comme le montre la figure, le courtier peut utiliser plusieurs modules de connexion. (La relation existant entre le fichier de configuration, le module de connexion et le courtier est représentée sur la Figure 1–3.)

Le fait que le courtier utilise un service d'authentification de plug-in JAAS reste entièrement transparent pour le client Message Queue. Ce dernier continue de se connecter au courtier comme avant, à l'aide d'un nom d'utilisateur et d'un mot de passe. En réponse, le courtier utilise

un gestionnaire de rappels pour transmettre ces informations au service d'authentification et ce dernier utilise ces données pour authentifier l'utilisateur et retourner les résultats correspondants. Si l'authentification réussit, le courtier autorise la connexion, sinon l'exécution client retourne une exception de sécurité JMS devant être gérée par ce dernier.

Une fois le client Message Queue authentifié, si une autorisation plus complexe est nécessaire, le courtier reprend une activité normale ; il consulte alors le fichier de contrôle d'accès pour déterminer si le client authentifié est autorisé à exécuter les actions suivantes : accéder à une destination, consommer un message, parcourir une file d'attente, etc.

Configuration de l'authentification compatible JAAS

La configuration de l'authentification compatible JAAS implique la définition des propriétés du système et du courtier pour pouvoir sélectionner ce type d'authentification, spécifier l'emplacement du fichier de configuration et spécifier les entrées vers les modules de connexion qui seront utilisés.

Cette section illustre les liens existant entre le client JAAS, les modules de connexion et le fichier de configuration JAAS, puis décrit le processus requis pour configurer une authentification compatible JAAS. La figure suivante illustre la relation existant entre le fichier de configuration, le module de connexion et le courtier.

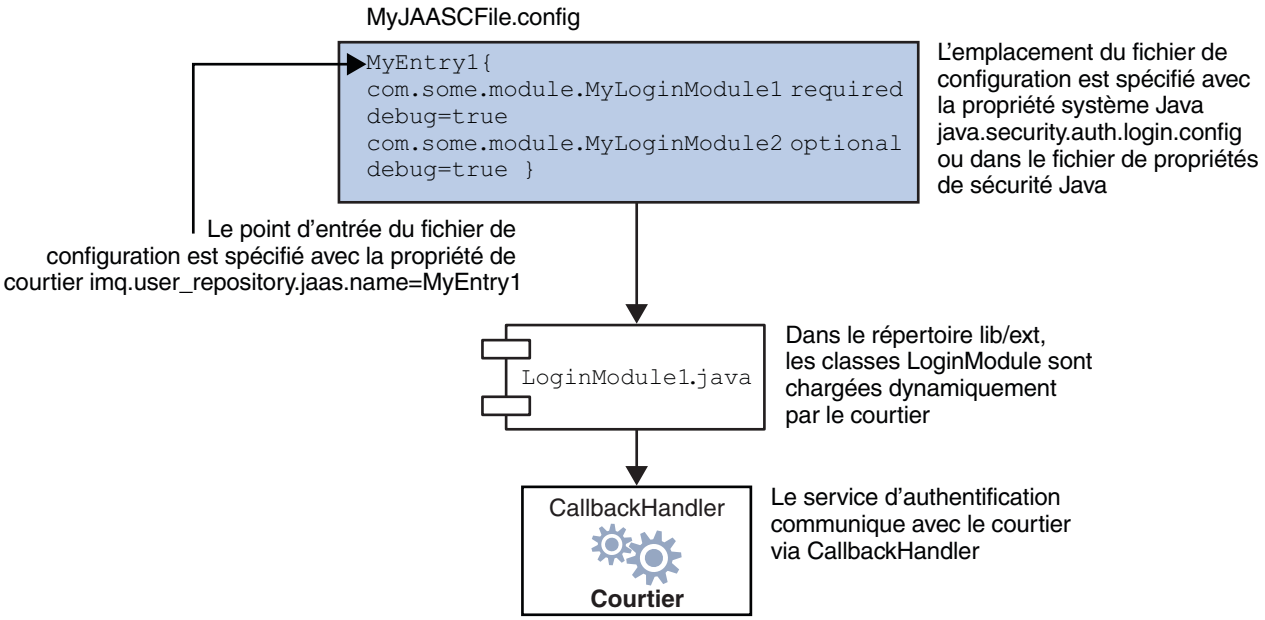

FIGURE 1–3 Configuration de la prise en charge de JAAS

Comme indiqué, le fichier de configuration JAAS, `MyJAASCFile.config`, contient des références à plusieurs modules de connexion, regroupés dans un point d'entrée. Le courtier localise le fichier de configuration en consultant la propriété système Java `java.security.auth.login.config` ou le fichier de propriétés de sécurité Java. Les modules de connexion à utiliser sont déterminés par la consultation de la propriété du courtier `imq.user_repository.jaas.name` , spécifiant l'entrée souhaitée dans le fichier de configuration. Les classes de ces modules se trouvent dans le répertoire `lib/ext`.

Afin de configurer la prise en charge de JAAS pour Message Queue, vous devez suivre les étapes suivantes. (Dans un environnement de développement, l'ensemble de ces étapes peut être pris en charge par le développeur. Dans un environnement de production, l'administrateur peut effectuer certaines de ces tâches.)

1. Créez une ou plusieurs classes de module de connexion chargées d'implémenter le service d'authentification. Les types de rappel JAAS pris en charge par le courtier sont répertoriés ci-dessous.

   `javax.security.auth.callback.LanguageCallback`
   Le courtier utilise ce rappel pour transmettre au service d'authentification la langue dans laquelle il est exécuté. Cette valeur peut être utilisée pour la localisation.

   `javax.security.auth.callback.NameCallback`
   Le courtier utilise ce rappel pour transmettre au service d'authentification le nom d'utilisateur spécifié par le client Message Queue à la demande de connexion.

   `javax.security.auth.callback.TextInputCallback`
   Le courtier utilise ce rappel pour spécifier la valeur de `imq.authentication.type` au service d'authentification lorsque `TextInputCallback.getPrompt()` est `imq.authentication.type`. Pour l'instant, la seule valeur possible pour ce champ est `basic`. Celle-ci indique un codage de mot de passe en Base64.

   `javax.security.auth.callback.PasswordCallback`
   Le courtier utilise ce rappel pour transmettre au service d'authentification le mot de passe spécifié par le client Message Queue à la demande de connexion.

   `javax.security.auth.callback.TextOutputCallback`
   Le courtier utilise ce rappel pour fournir des services de journalisation au service d'authentification en consignant la sortie texte vers son propre fichier journal. Les paramètres MessageType de rappel `ERROR, INFORMATION, WARNING` sont mappés respectivement vers les niveaux de journal du courtier `ERROR, INFO,` et `WARNING`.

2. Créez un fichier de configuration JAAS avec des entrées faisant référence aux classes de module de connexion et spécifiez l'emplacement de ce fichier à l'administrateur Message Queue. (Ce fichier peut être stocké à distance et son emplacement peut être spécifié à l'aide d'un URL.)
3. Notez le nom de l'entrée (faisant référence aux classes d'implémentation de connexion) dans le fichier de configuration JAAS.

4. Archivez ces classes dans un fichier jar, puis stockez ce dernier dans le répertoire Message Queue `lib/ext`.
5. Configurez les propriétés du courtier associées à la prise en charge de JAAS. Celles-ci sont présentées dans le Tableau 1–2.
6. Définissez la propriété système suivante pour spécifier l'emplacement du fichier de configuration JAAS.

   `java.security.auth.login.config=` *Emplacement_Fichier_Config_JAAS*

   Par exemple, vous pouvez spécifier le fichier de configuration au démarrage du courtier.

   `imqbrokerd -Djava.security.auth.login.config=`*Emplacement_Fichier_Config_JAAS*

   Il existe également d'autres procédures pour spécifier l'emplacement du fichier de configuration JAAS. Pour de plus amples informations, consultez :

   http://java.sun.com/j2se/.1.5.0/docs/guide/security/jaas/tutorials/LoginConfigFile.html

Le tableau suivant répertorie les propriétés du courtier requises pour configurer la prise en charge de JAAS.

TABLEAU 1–2 Propriétés du courtier pour la prise en charge de JAAS

Propriété	Description
`imq.authentication.type`	Définie sur `basic` pour indiquer un codage de mot de passe Base64. Il s'agit de l'unique valeur autorisée pour une authentification JAAS.
`imq.authentication.basic.user_repository`	Définie sur `jaas` pour spécifier une authentification JAAS.
`imq.accesscontrol.type`	Définie sur `file`.
`imq.user_repository.jaas.name`	Définie sur le nom de l'entrée souhaitée (dans le fichier de configuration JAAS), faisant référence aux modules de connexion choisis comme mécanisme d'authentification. Il s'agit du nom spécifié à l'étape 3.
`imq.user_repository.jaas.userPrincipalClass`	Cette propriété, utilisée par le contrôle d'accès Message Queue, spécifie la classe d'implémentation `java.security.Principal` dans le ou les modules utilisés par le courtier pour extraire le nom principal destiné à représenter l'entité utilisateur dans le fichier de contrôle d'accès Message Queue. Si celle-ci n'est pas spécifiée, le nom d'utilisateur transmis par le client Message Queue à la demande de connexion est utilisé.

TABLEAU 1–2 Propriétés du courtier pour la prise en charge de JAAS *(Suite)*

Propriété	Description
`imq.user_repository.jaas.groupPrincipalClass`	Cette propriété, utilisée par le contrôle d'accès Message Queue, spécifie la classe d'implémentation `java.security.Principal` dans le ou les modules de connexion utilisés par le courtier pour extraire le nom principal destiné à représenter l'entité groupe dans le fichier de contrôle d'accès Message Queue. Si celle-ci n'est pas spécifiée, les règles de groupe sont ignorées dans le fichier de contrôle d'accès Message Queue, le cas échéant.

Modification du format du magasin persistant

La version 4.1 de Message Queue modifie le magasin JDBC pour prendre en charge la fonction de haute disponibilité. La version du magasin JDBC a donc été portée à 410. Les versions 350, 370 et 400 sont automatiquement migrées vers le format de la version 410.

Notez toutefois que la version du magasin persistant basé sur les fichiers demeure à 370 car celui-ci n'a fait l'objet d'aucune modification.

Configuration du courtier

La propriété `IMQ_DEFAULT_EXT_JARS` a été ajoutée au fichier `imqenv.conf`. Vous pouvez définir cette propriété pour spécifier les noms de chemin des fichiers .jar externes à inclure dans `CLASSPATH` au démarrage du courtier. Si vous utilisez cette propriété pour spécifier l'emplacement des fichiers `.jar` externes, il ne vous sera plus nécessaire de copier ces fichiers dans le répertoire `lib/ext`. Ces fichiers peuvent faire référence aux pilotes JDBC ou aux modules de connexion JAAS. L'exemple de commande suivant spécifie l'emplacement des pilotes JDBC.

```
IMQ_DEFAULT_EXT_JARS=/opt/SUNWhadb4/lib/hadbjdbc4.jar:/opt/SUNWjavadb/derby.jar
```

Prise en charge de la structure de contrôle JES

Message Queue prend en charge la structure de contrôle Sun Java Enterprise System (JES), permettant un contrôle des composants Java Enterprise System à l'aide d'une interface graphique standard. Cette interface est implémentée par une console basée sur le Web, Sun Java System Monitoring Console. Si vous exécutez Message Queue en même temps que d'autres composants JES, vous trouverez peut-être plus pratique d'utiliser une seule interface pour la gestion de tous ces composants.

La structure de contrôle JES définit un modèle de données commun (CMM) pouvant être utilisé par tous les composants JES. Ce modèle propose un aperçu centralisé et uniforme de tous les composants JES. Message Queue expose les objets suivants à la structure de contrôle JES :

- le produit installé ;

- le nom de l'instance du courtier ;
- le mappeur de ports du courtier ;
- chaque service de connexions ;
- chaque destination physique ;
- le magasin persistant ;
- le référentiel utilisateur.

Chacun de ces objets est mappé vers un objet CMM dont les attributs peuvent être contrôlés à l'aide de la console de contrôle JES. Au cours de l'exécution, les administrateurs peuvent utiliser la console pour afficher les statistiques de performance, créer des règles de contrôle automatique et accuser réception des alarmes. Pour de plus amples informations sur le mappage des objets Message Queue vers les objets CMM, reportez-vous au manuel *Sun Java Enterprise System Monitoring Guide*.

Pour activer le contrôle JES, procédez comme suit :

1. Installez et configurez tous les composants de votre déploiement (Message Queue et autres) en suivant les instructions données dans le *Guide d'installation de Sun Java Enterprise System*.
2. Activez et configurez la structure de contrôle pour l'ensemble de vos composants sous contrôle, comme décrit dans le manuel *Sun Java Enterprise System Monitoring Guide*.
3. Installez la console de contrôle sur un hôte séparé, démarrez l'agent maître, puis le serveur Web, comme décrit dans le manuel *Sun Java Enterprise System Monitoring Guide*.

L'utilisation de la structure de contrôle JES n'a aucun effet sur les performances du courtier car le travail de rassemblement des mesures est effectué par la structure de contrôle, qui récupère les données à partir de l'infrastructure de données de contrôle existante du courtier.

Gestion des transactions

Auparavant, seules les transactions en état `PREPARED` pouvaient être annulées administrativement. Par exemple, lorsqu'une session, faisant partie d'une transaction distribuée, ne s'arrêtait pas normalement, la transaction était conservée dans un état qui ne pouvait pas être nettoyé par l'administrateur du courtier. Dans Message Queue 4.1, vous pouvez utiliser l'utilitaire imqcmd pour nettoyer (annuler) les transactions se trouvant dans les états suivants : `STARTED`, `FAILED`, `INCOMPLETE`, `COMPLETE`, `PREPARED`.

Pour vous aider à déterminer si une transaction particulière peut être annulée ou pas (en particulier lorsque celle-ci n'est pas en état `PREPARED`), l'utilitaire imqcmd fournit des données supplémentaires dans la sortie `imqcmd query txn` : il fournit l'ID de connexion pour la connexion ayant démarré la transaction et spécifie l'heure de création de cette dernière. À l'aide de ces informations, l'administrateur peut alors décider d'annuler ou pas la transaction. En règle générale, il est préférable que l'administrateur évite d'annuler une transaction prématurément.

Ports fixes pour les connexions de clients C

Les clients C peuvent utiliser la propriété de connexion `MQ_SERVICE_PORT_PROPERTY` pour spécifier un port fixe auquel se connecter. Cela peut s'avérer pratique si vous essayez de passer au travers d'un pare-feu ou si vous avez besoin de contourner le service de mappage de ports du courtier (chargé d'assigner les ports dynamiquement).

Veillez à configurer le port du service JMS également du côté courtier. Par exemple, si vous souhaitez connecter votre client via `ssljms` au port 1756, il vous faut procéder comme suit.

- Du côté client : Définissez `MQ_SERVICE_PORT_PROPERTY` sur 1756 et `MQ_CONNECTION_TYPE_PROPERTY` sur `SSL`.
- Du côté courtier : Définissez la propriété `imq.`*serviceNameType.protocol* `.port` sur 1756 comme suit.

```
imq.ssljms.ssl.port=1756
```

Remarque – La propriété de connexion `MQ_SERVICE_PORT_PROPERTY` a été introduite avec la version 3.7 Update 2 de Message Queue.

Configurations matérielle et logicielle requises

Pour obtenir les configurations matérielle et logicielle requises pour la version 4.1, consultez le manuel *Sun Java System Message Queue 4.1 Installation Guide* .

À propos de Message Queue 4.0

Message Queue 4.0 vise essentiellement à prendre en charge Application Server 9 PE. Il s'agit d'une version mineure comportant de nouvelles fonctionnalités, certaines améliorations et des résolutions de bogue. Cette section comprend les rubriques suivantes :

- “Nouveautés de la version 4.0” à la page 17
- “ Configurations matérielle et logicielle requises” à la page 33

Nouveautés de la version 4.0

Message Queue 4.0 propose les nouvelles fonctionnalités suivantes :

- “Modifications de l'interface pour l'exécution de l'API et du client C” à la page 18
- “Modifications de l'interface pour l'exécution de l'API et du client Java” à la page 18
- “Affichage d'informations à propos du magasin persistant” à la page 18
- “Modifications du format du magasin persistant” à la page 19

- “Administration du courtier” à la page 20
- “Prise en charge de la persistance JDBC” à la page 21
- “Prise en charge de SSL” à la page 21
- “Prise en charge de JMX” à la page 21
- “Journalisation à l'exécution client” à la page 26
- “Notification d'événements de connexion” à la page 31

Ces fonctions sont décrites dans les sous-sections suivantes.

Attention – Parmi les modifications mineures mais radicales de la version 4.0, on peut souligner la désapprobation de l'option de ligne de commande pour spécifier un mot de passe. Par conséquent, vous devez stocker tous vos mots de passe dans un fichier, comme décrit à la section “Option de mot de passe désapprouvée” à la page 43.

Modifications de l'interface pour l'exécution de l'API et du client C

La version 4.0 de Message Queue propose deux nouvelles propriétés qui s'appliqueront à tous les messages placés dans la file d'attente des messages bloqués.

- `JMS_SUN_DMQ_PRODUCING_BROKER` indique le courtier dans lequel le message a été créé.
- `JMS_SUN_DMQ_DEAD_BROKER` indique le courtier ayant marqué le message comme bloqué.

Modifications de l'interface pour l'exécution de l'API et du client Java

La version 4.0 de Message Queue propose deux nouvelles propriétés qui s'appliqueront à tous les messages placés dans la file d'attente des messages bloqués.

- `JMS_SUN_DMQ_PRODUCING_BROKER` indique le courtier dans lequel le message a été créé.
- `JMS_SUN_DMQ_DEAD_BROKER` indique le courtier ayant marqué le message comme bloqué.

Affichage d'informations à propos du magasin persistant

La sous-commande `query` a été ajoutée à la commande `imqdbmgr`. Utilisez cette sous-commande pour afficher les informations relatives au magasin persistant, notamment sa version, l'utilisateur de la base de données et la création ou non de tables de base de données.

Voici un exemple des informations affichées par la commande.

```
requête imqdbmgr

[04/Oct/2005:15:30:20 PDT] Utilisation du magasin persistant intégré :
        version=400
        id du courtier=Mozart1756
        url de connexion à la base de données=jdbc:oracle:thin:@Xhome:1521:mqdb
        utilisateur de la base de données=scott
```

```
Exécution en mode autonome.
Les tables de base de données ont déjà été créées.
```

Modifications du format du magasin persistant

La version 3.7 UR1 de Message Queue comporte deux modifications du format du magasin persistant en vue d'améliorer les performances. La première modification a été apportée au magasin de fichiers et la deuxième au magasin JDBC.

- Format des données de transaction persistantes dans le magasin de fichiers

  Le format des informations d'état de transaction stockées dans le magasin persistant basé sur les fichiers de Message Queue a été modifié pour réduire l'E/S du disque et pour améliorer les performances des transactions JMS.

- Magasin JDBC Oracle

  Dans les anciennes versions de Message Queue, le schéma de magasin pour Oracle utilisait le type de données `LONG RAW` pour stocker les données de messages. Dans Oracle 8, le type `LONG RAW` a fait place aux types de données `BLOB`. Message Queue 3.7 UR1 est passé au type de données `BLOB` en vue d'améliorer les performances et les capacités de prise en charge.

Étant donné que ces modifications influent sur la compatibilité des magasins, la version du magasin de fichiers et du magasin JDBC a été modifiée de 350 à 370 dans la version 3.7 UR1 de Message Queue.

La version 4.0 de Message Queue présente des modifications sur le magasin JDBC pour une optimisation et la prise en charge des prochaines améliorations. C'est pourquoi la version du magasin JDBC a été portée à 400. Notez que dans la version 4.0, la version du magasin persistant basé sur les fichiers demeure à 370 car celui-ci n'a pas été modifié.

Message Queue 4.0 prend en charge la conversion automatique du magasin persistant vers les versions les plus récentes du magasin basé sur les fichiers et du magasin JDBC. Au premier démarrage de `imqbrokerd`, si l'utilitaire détecte un ancien magasin, celui-ci sera migré vers le nouveau format, en abandonnant l'ancienne version.

- Les versions 200 et 350 du magasin basé sur les fichiers seront migrées vers le format de la version 370.
- Les versions 350 et 370 du magasin JDBC seront migrées vers le format de la version 400. (Si vous souhaitez mettre à niveau un magasin 200, il vous faudra passer par une version intermédiaire 3.5 ou 3.6.)

Si vous souhaitez annuler cette mise à niveau, vous pouvez désinstaller Message Queue 4.0, puis réinstallez la version que vous utilisiez précédemment. Étant donné que l'ancienne copie du magasin est conservée, le courtier peut l'exécuter.

Administration du courtier

L'utilitaire Command (`imqcmd`) fournit une nouvelle sous-commande et de nouvelles options permettant aux administrateurs de mettre le courtier en attente, de fermer le courtier après un intervalle spécifié, de détruire une connexion ou de définir des propriétés système Java (par exemple, les propriétés liées à la connexion.)

- La mise en attente du courtier place ce dernier dans un état silencieux, permettant ainsi une purge des messages avant la fermeture ou le redémarrage de celui-ci. Il est impossible de créer une nouvelle connexion vers un courtier mis en attente. Pour mettre le courtier en attente, saisissez une commande similaire à l'exemple suivant.

  ```
  imqcmd quiesce bkr -b Wolfgang:1756
  ```

- Pour fermer le courtier après un intervalle spécifié, saisissez une commande similaire à l'exemple ci-dessous. (L'intervalle de temps spécifie le nombre de secondes à attendre avant la fermeture du courtier.)

  ```
  imqcmd shutdown bkr -b Hastings:1066 -time 90
  ```

  Si vous spécifiez un intervalle de temps, le courtier journalisera un message indiquant l'heure de fermeture. Par exemple,

  ```
  Shutting down the broker in 29 seconds (29996 milliseconds)
  ```

  Alors que le courtier est en attente de fermeture, son comportement est affecté de diverses manières :

  - Les connexions JMS administratives sont encore acceptées.
  - Aucune nouvelle connexion JMS n'est acceptée.
  - Les connexions JMS existantes continuent de fonctionner.
  - Le courtier n'est pas en mesure de remplacer un autre courtier dans un cluster haute disponibilité.
  - L'utilitaire imqcmd ne s'interrompt pas, il envoie la requête de fermeture au courtier et reprend directement une activité normale.

- Pour détruire une connexion, saisissez une commande similaire à l'exemple suivant.

  ```
  imqcmd destroy cxn -n 2691475382197166336
  ```

  Utilisez la commande `imqcmd list cxn` ou `imqcmd query cxn` pour obtenir l'ID de connexion.

- Pour définir une propriété système à l'aide de `imqcmd`, utilisez l'option –D Ceci s'avère pratique pour définir ou ignorer des propriétés de fabrique de connexion JMS ou des propriétés système Java liées à la connexion. Par exemple :

  ```
  imqcmd list svc -secure -DimqSSLIsHostTrusted=true
  imqcmd list svc -secure -Djavax.net.ssl.trustStore=/tmp/mytruststore
              -Djavax.net.ssl.trustStorePassword=mytrustword
  ```

Pour de plus amples informations sur la syntaxe de la commande `imqcmd`, reportez-vous au Chapitre 13, “Command Line Reference” du *Sun Java System Message Queue 4.1 Administration Guide*.

Prise en charge de la persistance JDBC

Apache Derby Version 10.1.1 est désormais pris en charge en tant que fournisseur de magasin persistant compatible JDBC.

Prise en charge de SSL

À partir de la version 4.0, la valeur par défaut pour la propriété de fabrique de connexion des clients `imqSSLIsHostTrusted` est `false`. Si votre application est basée sur la valeur par défaut précédente `true`, vous devez reconfigurer et définir explicitement la propriété sur `true`.

Vous pouvez choisir de faire confiance à l'hôte lorsque le courtier est configuré pour utiliser des certificats autosignés. Dans ce cas, en plus d'indiquer que la connexion doit utiliser un service de connexions SSL (à l'aide de la propriété imqConnectionType), vous devez définir la propriété `imqSSLIsHostTrusted` sur true.

Par exemple, pour exécuter de manière sécurisée les applications clientes lorsque le courtier utilise des certificats autosignés, utilisez une commande similaire à l'exemple suivant.

```
java -DimqConnectionType=TLS
      -DimqSSLIsHostTrusted=true <ClientAppName>
```

Pour exécuter de manière sécurisée l'outil d'administration `imqcmd` lorsque le courtier utilise des certificats autosignés, utilisez une commande similaire à l'exemple suivant.

```
imqcmd list svc -secure -DimqSSLIsHostTrusted=true
```

Prise en charge de JMX

Une nouvelle API a été ajoutée pour la configuration et le contrôle des courtiers Message Queue conformément à la spécification Java Management Extensions (JMX). À l'aide de cette API, vous pouvez configurer et contrôler les fonctions de courtier, à l'aide de programmes, depuis une application cliente Message Queue. Dans les anciennes versions de Message Queue, ces fonctions étaient uniquement accessibles à partir de la ligne de commande ou de la console d'aministration.

Cette API comporte un ensemble de *Managed Beans (MBeans)* JMX pour la gestion des ressources associées à Message Queue suivantes :

- courtiers de messages ;
- services de connexion ;
- connexions ;
- destinations ;

- producteurs de messages ;
- consommateurs de messages ;
- transactions ;
- clusters de courtiers ;
- journalisation ;
- machine virtuelle Java (JVM).

Ces MBeans fournissent des *attributs* et des *opérations* destinés à interroger et à manipuler, de manière synchrone, l'état des ressources sous-jacentes, ainsi que des *notifications* permettant à une application cliente d'écouter et de répondre, de manière asynchrone, aux modifications d'état en temps réel. À l'aide de cette API JMX, les applications clientes peuvent effectuer des tâches de configuration et de contrôle, telles que :

- définir un numéro de port du courtier ;
- définir la taille maximale de message du courtier ;
- interrompre un service de connexions ;
- définir le nombre maximal de threads pour un service de connexions ;
- obtenir le nombre actuel de connexions sur un service ;
- détruire une connexion ;
- créer une destination ;
- détruire une destination ;
- activer ou désactiver la création automatique de destinations ;
- purger tous les messages d'une destination ;
- obtenir le nombre cumulatif des messages reçus par une destination depuis le démarrage du courtier ;
- obtenir l'état actuel (en cours d'exécution ou suspendu) d'une file d'attente ;
- obtenir le nombre actuel de producteurs de messages pour une rubrique ;
- purger tous les messages d'un abonné durable ;
- obtenir la taille actuelle du tas JVM.

Pour consulter une présentation de l'API JMX ainsi que des informations de référence complètes, reportez-vous au manuel *Sun Java System Message Queue 4.1 Developer's Guide for JMX Clients* .

Prise en charge du courtier : propriétés associées à JMX

Plusieurs propriétés de courtier ont été ajoutées pour la prise en charge de l' API JMX (voir le Tableau 1–3). Aucune de ces propriétés ne peut être définie à partir de la ligne de commande avec l'utilitaire Command de Message Queue (`imqcmd`). En revanche, elles peuvent être définies à l'aide de l'option `-D` de l'utilitaire Broker (`imqbrokerd`) ou modifiées manuellement dans le fichier de configuration de l'instance du courtier (`config.properties`). En outre, certaines de

ces propriétés (`imq.jmx.rmiregistry.start`, `imq.jmx.rmiregistry.use`, `imq.jmx.rmiregistry.port`) peuvent être définies à l'aide des nouvelles options de l'utilitaire Broker décrites dans le Tableau 1–4. Le tableau énumère chaque option, spécifie son type et décrit son utilisation.

TABLEAU 1–3 Nouvelles propriétés de courtier pour la prise en charge de JMX

Propriété	Type	Description
`imq.jmx.rmiregistry.start`	`Booléenne`	Indique si le registre RMI doit être démarré au lancement du courtier. Avec une valeur `true`, le courtier démarre un registre RMI sur le port spécifié par `imq.jmx.rmiregistry.port` et l'utilise pour stocker le stub RMI pour les connecteurs JMX. Notez que la valeur de `imq.jmx.rmiregistry.use` n'est pas prise en compte dans ce cas. **Valeur par défaut :** `false`
`imq.jmx.rmiregistry.use`	`Booléenne`	Indique si un registre RMI externe doit être utilisé. S'applique uniquement si `imq.jmx.rmiregistry.start` est `false`. Avec une valeur `true`, le courtier utilise un registre RMI externe sur le port spécifié par `imq.jmx.rmiregistry.port` pour stocker le stub RMI pour les connecteurs JMX. Le registre RMI externe doit déjà être en cours d'exécution au démarrage du courtier. **Valeur par défaut :** `false`
`imq.jmx.rmiregistry.port`	`Entier`	Numéro de port du registre RMI. S'applique uniquement si `imq.jmx.rmiregistry.start` ou `imq.jmx.rmiregistry.use` est `true`. Il est alors possible de configurer les connecteurs JMX pour que ceux-ci utilisent le registre RMI en incluant ce numéro de port dans le chemin URL de leurs URL de service JMX. **Valeur par défaut :** `1099`
`imq.jmx.connector.list`	`Chaîne`	Noms des connecteurs JMX préconfigurés, séparés par une virgule. **Valeur par défaut :** `jmxrmi,ssljmxrmi`
`imq.jmx.connector.activelist`	`Chaîne`	Noms des connecteurs JMX devant être activés au démarrage du courtier, séparés par une virgule. **Valeur par défaut :** `jmxrmi`

TABLEAU 1–3 Nouvelles propriétés de courtier pour la prise en charge de JMX *(Suite)*

Propriété	Type	Description
`imq.jmx.connector.`*connectorName* `.urlpath`	`Chaîne`	Composant *cheminUrl* de l' URL de service JMX pour le *nomConnecteur* du connecteur. S'avère pratique dans les cas où le chemin de l'URL de service JMX doit être défini de manière explicite (par exemple, lorsqu'un registre RMI externe partagé est utilisé). **Valeur par défaut :** si un registre RMI est utilisé pour stocker le stub RMI pour les connecteurs JMX (c'est-à-dire, si `imq.jmx.registry.start` ou `imq.jmx.registry.use` est `true`) : `/jndi/rmi://`*hôteCourtier*`:`*portRmi* */hôteCourtier/portCourtier/nomConnecteur* Si un registre RMI n'est pas utilisé (dans le cas par défaut, `imq.jmx.registry.start` et `imq.jmx.registry.use` sont tous les deux `false`) : `/stub/`*stubRmi* *stubRmi* correspondant à une représentation codée et sérialisée du stub RMI lui-même.
`imq.jmx.connector.`*connectorName* `.useSSL`	`Booléenne`	Indique si un Secure Socket Layer (SSL) doit être utilisé pour le *nomConnecteur* des connecteurs. **Valeur par défaut :** `false`
`imq.jmx.connector.`*nomConnecteur* `.brokerHostTrusted`	`Booléenne`	Indique s'il faut faire confiance aux certificats présentés par le courtier pour le *nomConnecteur* des connecteurs. S'applique uniquement lorsque `imq.jmx.connector.`*nomConnecteur*`.useSSL` est `true`. Avec une valeur `false`, l'exécution client Message Queue valide tous les certificats présentés. La validation échoue si le signataire du certificat n'est pas dans le magasin d'approbations du client. Avec une valeur `true`, la validation des certificats est ignorée. Ceci peut s'avérer pratique, par exemple, lors d'un test logiciel pour lequel un certificat autosigné est utilisé. **Valeur par défaut :** `false`

La propriété `imq.jmx.connector.list` définit un ensemble de connecteurs JMX nommés, devant être créés au démarrage du courtier ; `imq.jmx.connector.activelist` spécifie les connecteurs à activer. Chaque connecteur nommé comporte son propre jeu de propriétés :

```
imq.jmx.connector.nomConnecteur .urlpath
imq.jmx.connector.nomConnecteur .useSSL
imq.jmx.connector.nomConnecteur .brokerHostTrusted
```

Par défaut, deux connecteurs JMX sont créés, `jmxrmi` et `ssljmxrmi` ; le premier est configuré pour ne pas utiliser le chiffrement SSL (`imq.jmx.connector.jmxrmi.useSSL` = `false`, le deuxième pour l'utiliser (`imq.jmx.connector.ssljmxrmi.useSSL` = `true`). Par défaut, seul le connecteur `jmxrmi` est activé au démarrage du courtier ; reportez-vous à la section "Prise en charge de SSL pour les clients JMX" à la page 25 pour obtenir des informations sur la procédure d'activation du connecteur `ssljmxrmi` pour des communications sécurisées.

Par souci de simplicité, de nouvelles options (Tableau 1–4) ont été ajoutées à l'utilitaire Broker de ligne de commande (`imqbrokerd`) en vue de contrôler l'utilisation, le démarrage et le port du registre RMI. L'utilisation et les effets de ces options sont identiques aux propriétés de courtier équivalentes, comme décrit dans le Tableau 1–3. Ce tableau énumère chaque option, spécifie sa propriété de courtier équivalente et décrit son utilisation.

TABLEAU 1–4 Nouvelles options de l'utilitaire Broker pour la prise en charge de JMX

Option	Propriété de courtier équivalente	Description
`-startRmiRegistry`	`imq.jmx.rmiregistry.start`	Indique si le registre RMI doit être démarré au lancement du courtier.
`-useRmiRegistry`	`imq.jmx.rmiregistry.use`	Indique si le registre RMI externe doit être utilisé.
`-rmiRegistryPort`	`imq.jmx.rmiregistry.port`	Numéro de port du registre RMI.

Une nouvelle sous-commande (Tableau 1–5) a été ajoutée à l'utilitaire Command de ligne de commande (`imqcmd`) pour lister les URL de service JMX des connecteurs JMX créés et lancés au démarrage du courtier. Ces informations sont requises par les clients JMX, n'utilisant pas la classe de référence Message Queue `AdminConnectionFactory`, en vue d'obtenir leurs connecteurs JMX, et peuvent être également utilisées pour gérer ou contrôler Message Queue via un navigateur JMX générique, tel que la console de gestion et de contrôle Java (`jconsole`).

TABLEAU 1–5 Nouvelle sous-commande de l'utilitaire Command

Sous-commande	Description
`list jmx`	Liste les URL de service JMX des connecteurs JMX .

Prise en charge de SSL pour les clients JMX

Comme nous l'avons mentionné précédemment, un courtier de messages Message Queue est configuré par défaut pour les communications non sécurisées à l'aide du connecteur JMX préconfiguré, `jmxrmi`. Si vous souhaitez utiliser le Secure Socket Layer (SSL) sur certaines

applications pour des communications sécurisées, vous devez activer le deuxième connecteur JMX sécurisé, `ssljmxrmi`. Pour ce faire, procédez comme suit :

1. Obtenez et installez un certificat signé de la même manière que pour le service de connexion `ssljms`, `ssladmin`, ou `cluster` , comme décrit dans le manuel *Message Queue Administration Guide*.
2. Installez le certificat d'autorité de certification racine dans le magasin d'approbations, si nécessaire.
3. Ajoutez le connecteur `ssljmxrmi` à la liste des connecteurs JMX à activer au démarrage du courtier :

   ```
   imq.jmx.connector.activelist=jmxrmi,ssljmxrmi
   ```

4. Démarrez le courtier à l'aide de l'utilitaire Broker de Message Queue(`imqbrokerd`), soit en lui transmettant le mot de passe du keystore dans un fichier de mots de passe, soit en saisissant celui-ci dans une ligne de commande à l'invite correspondante.
5. Par défaut, le connecteur `ssljmxrmi` (ou tout autre connecteur SSL) est configuré de manière à valider tous les certificats SSL de courtier qui lui sont présentés. Pour éviter cette validation (par exemple, lors de l'utilisation de certificats autosignés pour un test logiciel), définissez la propriété de courtier `imq.jmx.connector.ssljmxrmi.brokerHostTrusted` sur `true`.

Du côté client, la fabrique de connexion administrateur (`AdminConnectionFactory`) doit être configurée avec un URL spécifiant `ssljmxrmi` comme connecteur favori :

```
AdminConnectionFactory  acf = new AdminConnectionFactory();
acf.setProperty(AdminConnectionConfiguration.imqAddress, "mq://myhost:7676/ssljmxrmi");
```

Si nécessaire, utilisez les propriétés système `javax.net.ssl.trustStore` et `javax.net.ssl.trustStorePassword` pour que le client JMX pointe vers le magasin d'approbations.

Journalisation à l'exécution client

Cette section décrit la prise en charge Message Queue 4.0 pour une journalisation à l'exécution client des événements de connexion et de ceux associés à la session.

JDK 1.4 (et versions supérieures) inclut la bibliothèque `java.util.logging`. Cette bibliothèque implémente une interface d'enregistrement standard pouvant être utilisée pour une journalisation basée sur les applications.

L'exécution client Message Queue utilise l'API de journalisation Java pour mettre en œuvre ses fonctions de journalisation. Vous pouvez utiliser tous les services de journalisation de J2SE 1.4 pour la configuration des activités de journalisation. Par exemple, une application peut utiliser les services de journalisation Java suivants pour configurer le mode d'envoi des informations de journalisation par l'exécution client Message Queue :

- gestionnaires de journalisation ;

- filtres de journalisation ;
- formateurs de journalisation ;
- niveaux de journalisation.

Pour plus d'informations sur l'API de journalisation Java, consultez la présentation relative à ce sujet à l'adresse suivante :
`http://java.sun.com/j2se/1.4.2/docs/guide/util/logging/overview.html`

Espaces de noms, niveaux et activités de journalisation

Le fournisseur Message Queue définit un ensemble d'espaces de noms de journalisation associés aux niveaux et activités de journalisation et permettant aux clients Message Queue d'enregistrer les événements de connexion et de session avec une configuration de journalisation correctement définie.

L'espace de noms de journalisation racine pour l'exécution client Message Queue est nommé `javax.jms`. Tous les journaux de l'exécution client Message Queue utilisent ce nom en tant qu'espace de noms parent.

Les niveaux de journalisation utilisés pour l'exécution client Message Queue sont identiques à ceux utilisés dans la classe `java.util.logging.Level`. Cette classe définit sept niveaux de journalisation standard, ainsi que deux paramètres supplémentaires pouvant être utilisés pour activer et désactiver la journalisation.

`OFF`	Désactive la journalisation.
`SEVERE`	Priorité la plus haute, valeur la plus élevée. Défini par l'application.
`WARNING`	Défini par l'application.
`INFO`	Défini par l'application.
`CONFIG`	Défini par l'application.
`FINE`	Défini par l'application.
`FINER`	Défini par l'application.
`FINEST`	Priorité la plus basse, valeur la plus faible. Défini par l'application.
`ALL`	Permet d'enregistrer tous les messages.

En règle générale, les exceptions et les erreurs survenues à l'exécution client Message Queue sont consignées dans le journal avec l'espace de noms `javax.jms`.

- Les exceptions levées à partir de la JVM et interceptées par l'exécution client, telles que `IOException`, sont consignées dans le journal avec l'espace de noms `javax.jms` au niveau `WARNING`.
- Les exceptions JMS levées à partir de l'exécution client, telles que `IllegalStateException` , sont consignées dans le journal avec l'espace de noms `javax.jms` au niveau `FINER`.

- Les erreurs levées à partir de la JVM et interceptées à l'exécution client, telles que `OutOfMemoryError`, sont consignées dans le journal avec l'espace de noms `javax.jms` au niveau `SEVERE`.

Les tableaux suivants répertorient les événements pouvant être consignés, ainsi que le niveau nécessaire à la journalisation d'événements pour les connexions JMS et pour les sessions.

Le tableau suivant décrit les niveaux de journal et les événements relatifs aux connexions.

TABLEAU 1–6 Niveaux de journal et événements pour l'espace de noms `javax.jms.connection`

Niveau de journal	Événements
`FINE`	Connexion créée.
`FINE`	Connexion démarrée.
`FINE`	Connexion fermée.
`FINE`	Connexion rompue.
`FINE`	Connexion rétablie.
`FINER`	Diverses activités de connexion, telles que `setClientID`.
`FINEST`	Messages, accusés de réception, messages d'action et de contrôle de Message Queue (par exemple, validation d'une transaction).

Pour les sessions, les informations suivantes sont consignées dans l'enregistrement du journal.

- Chaque enregistrement de journal, associé à un message remis à un consommateur, inclut un ID de connexion, un ID de session et un ID de consommateur.
- Chaque enregistrement de journal, associé à un message envoyé par un producteur, inclut un ID de connexion, un ID de session, un ID de producteur et un nom de destination.

Le tableau ci-dessous décrit les niveaux de journal et les événements relatifs aux sessions.

TABLEAU 1–7 Niveaux de journal et événements pour l'espace de noms `javax.jms.session`

Niveau de journal	Événement
`FINE`	Session créée.
`FINE`	Session fermée.
`FINE`	Producteur créé.
`FINE`	Consommateur créé.
`FINE`	Destination créée.

TABLEAU 1–7 Niveaux de journal et événements pour l'espace de noms `javax.jms.session` *(Suite)*

Niveau de journal	Événement
`FINER`	Diverses activités de session, telles que la validation d'une session.
`FINEST`	Messages produits et consommés. (Les propriétés et le corps de message ne sont pas consignés dans les enregistrements du journal.)

Par défaut, le niveau du journal de sortie est hérité du JRE dans lequel l'application est exécutée. Consultez le fichier `JRE_DIRECTORY/lib/logging.properties` pour vérifier de quel niveau il s'agit.

Vous pouvez configurer la journalisation à l'aide de programmes ou en utilisant des fichiers de configuration. En outre, vous pouvez contrôler l'étendue de ce processus. Ces possibilités sont décrites dans les sections suivantes.

Utilisation du fichier de configuration de journalisation du JRE

L'exemple suivant présente le mode de définition des espaces de noms et des niveaux de journalisation dans le fichier `JRE_DIRECTORY/lib/logging.properties`, utilisé pour définir le niveau de journal de l'environnement d'exécution Java. Toutes les applications utilisant ce JRE présenteront la même configuration de journalisation. L'exemple de configuration ci-dessous définit le niveau de journalisation sur `INFO` pour l'espace de noms `javax.jms.connection` et indique que la sortie doit être écrite dans `java.util.logging.ConsoleHandler` .

```
#logging.properties file.
# « handlers » spécifie une liste des classes de gestionnaire de journaux,
# séparées par une virgule. Ces gestionnaires seront installés au
# démarrage de la VM. Notez que ces classes doivent être comprises
# dans le chemin de classe système.
# Par défaut, seul un ConsoleHandler (gestionnaire de consoles) est
# configuré, affichant ainsi uniquement des messages aux niveaux INFO
# et supérieurs.

    handlers= java.util.logging.ConsoleHandler

# Niveau de journalisation global par défaut.
# Celui-ci permet de spécifier quels types d’événements sont consignés
# dans l’ensemble des journaux. Pour tout service donné, ce niveau global
# peut être remplacé par un niveau spécifique au service.
# Notez que le ConsoleHandler dispose également d’un paramètre de
# niveau distinct pour limiter les messages imprimés dans la console.

    .level= INFO

# Seuls les messages de niveaux INFO et supérieurs sont imprimés dans la console.
```

```
    java.util.logging.ConsoleHandler.level = INFO
    java.util.logging.ConsoleHandler.formatter =
                                    java.util.logging.SimpleFormatter

# Le journal, dont l'espace de noms est javax.jms.connection, écrira les messages
# Level.INFO dans son ou ses gestionnaires de sortie. Dans cette configuration,
# le gestionnaire de sortie est défini sur java.util.logging.ConsoleHandler.

    javax.jms.connection.level = INFO
```

Utilisation d'un fichier de configuration de journalisation pour une application spécifique

Vous avez également la possibilité de définir un fichier de configuration de journalisation à partir de la ligne de commande Java utilisée pour exécuter une application. L'application utilisera alors la configuration définie dans le fichier de journalisation spécifié. Dans l'exemple ci-dessous, `configFile` utilise le même format que celui défini dans le fichier `JRE_DIRECTORY/lib/logging.properties` .

```
java -Djava.util.logging.config.file=configFile MQApplication
```

Définition de la configuration de journalisation à l'aide de programmes

Le code suivant utilise l'API `java.util.logging` pour consigner les événements de connexion en modifiant le niveau de journal de l'espace de noms `javax.jms.connection` sur `FINE`. Vous pouvez inclure ce type de code dans votre application pour définir la configuration de la journalisation via des programmes.

```
import java.util.logging.*;
//créer un gestionnaire de fichiers et l'envoyer vers le fichier mq.log
//dans le répertoire temporaire du système.

    Handler fh = new FileHandler("%t/mq.log");
    fh.setLevel (Level.FINE);

//Obtenir un journal pour le domaine « javax.jms.connection ».

    Logger logger = Logger.getLogger("javax.jms.connection");
    logger.addHandler (fh);

//le journal javax.jms.connection consigne les activités
//de niveaux FINE et supérieurs.

    logger.setLevel (Level.FINE);
```

Notification d'événements de connexion

Les notifications d'événements de connexion permettent à un client Message Queue d'écouter les événements de fermeture et de reconnexion et d'entreprendre l'action appropriée selon le type de notification et l'état de connexion. Par exemple, lorsqu'un basculement se produit et que le client est reconnecté à un autre courtier, une application peut vouloir nettoyer l'état de transaction correspondant et utiliser une nouvelle transaction.

Si le fournisseur Message Queue détecte un problème sérieux au niveau d'une connexion, celui-ci appelle le listener d'exceptions enregistrées de l'objet de connexion. Il appelle la méthode `onException` du listener et lui transmet un argument `JMSException` de description du problème. Le fournisseur Message Queue offre également une API de notification d'événements permettant à l'exécution client d'informer l'application des modifications de l'état de connexion. L'API de notification est définie à l'aide des éléments suivants :

- Le package `com.sun.messaging.jms.notification`, chargé de définir le listener d'événements et les objets d'événements de notification.
- L'interface `com.sun.messaging.jms.Connection`, chargée de définir les extensions vers l'interface `javax.jms.Connection`.

Les sections suivantes présentent les événements pouvant déclencher une notification, ainsi que la procédure de création d'un listener d'événements.

Événements de connexion

Le tableau suivant répertorie et décrit les événements pouvant être retournés par le listener d'événements.

Notez que le listener d'exceptions JMS n'est pas appelé lorsqu'un événement de connexion se produit. Celui-ci est uniquement appelé une fois que l'exécution client a épuisé ses tentatives de reconnexion. L'exécution client appelle toujours le listener d'événements avant le listener d'exceptions.

TABLEAU 1–8 Événements de notification

Type d'événement	Signification
`ConnectionClosingEvent`	L'exécution client Message Queue génère cet événement à la réception d'une notification du courtier informant de la fermeture d'une connexion suite à la demande d'arrêt de la part de l'administrateur.

TABLEAU 1–8 Événements de notification *(Suite)*

Type d'événement	Signification
`ConnectionClosedEvent`	L'exécution client Message Queue génère cet événement à la fermeture d'une connexion suite à une erreur du courtier ou à la demande d'arrêt ou de redémarrage de la part de l'administrateur. Lorsqu'un listener d'événements reçoit un `ConnectionClosedEvent`, l'application peut utiliser la méthode `getEventCode()` de l'événement reçu pour obtenir un code d'événement spécifiant la cause de la fermeture.
`ConnectionReconnectedEvent`	L'exécution client Message Queue s'est reconnectée à un courtier. Il peut s'agir du même courtier auquel le client était précédemment connecté ou d'un autre courtier. L'application peut utiliser la méthode `getBrokerAddress` de l'événement reçu pour obtenir l'adresse du courtier auquel elle a été reconnectée.
`ConnectionReconnectFailedEvent`	L'exécution client Message Queue n'est pas parvenue à se reconnecter à un courtier. Chaque fois qu'une tentative de reconnexion échoue, l'exécution génère un nouvel événement et le transmet au listener d'événements. Le listener d'exceptions JMS n'est pas appelé lorsqu'un événement de connexion se produit. Celui-ci est uniquement appelé une fois que l'exécution client a épuisé ses tentatives de reconnexion. L'exécution client appelle toujours le listener d'événements avant le listener d'exceptions.

Création d'un listener d'événements

L'exemple de code suivant illustre la procédure de définition d'un listener d'événements de connexion. À chaque événement de connexion, la méthode `onEvent` du listener d'événements est invoquée par l'exécution client.

```
//créer une fabrique de connexion MQ.

com.sun.messaging.ConnectionFactory factory =
        new com.sun.messaging.ConnectionFactory();

//créer une connexion MQ.

com.sun.messaging.jms.Connection connection =
       (com.sun.messaging.jms.Connection )factory.createConnection();
```

```
//définir un listener d'événements MQ.  Ce dernier implémente l'interface
//com.sun.messaging.jms.notification.EventListener.

com.sun.messaging.jms.notification.EventListener eListener =
       new ApplicationEventListener();

//définir le listener d'événements sur la connexion MQ.

connection.setEventListener ( eListener );
```

Exemples de listener d'événements

Dans cet exemple, une application détermine que son listener d'événements consigne l'événement de connexion dans son propre système de journalisation :

```
la classe publique ApplicationEventListener implémente
                com.sun.messaging.jms.notification.EventListener {

public void onEvent ( com.sun.messaging.jms.notification.Event connEvent ) {
          log (connEvent);
}
private void log ( com.sun.messaging.jms.notification.Event connEvent ) {
          String eventCode = connEvent.getEventCode();
          String eventMessage = connEvent.getEventMessage();
          //écrire les informations de l'événement dans le flux de sortie.
         }
}
```

Configurations matérielle et logicielle requises

Pour obtenir les configurations matérielle et logicielle de la version 4.0, consultez les Notes de version de Sun Java System Application Server Platform Edition 9.

Bogues résolus dans la présente version

Le tableau suivant présente les bogues qui ont été résolus dans la version 4.1 de Message Queue.

TABLEAU 1–9 Bogues résolus dans Message Queue 4.1

Bogue	Description
6381703	Les messages distants transactionnels peuvent être validés deux fois si le courtier à l'origine des messages redémarre.
6388049	Impossible de nettoyer une transaction distribuée incomplète.
6401169	Les options de validation et d'annulation de imqcmd n'envoient pas d'invite de confirmation.
6473052	Par défaut, les files d'attente créées automatiquement doivent être alternées. (`MaxNumberConsumers` = -1).
6474990	Le journal du courtier affiche une `ConcurrentModificationException` pour la commande `imqcmd list dst`.
6487413	Fuite de mémoire lorsque le comportement aux limites est `REMOVE_OLDEST` ou `REMOVE_LOWER_PRIORITY`.
6488340	Basculement du courtier et attente de la réponse par le client pour en accuser réception.
6502744	Le courtier ne respecte pas la limite par défaut de la file d'attente de message bloqués de 1000 messages.
6517341	L'exécution client doit améliorer la logique de reconnexion lorsque le client est connecté à un cluster haute disponibilité, en autorisant celui-ci à se reconnecter quelle que soit la valeur de la propriété *imqReconnectEnabled*.
6528736	Le service de démarrage automatique de Windows (*imqbrokersvc*) s'arrête brutalement au démarrage.
6561494	Les messages sont transmis au mauvais consommateur lorsque ceux-ci partagent une session.
6567439	Les messages produits dans une transaction de niveau `PREPARED` sont transmis en désordre s'ils sont validés après le redémarrage du courtier.

Le tableau suivant décrit les bogues résolus dans Message Queue 4.0.

TABLEAU 1–10 Bogues résolus dans Message Queue 4.0

Référence	Description
4986481	Dans Message Queue 3.5, l'appel de `Session.recover` peut être bloqué en mode de reconnexion automatique.
4987325	L'indicateur de redistribution a été défini sur `false` pour les messages redistribués après l'appel de `Session.recover`.
6157073	Modification du nouveau message de connexion pour inclure le nombre de connexions sur le service, en plus du nombre total de connexions.

TABLEAU 1–10 Bogues résolus dans Message Queue 4.0 *(Suite)*

Référence	Description
6193884	Message Queue envoie un message parasite vers le syslog dans des langues utilisant des caractères non ASCII pour les messages.
6196233	La sélection de messages à l'aide de `JMSMessageID` ne fonctionne pas.
6251450	`ConcurrentModificationException` sur `connectList` durant la fermeture du cluster.
6252763	`java.nio.BufferOverflowException` dans `java.nio.HeapByteBuffer.putLong/Int` .
6260076	Le premier message publié après le démarrage est lent avec le stockage Oracle.
6260814	Le sélecteur traitant `JMSXUserID` donne une évualuation toujours `false`.
6264003	Le navigateur de file d'attente affiche des messages qui font partie des transactions qui n'ont pas été validées.
6271876	Le contrôle de flux de connexions ne fonctionne pas correctement lors de la fermeture d'un consommateur avec des messages non consommés.
6279833	Message Queue ne doit pas autoriser deux courtiers à utiliser les mêmes tables JDBC.
6293053	Le courtier maître ne démarre pas correctement si l'adresse IP du système a été modifiée, à moins que le magasin soit nettoyé (via `–reset store`.)
6294767	Le courtier de Message Queue doit définir `SO_REUSEADDR` sur les sockets de réseau qu'il ouvre.
6304949	Impossible de définir la propriété `ClientID` pour `TopicConnectionFactory`.
6307056	Le journal txn est un goulot d'étranglement des performances.
6320138	L'API C de Message Queue manque de capacités pour déterminer le nom d'une file d'attente à partir d'un en-tête Répondre à.
6320325	Le courtier sélectionne parfois JDK 1.4 avant JDK 1.5 sur Solaris même lorsque les deux versions sont installées.
6321117	L'initialisation de cluster multicourtier émet une `java.lang.NullPointerException` .
6330053	Le client JMS lève une `java.lang.NoClassDefFoundError` lors de la validation d'une transaction de l'abonné.
6340250	Prise en charge du type `MESSAGE` dans l'API C.
6351293	Ajout d'une prise en charge pour la base de données Apache Derby.

Informations importantes

Cette section comprend les dernières informations qui n'ont pu être incluses dans la documentation de base des produits. Cette section aborde les rubriques suivantes :

Notes relatives à l' installation

Reportez-vous au manuel *Sun Java System Message Queue 4.1 Installation Guide* pour obtenir des informations sur les instructions de pré-installation, les procédures de mise à niveau, et autres informations relative à l'installation de Message Queue, Platform Edition sur des plates-formes Solaris, Linux et Windows.

Reportez-vous au *Guide d'installation de Sun Java Enterprise System* pour obtenir des informations sur les instructions de pré-installation et autres informations relatives à l'installation de Message Queue, Enterprise Edition sur des plates-formes Solaris, Linux, et HPUX.

Reportez-vous au manuel *Sun Java Enterprise System Upgrade and Migration Guide* pour obtenir des informations sur les instructions de migration et de mise à niveau relatives à la mise à niveau de Message Queue Enterprise Edition sur des plates-formes Solaris, Linux, HPUX et Windows.

Problèmes de compatibilité

Cette section présente les problèmes de compatibilité dans Message Queue 4.1.

Stabilité de l'interface

Sun Java System Message Queue utilise de nombreuses interfaces pouvant être modifiées dans le temps. L'Annexe B, "Stability of Message Queue Interfaces" du *Sun Java System Message Queue 4.1 Administration Guide* classe ces interfaces selon leur degré de stabilité. Plus l'interface est stable, plus il y a de chances pour qu'elle ne soit pas modifiée dans les versions à venir d'un produit.

Problèmes liés à la prochaine version principale de Message Queue

La prochaine version principale de Message Queue peut présenter des modifications affectant la compatibilité des clients. Cette information vous est fournie dès maintenant de sorte que vous puissiez vous y préparer.

- L'emplacement des fichiers individuels installés pour Sun Java System Message Queue peut changer. Cette modification risque de briser les applications existantes qui dépendent de l'emplacement actuel de certains fichiers Message Queue.
- Il est possible que les courtiers 3.5 et antérieurs ne puissent plus fonctionner dans un cluster avec de nouveaux courtiers.
- Dans les prochaines versions, il est possible que Message Queue ne puisse plus utiliser les versions JDK antérieures à 1.5.

Mises à jour de la documentation relative à Message Queue 4.1

Outre ces *Notes de version* , Message Queue 4.1 comporte un seul nouveau document : *Sun Java System Message Queue 4.1 Developer's Guide for JMX Clients* . Ce manuel a été ajouté avec la version 4.0 de Message Queue. Dans la version 4.1, des informations conceptuelles ont été ajoutées pour présenter le modèle JMX.

La documentation relative à Message Queue 3.6 SP3, 2005Q4 est aujourd'hui à jour, conformément aux besoins des clients Application Server 9 PE. Ces divers documents sont disponibles à l'adresse suivante :

`http://docs.sun.com/app/docs/coll/1307.1`

Informations d'installation et de mise à niveau

Le manuel *Sun Java System Message Queue 4.1 Installation Guide* a été mis à jour pour évoquer les informations spécifiques aux plates-formes. Ce document contient désormais des informations relatives à l'installation et la mise à niveau de Message Queue 4.1.

Administration Guide

Le manuel *Administration Guide* a été mis à jour pour fournir des informations sur les clusters haute disponibilité, la prise en charge de JAAS et la prise en charge de JMX.

Developer's Guide for Java Clients

Le manuel *Developer's Guide for Java Clients* a été mis à jour pour signaler l'ajout de la prise en charge de la journalisation à l'exécution client et des notifications d'événements de connexion.

Developer's Guide for C Clients

Le manuel *Developer's Guide for C Clients* a été mis à jour pour signaler l'ajout de la fonction `MQGetDestinationName`, du type de message `MQ_Message` et de ports fixes.

Problèmes et restrictions connus

Cette section contient une liste des problèmes connus concernant Message Queue 4.1. Elle aborde plus particulièrement les questions suivantes :

Pour obtenir une liste des bogues actuels, de leur état et de leurs solutions, les membres de Java Developer Connection™ peuvent consulter la « Bug Parade » sur le site Web de Java Developer Connection. Avant de signaler tout nouveau bogue, merci de consulter cette page. Bien que tous les bogues de Message Queue n'y soient pas répertoriés, il est préférable ce consulter cette page pour savoir si un problème a déjà été signalé.

`http://bugs.sun.com/bugdatabase/index.jsp`

Remarque – L'adhésion à Java Developer Connection est gratuite, mais elle requiert une inscription. Pour savoir comment devenir membre de Java Developer Connection, consultez la page Web « For Developers » de Sun .

Pour signaler un nouveau bogue ou soumettre une demande d'amélioration, envoyez un e-mail à l'adresse suivante : `imq-feedback@sun.com` .

Problèmes d'installation

Cette section décrit les problèmes liés à l'installation de Message Queue version 4.1.

Registre de produit et JES

La version 4.1 de Message Queue est installée par un nouveau programme d'installation, qui permet également d'installer et de mettre à niveau les composants partagés requis par Message Queue (par exemple, JDK, bibliothèques NSS, JavaHelp, etc.) Le programme d'installation et Java Enterprise System (JES) ne partagent pas le même registre de produit. Si une ancienne version de Message Queue, précédemment installée avec JES, est supprimée et mise à niveau vers Message Queue 4.1 par le programme d'installation, le registre de produit JES peut alors se trouver en état instable. Ainsi, lorsque le programme de désinstallation de JES est exécuté, il est possible qu'il supprime par accident Message Queue 4.1 et les composants partagés dont il dépend mais qu'il n'a pas installés.

La meilleure procédure à suivre pour mettre à niveau un logiciel installé par le programme d'installation de JES est la suivante :

1. Utilisez le programme de désinstallation de JES pour supprimer Message Queue et ses composants partagés.
2. Utilisez le programme d'installation de Message Queue pour installer Message Queue 4.1.

Sélection du JRE approprié

L'écran de sélection du JDK du programme d'installation de Message Queue 4.1 vous permet de sélectionner un JDK/JRE existant sur le système pour l'utiliser avec Message Queue. Malheureusement, la liste proposée comprend également le JRE utilisé pour exécuter l'application du programme d'installation. Ce JRE fait partie du package d'installation et n'est pas réellement installé sur le système. *(Bogue 6585911)*

Le JRE utilisé par le programme d'installation est reconnaissable par son chemin, qui doit se trouver dans le répertoire du programme d'installation décompressé et inclure le sous-répertoire `mq4_1–installer`. Par exemple :

some_directory`/mq4_1–installer/usr/jdk/instances/jdk1.5.0/jre`

Ne sélectionnez pas ce JRE pour Message Queue. Sélectionnez plutôt un autre JDK sur le système. Sinon, suivez la procédure appropriée à votre plate-forme.

- Sous Solaris ou Linux : sélectionnez « Installer et utiliser le JDK par défaut ».
- Sous Windows : téléchargez et installez un JDK avant d'exécuter le programme d'installation de Message Queue 4.1.

Installation sous Windows

Lorsque vous installez Message Queue sous Windows, veuillez prendre en compte les limitations suivantes.

- Le programme d'installation n'ajoute pas d'entrée pour Message Queue dans le menu Démarrer > Programmes (*Bogue 6567258*). Pour démarrer la console d'administration, utilisez la ligne de commande comme décrit dans la section “Starting the Administration Console” du *Sun Java System Message Queue 4.1 Administration Guide*.
- Le programme d'installation n'ajoute pas le répertoire `IMQ_HOME\mq\bin` à la variable d'environnement `PATH`.*(Bogue 6567197)*. Les utilisateurs doivent ajouter cette entrée à leur variable d'environnement `PATH` ou fournir un nom de chemin complet lorsqu'ils invoquent les utilitaires de Message Queue (`IMQ_HOME\mq\bin\`*commande*).
- Le programme d'installation n'ajoute pas d'entrée dans le registre Windows pour indiquer que Message Queue a été installé.
- Lorsqu'il est exécuté en mode silencieux, le programme d'installation est directement renvoyé. L'installation s'effectue mais l'utilisateur n'a pas la possibilité de savoir quand est ce que l'installation silencieuse est réellement exécutée. *(Bogue 6586560)*

- Le mode Texte (`installer –t`) n'est pas pris en charge sous Windows. L'exécution du programme d'installation en mode texte sous Windows entraîne l'affichage d'un message d'erreur. Ce message apparaît en anglais même si le programme d'installation est exécuté en langue non anglaise. *(Bogue 6594142)*
- La chaîne « Install Home » affichée sur l'écran d'accueil du programme d'installation apparaît en anglais même si ce dernier est exécuté en langue non anglaise. *(Bogue 6592491)*

Installation sous Solaris

Le message d'erreur et l'état de résumé « incomplet » induit en erreur l'utilisateur tentant une installation à l'aide de la commande `installer-n`. En réalité, cette commande fonctionne correctement. *(Bogue 6594351)*

Installation sous Linux

Les problèmes suivants affectent le processus d'installation sur une plate-forme Linux.

- Sur le panneau de sélection du JDK, la liste déroulante n'affiche qu'un seul élément. Il est donc difficile de sélectionner tout autre JDK dans la liste. *(Bogue 6584735)*
- Si le JDK est actif et que l'utilisateur sélectionne « Installer le JDK par défaut » à l'écran de sélection du JDK, le programme d'installation tente toujours de l'installer et signale qu'il n'est pas en mesure d'installer le package. L'installation se termine correctement malgré ce problème. *(Bogue 6581310)*
- Lorsque le programme d'installation est exécuté en mode d'exécution sèche (`installer –n`), l'écran de résumé affiche des messages d'erreur, ainsi qu'un état d'installation « Incomplet ». Cela est erroné ; ce processus n'installe aucun élément sur le système ; il crée uniquement un fichier de réponse qui peut être utilisé par la suite pour l'installation.*(Bogue 6594351)*
- Si d'anciennes versions de RPM de localisation de Message Queue existent sur votre système, l'installation des RPM de localisation de Message Queue 4.1 (survenant lorsque vous cochez la case d'installation des packages multilingues de Message Queue à l'écran correspondant) échoue. Ceci s'explique du fait que les packages Il8 de l'installation 3.7 UR1 provoquent un conflit. *(Bogue 6594381)*

  Solution : supprimez manuellement les RPM de localisation à l'aide de la commande `rpm –e` avant d'exécuter le programme d'installation de la version 4.1. Pour repérer les RPM concernés, reportez-vous à la section “Message Queue Packages (RPMs)” du *Sun Java System Message Queue 4.1 Installation Guide*.

Installation sur toutes les plates-formes

Ces problèmes affectent le processus d'installation sur toutes les plates-formes.

- Lorsque le programme d'installation est en cours d'exécution pour l'installation de Message Queue 4.1 et que l'écran de progression est affiché, le bouton Annuler est actif. Si vous sélectionnez ce bouton à ce stade, l'installation sera incomplète ou endommagée. *(Bogue 6595578)*

- L'écran de résumé du programme d'installation contient certains liens qui, lorsque vous cliquez dessus, lancent un visualiseur de page de synthèse ou de journal. Si vous fermez la fenêtre du visualiseur en utilisant le bouton « X » au lieu du bouton Fermer, il sera alors impossible de la faire réapparaître. *(Bogue 6587138)*

  Solution : utilisez le bouton Fermer pour fermer la fenêtre.

- Lorsqu'un système dispose d'anciennes versions de Message Queue et de NSS/NSPR, la fonction de mise à niveau du programme d'installation signale uniquement Message Queue dans les besoins de mise à niveau ; NSS/NSPR n'est pas mentionné. Ce problème concerne uniquement l'écran de mise à jour étant donné que les logiciels nécessaires sont mis à niveau durant le processus d'installation (comme indiqué à l'écran des éléments prêts à être installés sur lequel apparaissent les informations correctes). *(Bogue 6580696)*

  Solution : aucune action n'est requise étant donné que les fichiers NSS/NSPR sont installés si ceux-ci sont inactifs et que les anciennes versions sont désinstallées.

- Lorsque le programme d'installation ou de désinstallation est exécuté en mode texte (`installer –t`), l'écran de résumé affiche le répertoire contenant les fichiers journaux ou de synthèse mais ne répertorie pas le nom de ces fichiers. *(Bogue 6581592)*

- La spécification d'un nom de fichier inexistant entraîne des messages d'erreur incohérents et erronés. *(Bogue 6587127)*

Informations relatives à la version

Le programme d'installation affiche les informations de version de Message Queue sous une forme opaque. *(Bogue 6586507)*

Pour les plates-formes Solaris, reportez-vous au tableau ci-dessous pour déterminer la version en cours d'installation.

TABLEAU 1–11 Formats de version

Version affichée par le programme d'installation	Version de Message Queue
4.1.0.0	4.1
3.7.0.1	3.7 UR1
3.7.0.2	3.7 UR2
3.7.0.3	3.7 UR3
3.6.0.0	3.6
3.6.0.1	3.6 SP1
3.6.0.2	3.6 SP2
3.6.0.3	3.6 SP3

TABLEAU 1–11 Formats de version *(Suite)*

Version affichée par le programme d'installation	Version de Message Queue
3.6.0.4	3.6 SP4

Remarque – Pour les versions de patch jusqu'au 3.6 SP4 (par exemple, 3.6 SP4 Patch 1), la chaîne de version affichée par le programme d'installation reste la même. Voud devez exécuter la commande `imqbrokerd –version` pour déterminer la version exacte.

Sur les plates-formes Linux, il est impossible de fournir une traduction de format simple. Le numéro de version affiché par le programme d'installation sous Linux adopte le format suivant :

```
<majorReleaseNumber>.<minorReleaseNumber>-<someNumber>
```

Par exemple, 3.7–22. Ce numéro signale qu'il s'agit de l'une des versions 3.7 sans spécifier laquelle. Pour obtenir la version exacte, exécutez la commande `imqbrokerd –version` .

Problèmes de localisation

Les bogues suivants sont liés aux problèmes de localisation.

- Lorsque le programme d'installation est exécuté en mode texte (`installer –t`), dans une langue non anglaise, les caractères multioctet apparaissent corrompus. *(Bogue 6586923)*
- L'écran de résumé de l'installation permet à l'utilisateur de consulter un rapport de synthèse. Toutefois, ce rapport (sous forme de page HTML) affiche des caractères corrompus lorsque le programme d'installation est exécuté en langues multioctet. *(Bogue 6587112)*

  Solution : modifiez le fichier HTML pour corriger le jeu de caractères qu'il contient. Son contenu devrait être similaire à l'exemple suivant :

  ```
  meta http-equiv="Content-Type" content="text/html; charset=UTF-8
  ```

  Remplacez « UTF-8 » par *nom_langue* `.UTF-8`. Par exemple, `ja_JA.UTF-8` ou `ko.UTF-8` sous Solaris ; `ja_JA.utf8` ou `ko_KO.utf8` sous Linux.
- Sur l'écran de progression de l'installation, la barre de progression affiche des caractères inconnus. L'infobulle est codée en dur pour les langues non anglaises. *(Bogue 6591632)*
- Le mode Texte (`installer –t`) n'est pas pris en charge sous Windows. L'exécution du programme d'installation en mode texte, sous Windows, entraîne l'affichage d'un message d'erreur. Ce message n'est pas localisé lorsque le programme d'installation est exécuté en langues non anglaises. *(Bogue 6594142)*
- L'écran relatif à la licence du programme d'installation affiche le texte correspondant en anglais, quelle que soit la langue d'exécution du programme.*(Bogue 6592399)*

  Solution : pour obtenir les fichiers de licence localisés, consultez le fichier `LICENSE_MULTILANGUAGE.pdf`.

- L'aide relative à l'utilisation du programme d'installation n'est pas localisée. *(Bogue 6592493)*
- La chaîne « None » apparaissant sur la page HTML de synthèse du programme d'installation est codée en dur en anglais. *(Bogue 6593089)*
- La page de copyright n'est pas localisée, sauf pour le français. *(Bogue 6590992)*
- Lorsque le programme d'installation est exécuté dans un environnement linguistique allemand, l'écran d'accueil n'affiche pas le texte complet qui apparaît pour les autres langues. *(Bogue 6592666)*
- La chaîne « Install Home » apparaissant sur la page d'accueil de l'installation n'est pas localisée. Elle s'affiche en anglais même si le programme d'installation est exécuté en langues non anglaises. *(Bogue 6592491)*
- Lorsque le programme d'installation est exécuté en mode texte (`installer –t`), les choix de réponse en anglais « Yes » et « No » sont utilisés quelle que soit la langue d'exécution choisie. *(Bogue 6593230)*
- L'infobulle du bouton Parcourir à l'écran de sélection du JDK est codée en dur en anglais. *(Bogue 6593085)*

Option de mot de passe désapprouvée

Dans les versions précédentes de Message Queue, vous aviez la possibilité d'utiliser l'option `–p` ou `–password` pour spécifier un mot de passe, de manière interactive, pour les commandes suivantes : `imqcmd`, `imqbrokerd` et `imdbmgr`. À partir de la version 4.0, ces options ont été désapprouvées. Vous devez désormais spécifier vos mots de passe de la manière suivante.

1. Définissez la propriété de mot de passe sur la valeur choisie dans un fichier uniquement utilisé pour le stockage des mots de passe.

   Utilisez la syntaxe suivante pour spécifier vos mots de passe dans ce fichier :

   NomPropriétéMotdepasse= MonMotdepasse
2. Transmettez le nom du fichier de mots de passe à l'aide de l'option `–passfile` .

Un fichier de mots de passe peut contenir un ou plusieurs des mots de passe énumérés ci-dessous :

- Un mot de passe de keystore utilisé pour ouvrir le keystore SSL. Utilisez la propriété `imq.keystore.password` pour spécifier ce mot de passe.
- Un mot de passe de référentiel LDAP utilisé pour se connecter, de manière sécurisée, à l'aide d'un répertoire LDAP si la connexion n'est pas anonyme. Utilisez la propriété `imq.user_repository.ldap.password` pour spécifier ce mot de passe.

- Un mot de passe de base de données JDBC utilisé pour se connecter à une base de données compatible JDBC. Utilisez la propriété `imq.persist.jdbc.vendorName.password` pour spécifier ce mot de passe. Le composant *nomFournisseur* du nom de la propriété est une variable spécifiant le fournisseur de la base de données. Vous avez le choix entre `hadb`, `derby`, `pointbase`, `oracle` ou `mysql`.
- Un mot de passe pour la commande `imqcmd` (en vue d'effectuer des tâches d'administration du courtier). Utilisez la propriété `imq.imqcmd.password` pour spécifier ce mot de passe.

Dans l'exemple suivant, le mot de passe pour la base de données JDBC est défini sur `abracadabra`.

```
imq.persist.jdbc.mysql.password=abracadabra
```

Vous pouvez configurer le courtier de manière à ce qu'il utilise le fichier de mots de passe créé en suivant l'une des procédures suivantes :

- Définissez les propriétés suivantes dans le fichier `config.properties` du courtier.

  ```
  imq.passfile.enabled=true
  imq.passfile.dirpath=MonRépertoiredeFichiers
  imq.passfile.name=MonNomdeFichier
  ```

- Utilisez l'option `–passfile` de la commande `imqbrokerd`.

  `imqbrokerd –passfile` *MonNomdeFichier*

Généralités

Cette section aborde des problèmes d'ordre général dans Message Queue 4.1. Certains de ces problèmes proviennent des versions précédentes de Message Queue.

- Lorsqu'un client JMS utilisant le transport HTTP met brutalement fin à la connexion (en utilisant, par exemple, `Ctrl-C`), le courtier met environ une minute avant de libérer la connexion client et toutes les ressources associées.

  Si une autre instance du client est démarrée dans la minute d'attente en essayant d'utiliser le même ID client, la même souscription durable ou file d'attente, il est possible que celle-ci reçoive une exception « L'ID client est déjà utilisé ». Il ne s' agit pas d'un vrai problème, mais d'un effet secondaire du processus de fin décrit précédemment. Si un client est démarré après un délai d' environ une minute, tout doit fonctionner correctement.
- Clients SOAP : auparavant, le fichier jar d'implémentation SAAJ 1.2 faisait référence à `mail.jar` et `mail.jar` ne devait pas nécessairement être stocké dans `CLASSPATH`. Dans SAAJ 1.3, cette référence a été supprimée ; par conséquent, les clients Message Queue doivent explicitement placer `mail.jar` dans `CLASSPATH`.

Problèmes d'administration/de configuration

Les problèmes suivants sont liés à l'administration et à la configuration de Message Queue

- Les utilitaires `imqadmin` et `imqobjmgr` lèvent une erreur lorsque `CLASSPATH` contient des guillemets doubles sur des machines Windows (*ID de bogue 5060769*).

  Solution : vous pouvez ignorer ce message d'erreur ; le courtier gère correctement la notification d'erreur aux consommateurs. Cette erreur n'affecte pas la fiabilité du système.

- L'option `-javahome` dans tous les scripts Solaris et Windows ne fonctionne pas si la valeur fournie contient un espace (*ID de bogue 4683029*).

  L'option `javahome` est utilisée par les commandes et utilitaires de Message Queue pour spécifier une autre exécution Java 2 compatible à utiliser. Cependant, le nom de chemin vers l'exécution Java alternative ne doit pas contenir d'espace. Voici quelques exemples de chemins contenant des espaces :

  Windows : `C:/jdk 1.4`

  Solaris : `/work/java 1.4`

  Solution : installez le programme d'exécution Java à un emplacement ou un chemin ne contenant pas d'espace.

- L'attribut `imqQueueBrowserMaxMessagesPerRetrieve` spécifie le nombre maximal de messages pouvant être récupérés en une seule fois par l'exécution client lors de la navigation dans une file d'attente. Notez que l'application cliente obtient toujours tous les messages de la file d'attente. Par conséquent, l'attribut `imqQueueBrowserMaxMessagesPerRetrieve` affecte le mode de découpage des messages en file d'attente, quoi doivent être transmis à l'exécution client (un petit nombre de grands blocs ou un grand nombre de petits blocs), mais n'affecte pas le nombre total de messages parcourus. La modification de la valeur de cet attribut peut affecter les performances mais n'entraînera pas un flux de données moindre ou plus important dans l'application cliente (*ID de bogue 6387631*).

Problèmes relatifs au courtier

Les problèmes suivants concernent le courtier de Message Queue.

- Il existe une certaine confusion sur le mode de configuration du courtier pour une transmission à tour de rôle. La solution est simple et configurable.

  1. Définissez l'attribut de destination `maxNumActiveConsumers` sur -1. Cela permet d'activer la transmission à tour de rôle.
  2. Définissez l'attribut de destination `consumerFlowLimit` sur 1. Cela spécifie le nombre de messages transmis à un seul consommateur avant que la transmission ne passe au consommateur suivant. Pour un découpage différent, définissez cet attribut sur la valeur souhaitée. Par défaut, chaque consommateur reçoit cent messages.

- Le courtier devient inaccessible lorsque le magasin persistant ouvre trop de destinations. (*ID de bogue 4953354*).

  Solution : ce problème est dû au fait que le courtier atteint la limite du descripteur de fichier ouvert définie pour le système. Sur Solaris et Linux, utilisez la commande `ulimit` pour augmenter cette limite.

- Les consommateurs sont orphelins lorsqu'une destination est supprimée (*ID de bogue 5060787*).

  Les consommateurs actifs sont orphelins lorsqu'une destination est supprimée. Une fois orphelins, ils ne peuvent plus recevoir de messages (même si la destination est recréée).

  Solution : il n'existe aucune solution pour ce problème.

Clusters de courtier

Les problèmes suivants affectent les courtiers clusterisés.

- Seuls les clusters de courtier entièrement connectés sont pris en charge par cette version. Autrement dit, tous les courtiers d'un cluster doivent communiquer directement avec tous les autres. Si vous essayez de connecter les courtiers à l'aide de l'argument de ligne de commande `imqbrokerd -cluster`, assurez-vous que tous les courtiers du cluster sont bien inclus.
- Un courtier utilisant HADB ne peut pas gérer des messages supérieurs à 10 Mo. *(Bogue 6531734)*
- Si un client est connecté à un courtier haute disponibilité, l'exécution client effectue des tentatives de reconnexion jusqu'à ce que celle-ci réussisse (quelle que soit la valeur de `imqAddressListIterations`.)
- Un client connecté à un courtier appartenant à un cluster ne peut actuellement pas utiliser `QueueBrowser` pour parcourir les files d'attente situées sur les courtiers distants de ce cluster. Il ne peut que parcourir le contenu des files d'attente situées sur le courtier auquel il est directement connecté. Il peut toutefois continuer d'envoyer des messages vers les files d'attente ou de consommer les messages provenant des files d'attente sur n'importe quel courtier du cluster, la limitation ne s'appliquant en effet qu'à la navigation.
- Dans un cluster conventionnel, si vous souhaitez clusteriser un courtier 4.1 avec un courtier 3.x, vous devez définir la propriété `imq.autocreate.queue.maxNumActiveConsumers=1` pour le courtier 4.1. Sinon, les courtiers ne seront pas en mesure d'établir une connexion au cluster.
- Lors de la conversion en un cluster haute disponibilité, vous pouvez utiliser l'utilitaire Message Queue Manager (imqdbmgr) pour convertir un magasin de données persistantes HADB autonome en un magasin HADB partagé. Pour ce faire, utilisez la commande suivante.

  ```
  imqdbmgr upgrade hastore
  ```

Vous pouvez utiliser cet utilitaire dans les cas suivants :

- Basculement d'un magasin HADB autonome 4.0 à un magasin HADB partagé 4.1. Dans ce cas, le courtier met le magasin automatiquement à niveau. Vous pouvez ensuite exécuter la commande `imqdbmgr` pour convertir le magasin de données mis à niveau en vue d'une utilisation partagée.
- Basculement d'un magasin HADB autonome 4.1 à un magasin HADB partagé. Dans ce cas, il vous suffit d'exécuter la commande `imqdbmgr` susmentionnée pour convertir le magasin de données en vue d'une utilisation partagée.

Étant donné que cette commande prend uniquement en charge la conversion de magasins HADB, vous ne pouvez pas l'utiliser pour convertir des magasins basés sur des fichiers ou tout autre magasin JDBC en un magasin HADB partagé. Si vous utilisiez auparavant une version 3.x de Message Queue, vous devez créer un magasin HADB, puis migrer manuellement vos données vers ce magasin pour pouvoir utiliser la fonctionnalité de haute disponibilité.

- Ce processus de conversion, effectué à l'aide de la commande `imqdbmgr upgrade hastore`, peut échouer, en affichant le message « nombre de verrous trop élevé » si le magasin contient plus de 10 000 messages. (*ID de bogue 6588856)*).

  (*Solution*) Utilisez la commande suivante pour augmenter le nombre de verrous.

  ```
  hadbm set NumberOfLocks=<desiredNumber>
  ```

  Pour de plus amples informations, reportez-vous à la section « HADB Problems » du manuel *Sun Java System Application Server 9.1 Enterprise Edition Troubleshooting Guide*.

- Si plus de 500 messages distants sont validés dans une transaction, le courtier peut renvoyer l'erreur « HADB-E-12815 : espace de mémoire de table épuisé.» *(ID de bogue 6550483)*

  Pour de plus amples informations, reportez-vous à la section « HADB Problems » du manuel *Sun Java System Application Server 9.1 Enterprise Edition Troubleshooting Guide*.

- Dans un cluster de courtiers, un courtier place des messages en file d'attente sur une connexion distante non démarrée (*ID de bogue 4951010*).

  Solution : le consommateur recevra les messages une fois la connexion démarrée. Les messages seront renvoyés à un autre consommateur si la connexion du consommateur est fermée.

- Lorsque plusieurs messages sont consommés à partir d'un courtier distant dans une transaction, il est possible que le message d'erreur suivant soit consigné vers le courtier. Celui-ci n'est pas important et peut donc être ignoré :

```
[26/Jul/2007:13:18:27 PDT] WARNING [B2117]:
Échec de la réception du message à partir de
mq://129.145.130.95:7677/?instName=a&brokerSessionUID=3209681167602264320:
    ackStatus = NOT_FOUND(404)\
    Raison = Mise à jour de l’état de la transaction distante sur COMMITED(6) :
transaction 3534784765719091968 introuvable, celle-ci a déjà dû être validée.
```

```
    AckType = MSG_CONSUMED
    MessageBrokerSession = 3209681167602264320
    TransactionID = 3534784765719091968
        SysMessageID = 8-129.145.130.95(95:fd:93:91:ec:a0)-33220-1185481094690
        ConsumerUID = 3534784765719133952\par

[26/Jul/2007:13:18:27 PDT] WARNING Notifier la validation d'une transaction
[8-129.145.130.95(95:fd:93:91:ec:a0)-33220-1185481094690,
[consumer:3534784765719133952, type=NONE]]
TUID=3534784765719091968 got response:
com.sun.messaging.jmq.jmsserver.util.BrokerException:
 Mise à jour de l'état de la transaction distante sur COMMITED(6) :
  transaction 3534784765719091968 introuvable, celle-ci a déjà dû être validée.:
com.sun.messaging.jmq.jmsserver.util.BrokerException: Mise à jour de la transaction distante sur
COMMITED(6) : transaction 3534784765719091968 introuvable, celle-ci a déjà dû
 être validée.r
```

Ce message est consigné lors de la notification de validation au courtier de base du message pour les prochains messages de la transaction lorsque la valeur de la propriété `imq.txn.reapLimit` est faible par rapport au nombre de messages distants contenus dans une transaction. *(Bogue 6585449)*

Solution : pour ne pas recevoir ce message, augmentez la valeur de la propriété `imq.txn.reapLimit`.

Problèmes relatifs à JMX

Pour les plates-formes Windows, la méthode `getTransactionInfo` du MBean de contrôle du gestionnaire de transactions retourne des informations de transaction comportant des heures de création incorrectes (*ID de bogue 6393359*).

Solution : utilisez plutôt la méthode `getTransactionInfoByID` de ce MBean.

Prise en charge de SOAP

Deux problèmes principaux à prendre en compte pour la prise en charge de SOAP.

- À partir de la version 4.0 de Message Queue, la prise en charge des objets administrés par SOAP a été interrompue.
- Le développement de SOAP dépend de plusieurs fichiers : `SUNWjaf`, `SUNWjmail`, `SUNWxsrt` et `SUNWjaxp`. Dans la version 4.1 de Message Queue, ces fichiers sont uniquement disponibles lorsque Message Queue est exécuté avec JDK version 1.6.0 ou version supérieure.

Fichiers redistribuables

Sun Java System Message Queue 4.1 contient l'ensemble de fichiers ci-dessous pouvant être utilisés et distribués librement sous forme binaire :

`fscontext.jar`
`imq.jar`
`imqjmx.jar`
`imqxm.jar`
`jaas.jar`
`jms.jar`
`libmqcrt.so` (HPUX)
`libmqcrt.so` (UNIX)
`mqcrt1.dll` (Windows)

Vous pouvez également redistribuer les fichiers `LICENSE` et `COPYRIGHT`.

Fonctions d'accessibilité destinées aux personnes handicapées

Pour obtenir la liste des fonctions d'accessibilité mises à disposition depuis la publication de ce média, consultez les évaluations de produit de la Section 508, disponibles sur demande auprès de Sun, afin de déterminer les versions les mieux adaptées au déploiement des solutions accessibles. Les mises à jour des applications sont disponibles à l'adresse `http://sun.com/software/javaenterprisesystem/get.html`.

Pour plus d'informations sur l'engagement de Sun en matière d'accessibilité, consultez le site suivant : `http://sun.com/access`.

Comment signaler des problèmes et apporter des commentaires

Si vous rencontrez des problèmes avec Sun Java System Message Queue, contactez le service clientèle Sun de l'une des manières suivantes :

- Services de support logiciel en ligne : `http://www.sun.com/service/sunone/software`.

  Ce site contient des liens vers la base de savoir, le centre d'assistance en ligne et ProductTracker, ainsi que vers des programmes de maintenance et des coordonnées pour l'assistance.
- Le numéro de téléphone indiqué sur votre contrat de maintenance.

Afin de vous aider au mieux à résoudre vos problèmes, nous vous suggérons de réunir les informations suivantes lorsque vous contactez le support technique de Sun :

- Description du problème, notamment les conditions dans lesquelles le problème se produit et sa répercussion sur l'opération effectuée.

- Le type de machine, les versions du système d'exploitation et du produit, y compris les patchs et autres logiciels pouvant avoir un lien avec le problème.
- Étapes détaillées des méthodes utilisées pour reproduire le problème.
- Journaux des erreurs ou core dumps éventuels.

Forum de Sun Java System

Accédez au forum de Sun Java System Message Queue à partir de l'adresse suivante :

`http://swforum.sun.com/jive/forum.jspa?forumID=24`

Votre participation est la bienvenue.

Forum sur la technologie Java

Il existe un forum JMS au sein des forums sur la technologie Java qui peut être utile.

`http://forum.java.sun.com`

Vos commentaires sont les bienvenus

Dans le souci d'améliorer notre documentation, nous vous invitons à nous faire parvenir vos commentaires et vos suggestions.

Pour nous faire part de vos commentaires, rendez-vous sur le site `http://docs.sun.com`, puis cliquez sur Envoyer des commentaires. Dans le formulaire en ligne, indiquez le titre et le numéro du document. Le numéro de référence est constitué de sept ou neuf chiffres et figure sur la page de titre du manuel ou en haut du document. Par exemple, le titre de ce document est Notes de version de Sun Java System Message Queue 4.1, et son numéro de référence est 820-3190-10.

Ressources Sun supplémentaires

Vous pouvez obtenir des informations utiles concernant Sun Java System sur les sites Internet suivants :

- Documentation

  `http://docs.sun.com/prod/java.sys`
- Services professionnels

  `http://www.sun.com/service/sunps/sunone`

- Produits et services logiciels

  `http://www.sun.com/software`
- Services de support logiciel

  `http://www.sun.com/service/sunone/software`
- Base de connaissances et support

  `http://www.sun.com/service/support/software`
- Services de formation et de support Sun

  `http://training.sun.com`
- Services professionnels et de conseil

  `http://www.sun.com/service/sunps/sunone`
- Informations pour les développeurs

  `http://developers.sun.com`
- Services de support pour développeurs Sun

  `http://www.sun.com/developers/support`
- Formation sur les logiciels

  `http://www.sun.com/software/training`