Sun Java™ System Message Queue Notes de mise à jour

Version 3.5 SP1

Numéro de document 817-7200-10

Ces notes de mise à jour contiennent des informations importantes disponibles au moment du lancement de la version 3.5 SP1 de Sun Java™ System Message Queue (anciennement Sun™ ONE Message Queue). Ce document contient également les notes de mise à jour vers Message Queue 3.5, destinées aux clients mettant leur version à niveau à partir des versions antérieures à la 3.5. Vous y trouverez des renseignements sur les nouvelles fonctions, les améliorations, les restrictions et problèmes connus, les notes techniques, etc. concernant les versions Message Queue 3.5.

Vous trouverez la dernière version de ces notes de mise à jour sur le site Web de la documentation relative à Sun Java System : `http://docs.sun.com/coll/MessageQueue_35_SP1`. Consultez ce site Web avant d'installer et de configurer votre logiciel, puis régulièrement pour vous procurer les manuels et les notes de mise à jour les plus récents.

Ces notes de mise à jour contiennent les sections suivantes :

Historique des mises à jour

Tableau 1 Historique des mises à jour

Date	Description des modifications
12 mars 2004	Mise à jour des informations concernant les bogues. Mise à jour de la section « Problèmes et limites connus ».
	Ajout de la section « Fichiers redistribuables ». Mise à jour de la section « Mises à jour de la documentation ». Mise à jour de la section « Problèmes de compatibilité ». Mise à jour de la section « Informations relatives à Sun Java System ».
9 janvier 2004	Mise à jour des informations concernant la prise en charge de PointBase version 4.8 ; mise à jour des informations liées aux fonctions C-API.

À propos de Message Queue 3.5 SP1

Message Queue 3.5 SP1 est une mise à jour de Message Queue 3.5 ; elle contient toutes les nouvelles fonctions de Message Queue 3.5. En outre, Message Queue 3.5 SP1 propose des corrections de bogues et porte un nouveau nom commercial. En effet, ce produit appartient désormais à la famille de produits Sun Java™ System.

Message Queue 3.5 SP1 a été reconnu conforme à la spécification Java™ Message Service (JMS) 1.1, après avoir satisfait aux tests CTS (Compatibility Test Suite) de JMS 1.1.

Cette section présente les modifications présentes dans Message Queue 3.5 SP1 ainsi que dans la version précédente, Message Queue 3.5.

Message Queue 3.5 SP1

Message Queue 3.5 SP1 propose des corrections pour certains bogues ; le nom commercial du produit et de la documentation a changé.

Message Queue 3.5

Message Queue 3.5 comprenait de nombreuses nouvelles fonctions :

Ces fonctions sont présentées dans les sous-sections suivantes.

Prise en charge de Client C (Enterprise Edition)

Message Queue 3.5 propose une prise en charge de l'exécution des fonctions C-API et C (ci-après désignée par le terme fonction Client C). La fonction client C permet d'intégrer d'anciens systèmes au système de messagerie Message Queue. Il s'agit d'une implémentation quasi totale de la spécification JMS. Elle prend en charge toutes les fonctions JMS, à l'exception de : certains types de corps (carte, flux et objet), de fonctions de navigateur de file d'attente, ainsi que certaines fonctionnalités de serveur d'applications J2EE (telles que les transactions distribuées et les objets `ConnectionConsumer`).

La prise en charge de la fonction Client C se fait par le biais d'un jeu de bibliothèques installées séparément, qui est activé uniquement grâce à une licence Enterprise Edition. Par conséquent, une mise à niveau de Platform Edition vers Enterprise Edition nécessite l'installation du fichier de licence d'Enterprise Edition et des bibliothèques C.

Les clients détenant Platform Edition qui activent la version d'essai de 90 jours de la licence Enterprise Edition peuvent utiliser la fonction client C s'ils demandent à Sun le kit du développeur de l'API C, par le biais de l'alias `imq-feedback@sun.com`. Le service ingénierie est chargé de répondre à ces demandes et de mettre à disposition le kit du développeur de l'API C sur le site FTP anonyme. À l'issue des 90 jours de la licence Enterprise Edition, les clients peuvent continuer à créer des fonctions clients C, mais ils ne peuvent pas les connecter au courtier de l'Enterprise Edition.

La fonction client C nécessitant des versions de compilateur spécifiques sur les différentes plates-formes de système d'exploitation, Enterprise Edition contient de nouvelles exigences en matière de configuration (voir le *Guide d'installation de Message Queue* pour plus d'informations). La fonction client C est aussi dépendante des bibliothèques Netscape Portable Runtime (NSPR) et Network Security Service (NSS). (En ce qui concerne Message Queue 3.5, la fonction client C a été testée avec succès sur Linux Red Hat Advanced Server 2.1. Les versions des bibliothèques NSPR et NSS sur lesquelles elle a été testée ne sont pas certifiées pour cette édition de Linux).

À l'heure actuelle, l'API C ne prend pas en charge le type d'authentification `basic`. Si vous configurez le courtier de sorte qu'il utilise ce type d'authentification, un appel à la fonction `MQCreateConnection` échoue ; le résultat est `MQ_UNSUPPORTED_AUTH_TYPE`.

La documentation de la fonction client C est composée de documentation de référence, de documentation de programmation et d'exemples de clients API C. Pour plus d'informations, voir le *Guide du développeur Message Queue Client C*.

Reprise après incident de la connexion au client Java (Enterprise Edition)

Message Queue 3.5 propose une fonction de reconnexion automatique améliorée, par laquelle une connexion qui a échoué peut être rétablie non seulement dans le courtier d'origine, mais aussi sur un courtier différent (reprise après incident de la connexion au client). La nouvelle connexion se fait au service de messagerie, plutôt qu'à une instance de courtier spécifique. Pour mettre en œuvre ce comportement, il vous suffit de configurer l'objet administré de fabrique de connexion (Message Queue 3.5 possède un nouveau schéma de spécification d'adresses de service de messagerie), en précisant un ensemble d'adresses de courtier (`imqAddressList`). Lorsque l'exécution du client doit établir (ou ré-établir) une connexion à un service de messagerie, elle essaie de se connecter aux courtiers de la liste par ordre de priorité, jusqu'à ce qu'elle trouve, ou non, un courtier disponible. Vous pouvez préciser le nombre limite de tentatives de connexion (`imqAddressListIterations`) à chacun de ces courtiers, ainsi que le laps de temps entre ces tentatives (`imqAddressListInterval`).

Si la reconnexion automatique se fait à une instance de courtier différente du courtier d'origine, les messages permanents et autres informations liées à l'état se trouvant dans le courtier en échec (ou déconnecté) peuvent être perdus. La raison en est que les diverses instances de courtier d'un cluster n'utilisent pas un magasin persistant partagé largement disponible. Cependant, la fonction de reconnexion automatique de l'exécution du client à une instance de courtier différente vous permet de créer des scénarios de récupération dans lesquels un courtier de remplacement ou un cluster de courtiers peut être utilisé dans le cadre d'une protection en cas de reprise après incident (incomplète).

De plus, si la fonction de reconnexion automatique est activée, Message Queue 3.5 rend désormais persistantes des destinations *temporaires* en cas d'échec de la connexion associée, car les clients peuvent se reconnecter et y accéder à nouveau. Les destinations temporaires sont traitées comme d'autres destinations physiques. Par conséquent, vous devez purger régulièrement un courtier de toutes les destinations temporaires inutilisées.

Pour plus d'informations, voir le *Guide du développeur Message Queue Client Java*.

Message Queue proposait déjà une fonction de reconnexion automatique qui permettait à l'exécution du client de se reconnecter automatiquement à un courtier en cas d'interruption de la connexion. Toutefois, elle ne fonctionnait pas dans les cas où il était impossible de restaurer complètement l'état côté client dans le courtier lors de la reconnexion (par exemple, lors de l'utilisation de sessions établies par le biais d'une transaction ou de destinations temporaires, qui existent uniquement pour la durée d'une connexion).

Meilleur contrôle du flux de messages du courtier

Des améliorations ont été apportées au courtier, afin de mieux contrôler le flux de messages vers les destinations et d'éviter des situations dans lesquelles la production de messages est beaucoup plus rapide que leur consommation. (En outre, d'autres nouvelles fonctions de Message Queue 3.5 peuvent aider à éliminer les goulots d'étranglement dans le flux des messages *sortant* des destinations. Voir les sections « Meilleur contrôle du flux du client Java » à la page 6 et « Meilleures stratégies de livraison de file d'attente (Enterprise Edition) » à la page 9).

Les améliorations apportées au flux de messages du courtier sont les suivantes :

- Limitation du nombre de producteurs associés à une destination. Les destinations possèdent désormais un nouvel attribut `maxNumProducers` ; lorsque cette limite est atteinte, il devient impossible de créer un producteur pour la destination concernée.
- Mise en place de nouvelles limites configurables et de comportements de limites lorsque les destinations atteignent les limites `maxTotalMsgBytes` et `maxNumMsgs`. En particulier, les changements suivants ont été apportés :
  - L'extension des attributs de destination `maxTotalMsgBytes` et `maxNumMsgs` aux destinations de rubriques (elles ne s'appliquaient jusqu'ici qu'aux destinations de file d'attente)
  - La possibilité de définir les attributs `maxTotalMsgBytes` et `maxNumMsgs` des *destinations créées automatiquement*

- La possibilité pour un administrateur de choisir un comportement si les limites ci-dessus (Enterprise Edition) sont atteintes. Vous pouvez préciser les comportements suivants : affichage des producteurs (`FLOW_CONTROL`), mise au rebut des messages les plus anciens (`REMOVE_OLDEST`), mise au rebut des messages dont la priorité est la plus basse, en fonction de leur ancienneté (`REMOVE_LOW_PRIORITY`) et/ou mise au rebut des messages les plus récents (`REJECT_NEWEST`).
- La mise en œuvre du comportement en cas d'atteinte de la limite de contrôle du flux du producteur (`FLOW_CONTROL`) *par producteur*, plutôt que par connexion. (La mise en œuvre précédente fermait tous les producteurs d'une connexion lorsqu'une destination avait reçu un trop grand nombre de messages par le biais de cette connexion). Le contrôle de flux par producteur ferme uniquement les producteurs d'une connexion associée à la destination inondée, ce qui permet à d'autres producteurs de la connexion de continuer à envoyer des messages à d'autres destinations.

- La possibilité pour un administrateur d'interrompre (et de reprendre) une destination spécifique. Vous pouvez interrompre la livraison de messages provenant des producteurs vers la destination ou de la destination vers les consommateurs, ou des deux. Pour ce faire, vous pouvez utiliser deux nouvelles sous-commandes `imqcmd` : `pause` et `resume`, illustrées dans l'exemple suivant :
  - `imqcmd pause dst -n myQueue -t q -pst PRODUCERS`
  - `imqcmd resume dst -n myQueue -t q`

Pour plus d'informations, voir le *Guide d'administration de Message Queue*.

Meilleur contrôle du flux du client Java

L'exécution du client Message Queue 3.5 gère le flux de messages *par consommateur*, ainsi que par connexion. Vous pouvez limiter le nombre de messages placés dans la mémoire tampon par consommateur, empêchant ainsi qu'un consommateur soit submergé par d'autres consommateurs. Cela signifie également que dans le cas d'une livraison de file d'attente à plusieurs consommateurs, vous pouvez mieux équilibrer la livraison des messages. Vous pourrez aussi gérer les ressources en matière de mémoire dans l'exécution du client de Message Queue.

Un nouvel attribut de fabrique de connexion, `imqConsumerFlowLimit`, limite le nombre de messages mis en mémoire tampon *par consommateur* pour tous les consommateurs partageant une connexion. Lorsque le nombre de messages se trouvant dans la mémoire tampon d'un consommateur devient inférieur à un pourcentage seuil (`imqConsumerFlowThreshold`) de `imqConsumerFlowLimit`, le courtier peut livrer un autre lot de messages à l'exécution du client pour qu'il soit consommé par ce consommateur. Si le nombre total de messages mis en mémoire tampon pour tous les consommateurs dans une connexion dépasse la valeur fixée pour `imqConnectionFlowLimit`, la livraison des messages par le biais de la connexion cesse jusqu'à ce que le total devienne inférieur à la limite de connexion.

(La mise en œuvre précédente du contrôle du flux d'exécution vous permettait de limiter le nombre de messages placés dans la mémoire tampon de l'exécution du client, avant d'être consommés (`imqConnectionFlowLimit`). Cette fonction avait pour objectif de limiter la quantité de mémoire du client utilisée pour la mise en mémoire tampon de messages, de sorte que les clients lents à consommer ne tombent pas en panne à cause du manque de mémoire. Cette fonction était mise en œuvre au niveau de la connexion, ce qui signifie que si une connexion prend en charge un grand nombre de clients, une avalanche de messages destinées à un consommateur peut empêcher les autres consommateurs de recevoir des messages).

Pour plus d'informations, voir le *Guide du développeur Message Queue Client Java*.

Nouveaux paramètres de destination

Message Queue 3.5 propose un suivi amélioré des messages et des consommateurs par destination, afin de permettre une meilleure surveillance et un meilleur contrôle de la mémoire et de son utilisation.

Les nouveaux paramètres apparaissent dans la sortie de la nouvelle sous-commande `imqcmd metrics dst`. Cette commande affiche les totaux cumulés (depuis le début de l'échantillonnage), les valeurs actuelles, les valeurs moyennes (calculées par rapport aux échantillons prélevés) et les valeurs maximales (depuis le début de l'échantillonnage) correspondant aux paramètres de message et de consommateur.

Par exemple, la commande `imqcmd metrics dst -m ttl` renvoie les informations suivantes :

- flux de messages
  - flux de messages entrants : total cumulé, débit ;
  - flux de messages sortants : total cumulé, débit.
- Messages stockés dans le courtier (messages sans accusé de réception dans la mémoire ou dans un magasin persistant)
  - nombre de messages : actuel, maximal, moyen ;
  - taille des messages : actuelle, maximale, moyenne.
- Message le plus volumineux, en octets

La commande `imqcmd metrics dst -m con` renvoie les informations suivantes :

- nombre de consommateurs actifs : actuel, maximal, moyen (voir « Meilleures stratégies de livraison de file d'attente (Enterprise Edition) » à la page 9) ;
- nombre de consommateurs de remplacement : actuel, maximal, moyen (voir « Meilleures stratégies de livraison de file d'attente (Enterprise Edition) » à la page 9).

Pour plus d'informations, voir le *Guide d'administration de Message Queue*.

API de contrôle à distance (Enterprise Edition)

Message Queue 3.5 propose une API message par le biais de laquelle les clients JMS distants (ou locaux) peuvent facilement contrôler et analyser les paramètres du courtier. Cette API repose sur la capacité qu'a le courtier à produire des messages contenant les paramètres liés au courtier, à la machine virtuelle Java et aux destinations individuelles (voir « Nouveaux paramètres de destination » à la page 7). Ces messages sont envoyés à des destinations de rubriques spécifiques, selon l'entité en cours de contrôle, chaque fois qu'un ou que plusieurs consommateurs sont abonnés à ces destinations. Le client consommateur peut alors extraire les messages, les filtrer grâce à une propriété d'en-tête (`type`), puis extraire les paramètres qu'ils contiennent.

Pour plus d'informations, voir le *Guide d'administration de Message Queue* et le *Guide du développeur Message Queue Client Java*.

(Message Queue prenait en charge uniquement l'enregistrement des données de paramètres de courtier et les requêtes à distance sur les paramètres à l'aide de ses outils d'administration. Ces fonctions, même si elles fournissaient des données de paramètres importantes, n'en facilitaient pas l'analyse).

Message Queue Adaptateur de ressources pour JMS (prise en charge du serveur d'applications J2EE)

Message Queue 3.5 propose un adaptateur de ressources JMS permettant de connecter le service de messagerie JMS de Message Queue à un serveur d'applications J2EE compatible.

Un adaptateur de ressources est un dispositif normalisé permettant d'apporter des fonctionnalités complémentaires à un serveur d'applications J2EE (en le connectant à un EIS, un système de messagerie ou autre), selon la spécification concernant l'architecture des connecteurs J2EE (JCA 1.5). Cette architecture permet à n'importe quel serveur d'applications J2EE, par exemple, de prendre en charge la messagerie JMS en se connectant à un fournisseur JMS implémentant la spécification JCA 1.5 : les composants J2EE déployés et exécutés dans l'environnement de serveur d'applications peuvent échanger des messages JMS par le biais du fournisseur JMS connecté (exécution du client et serveur).

Pour plus d'informations, voir le *Guide d'administration de Message Queue*.

Personnalisation de l'accusé de réception d'un message

Message Queue prend actuellement en charge le mode d'accusé de réception client JMS `CLIENT_ACKNOWLEDGE`, grâce auquel un client JMS reconnaît explicitement la consommation de messages. En mode `CLIENT_ACKNOWLEDGE`, le client appelle la méthode `acknowledge()` d'un objet de message. Cela a pour conséquence que la session accuse réception de tous les messages qu'elle a consommés depuis l'invocation précédente de la méthode.

Message Queue 3.5 améliore ce comportement en vous permettant d'accuser réception de messages *individuels*. Cela signifie que vous pouvez accuser réception d'un seul message spécifique, plutôt que de tous les messages consommés jusque-là en tant que lot. Cela se fait par le biais du code, en définissant le type de l'objet de message comme étant un type de message Message Queue spécial, sur lequel vous appelez une nouvelle méthode `acknowledge()`. Cela vous permet de vous écarter de la norme JMS pour pouvoir gérer des besoins applicatifs spéciaux.

Pour plus d'informations, voir le *Guide du développeur Message Queue Client Java*.

Meilleures stratégies de livraison de file d'attente (Enterprise Edition)

La mise en œuvre de la livraison de file d'attente à plusieurs consommateurs, anciennement mise en place sous la forme de trois stratégies différentes (simple, bascule et circulaire), a changé. Message Queue 3.5 fait appel à une méthode plus générale, selon laquelle la charge de chaque livraison est équilibrée sur un nombre de consommateurs actifs (et de remplacement) que vous pouvez configurer. La mise en œuvre de Message Queue 3.5 s'appuie sur les nouveaux attributs de destination suivants :

- `maxNumActiveConsumers` : précise le nombre de consommateurs (un ou plusieurs) actifs dans une livraison de file d'attente dont la charge est équilibrée.
- `maxNumBackupConsumers` : précise le nombre de consommateurs de remplacement (aucun ou plusieurs) pouvant remplacer les consommateurs actifs si ceux-ci font défaut.

(Les nouveaux consommateurs seront rejetés si le nombre de consommateurs dépasse la somme de ces deux attributs.)

L'édition Platform Edition de Message Queue prend en charge la livraison de file d'attente dont la charge est équilibrée à concurrence de deux consommateurs. L'édition Enterprise Edition, quant à elle, prend en charge un nombre illimité de consommateurs.

Le nouveau mécanisme d'équilibrage des charges tient compte du taux de consommation de messages des différents consommateurs. Il fonctionne de la manière suivante :

- Un nombre initial de messages placés en file d'attente dans une destination est acheminé vers des consommateurs actifs disponibles (dans l'ordre dans lequel ils sont inscrits auprès de la destination) dans des lots de taille configurable (attribut `consumerFlowLimit` de la destination). Une fois ces messages livrés, d'autres messages parvenant à une destination sont acheminés vers les consommateurs un par un, à mesure que ces derniers deviennent disponibles (c'est-à-dire, lorsque les consommateurs accusent réception de tous les messages qui leur ont été livrés). Si un consommateur actif fait défaut, le premier consommateur de remplacement devient actif, puis reprend le travail du consommateur faisant défaut.

- Dans un environnement de cluster de courtiers, vous pouvez définir le mécanisme de livraison de manière qu'il attribue des priorités aux consommateurs locaux. Un nouvel attribut de destination, `localDeliveryPreferred`, vous permet de préciser que les messages doivent être livrés aux consommateurs distants uniquement si aucun consommateur n'est présent dans le courtier local (courtier sur lequel la destination a été créée). Cela vous permet d'augmenter les performances dans les cas où l'acheminement vers les clients distants (par le biais de leurs courtiers locaux respectifs) pourrait ralentir le rendement. (Cet attribut nécessite que l'étendue de la livraison ne soit pas restreinte à la livraison locale. Voir « Amélioration des performances du cluster (Enterprise Edition) » à la page 10).

Pour plus d'informations, voir le *Guide d'administration de Message Queue*.

Amélioration des performances du cluster (Enterprise Edition)

Dans un environnement de clusters de courtiers, les destinations sont répliquées sur tous les courtiers et tous les messages livrés à ces destinations sont envoyés à tous les courtiers contenant des consommateurs inscrits pour ces destinations, même si seul un pourcentage infime des messages est livré à un consommateur donné (par exemple, dans le cas d'un abonné durable utilisant des critères de sélection, ou d'un récepteur de file d'attente impliqué dans une livraison de file d'attente dont la charge est équilibrée). Ce trafic de courtier à courtier peut engendrer des avalanches de messages, plus particulièrement lorsqu'un nouveau consommateur devient actif. Message Queue 3.5 apporte les améliorations suivantes, destinées à réduire le surplus de trafic de courtier à courtier dans un cluster :

- L'adoption de nouveaux mécanismes de contrôle de flux qui régulent la livraison des messages à une connexion de consommateur. En d'autres termes, le *consommateur* régule la livraison des messages (de la destination vers l'exécution du client), évitant ainsi la transmission des messages superflus de courtier à courtier. (Ces mécanismes doivent également permettre d'éviter les engorgements du côté de l'exécution du client ; voir « Meilleur contrôle du flux du client Java » à la page 6).
- La modification de la mise en œuvre de la livraison de file d'attente à plusieurs consommateurs (voir « Meilleures stratégies de livraison de file d'attente (Enterprise Edition) » à la page 9), afin de réduire la transmission superflue de messages de courtier à courtier. Cette mise en œuvre propose un nouvel attribut de destination de file d'attente, `localDeliveryPreferred`, qui vous permet de préciser que les consommateurs locaux sont prioritaires par rapport aux consommateurs distants lors de la livraison de file d'attente à plusieurs consommateurs (voir « Meilleures stratégies de livraison de file d'attente (Enterprise Edition) » à la page 9).

Pour plus d'informations, voir le *Guide d'administration de Message Queue*.

Destinations locales (Enterprise Edition)

Un nouvel attribut de destination, `isLocalOnly`, vous permet de préciser qu'une destination est limitée à la livraison de messages aux consommateurs locaux (consommateurs connectés au courtier sur lequel la destination est créée), par opposition aux consommateurs connectés à d'autres courtiers d'un cluster. De la même façon, cette destination peut recevoir uniquement les messages qui lui sont envoyés par des producteurs locaux. Cette propriété vous permet de créer des destinations indépendantes, non interactives, portant le même nom sur des courtiers différents d'un même cluster et d'établir des scénarios de type reprise après incident, dans lesquels un message est envoyé à deux destinations, au cas où l'une d'entre elles subirait un échec.

Pour plus d'informations, voir le *Guide d'administration de Message Queue*.

Sécurisation des clusters de courtiers (Enterprise Edition)

Dans les cas où il est indispensable que la livraison des messages entre client et serveur de messages soit sécurisée et cryptée, Message Queue 3.5 prend désormais en charge la livraison sécurisée de messages entre les courtiers d'un cluster. Pour parvenir à une livraison sécurisée et cryptée des messages au sein d'un cluster, vous devez configurer le service de connexion au cluster interne pour qu'il utilise un protocole de transport SSL.

Pour plus d'informations, voir le *Guide d'administration de Message Queue*.

Meilleures performances du magasin persistant

La mise en œuvre du magasin de données simple de Message Queue et du magasin de données conforme au JDBC a été modifiée dans Message Queue 3.5, afin d'en améliorer les performances. Ces améliorations sont présentées dans les deux sections suivantes. Pour plus d'informations, voir le *Guide d'administration de Message Queue*.

Persistance intégrée (magasin de données simple)

Les améliorations apportées aux performances du magasin de données simple de Message Queue impliquent des changements internes du format de données qui sont visibles seulement lors de la mise à niveau de Message Queue 3.01 (ou une version précédente) vers Message Queue 3.5.

La migration du magasin de données est gérée automatiquement lorsqu'une instance de courtier de Message Queue 3.5 est lancée pour la première fois et qu'elle référence une version précédente du magasin de fichiers. Une copie du magasin de fichiers précédent est enregistrée dans le répertoire d'instances. Vous devez la supprimer manuellement une fois la migration terminée. Pour supprimer automatiquement le magasin de fichiers précédent (si vous ne disposez pas de suffisamment d'espace disque pour contenir deux copies du magasin), vous pouvez démarrer le courtier en version 3.5 avec une option ajoutée, comme le montre la commande suivante :

```
imqbrokerd -upgrade-store-nobackup
```

(Il n'y a aucun espace dans `upgrade-store-nobackup`).

La racine du nouveau magasin de données simple a changé de :

…/instances/*instanceName*/filestore/

en :

…/instances/*instanceName*/fs350/.

En outre, l'utilitaire de commande de Message Queue (imqcmd) a été amélioré de manière à fournir les paramètres de magasin de fichiers suivants :

```
imqcmd metrics dst -n destName -t type -m dsk
```

Et imqcmd propose une nouvelle commande de compactage :

```
imqcmd compact dst -n destName -t type
```

Persistance intégrée (Magasin de données conforme au JDBC™)

Le magasin de données conforme au JDBC de Message Queue a été modifié dans Message Queue 3.5 pour prendre en charge les améliorations apportées à la gestion de la mémoire du courtier (voir « Meilleur contrôle du flux de messages du courtier » à la page 5 et « Nouveaux paramètres de destination » à la page 7) ainsi qu'une gamme plus étendue de fournisseurs de bases de données. Les modifications apportées pour améliorer la gestion de la mémoire (classement des messages par destination) sont des changements de schéma gérés de manière transparente. Toutefois, la prise en charge d'autres fournisseurs de bases de données implique des changements de configuration de la persistance intégrée.

La migration du magasin conforme au JDBC est gérée automatiquement lorsqu'une instance de courtier de Message Queue 3.5 est lancée pour la première fois avec une version précédente du magasin persistant. Les anciennes tables sont néanmoins conservées intactes. Elles doivent être supprimées manuellement à l'aide d'une nouvelle commande `imqdbmgr delete oldtbl` une fois la migration terminée. Pour supprimer les anciennes tables automatiquement, au cas où l'espace serait insuffisant pour contenir deux copies du magasin, vous pouvez utiliser une nouvelle option lors du premier lancement du courtier : `imqbrokerd -upgrade-store-nobackup`.

Pour ce qui est de la prise en charge d'autres bases de données, il était auparavant impossible de personnaliser les instructions SQL envoyées à la base de données conforme au JDBC que vous vouliez intégrer (les instructions SQL étaient générées en interne). Message Queue 3.5 propose de nouvelles propriétés de configuration d'instances qui vous permettent de personnaliser le code SQL qui crée le schéma de base de données de Message Queue. Il existe une propriété configurable pour chaque table de la base de données : la propriété est le code SQL qui crée la table. Ces propriétés sont nécessaires pour préciser correctement les types de données utilisés par la base de données intégrée. Les exemples proposés s'appuient sur une base de données intégrée PointBase, plutôt que sur l'ancienne base de données Cloudscape.

Authentification et autorisation spécifiques d'une instance

Par défaut, Message Queue 3.5 prévoit que chaque instance possède son propre référentiel utilisateur basé sur les fichiers et son propre fichier de contrôle d'accès, tous deux contenus dans un emplacement standard : …/`instances`/*instanceName*/`etc`/. Ces deux fichiers sont créés au premier démarrage d'une instance de courtier. Si le courtier trouve ces fichiers à l'ancien emplacement (ce qui se produit généralement lors d'une mise à niveau à partir d'une version précédente), il copie les fichiers vers l'emplacement spécifique de l'instance. En revanche, s'il ne les trouve pas à l'ancien emplacement (ce qui se produit généralement lors d'une nouvelle installation), il place des versions par défaut de ces fichiers à un emplacement spécifique de l'instance.

Afin de prendre en charge les référentiels utilisateur spécifiques des instances, une option -i *instanceName* a été ajoutée au gestionnaire des utilisateurs (`imqusermgr`) afin de préciser le référentiel utilisateur spécifique de l'instance auquel la commande `imqusermgr` s'applique.

Pour plus d'informations, voir le *Guide d'administration de Message Queue*.

Jusqu'ici, toutes les instances d'un courtier sur un ordinateur partageaient par défaut le même référentiel utilisateur basé sur les fichiers (et par conséquent le même mot de passe de connexion client), ainsi que le même fichier de contrôle d'accès. Vous pouviez toutefois configurer toutes les instances de courtier pour qu'elles utilisent un emplacement de référentiel utilisateur LDAP spécifique ou un fichier de contrôle d'accès spécifique, les deux étant identifiés dans le fichier de configuration de l'instance.

Installation sous Linux basée sur RPM

L'installation sous Linux de Message Queue 3.5 s'effectue à l'aide de RPM (Red Hat Package Manager), un système de gestion de paquets piloté par ligne de commande, capable d'installer, de désinstaller, de vérifier, d'interroger et de mettre à jour les logiciels (RPM).

En outre, la structure de répertoires installée pour Message Queue sous Linux a changé, pour correspondre aux emplacements standard utilisés pour les produits non fournis en standard avec Linux. (Les plates-formes Solaris™ et Linux possèdent des normes qui dépendent du fait qu'un produit est fourni ou non en standard avec le système d'exploitation.) En particulier, il n'existe plus de répertoire d'installation racine de Message Queue sous Linux, tout comme sous Solaris.

Pour plus d'informations, voir le *Guide d'installation de Message Queue*.

Prise en charge du système d'exploitation Solaris, X86 Platform Edition

Sous Solaris 9, Message Queue 3.5 est pris en charge par les processeurs X86 et SPARC.

Configurations matérielle et logicielle requises

Les configurations matérielle et logicielle requises pour cette version, ainsi que les produits et plates-formes pris en charge, sont décrits en détail dans le *Guide d'installation de Message Queue*.

Bogues résolus

Cette section présente rapidement les bogues résolus :

- Le Tableau 2 , page 15 présente les bogues résolus dans Message Queue 3.5 SP1.
- Le Tableau 3 , page 16 présente les bogues résolus dans Message Queue 3.5.

Pour obtenir les listes plus anciennes de bogues résolus, voir :

- Pour Message Queue 3.0.1 Service Pack 2, voir les Notes de mise à jour de *Message Queue 3.0.1 Service Pack 2* à l'adresse suivante :

  `http://docs.sun.com/coll/S1_MessageQueue_301_SP2`

- Pour Message Queue 3.0.1, voir les Notes de mise à jour de *Message Queue 3.0.1* à l'adresse suivante :

  `http://docs.sun.com/coll/S1_MessageQueue_301`

- Pour Message Queue 3.0, voir les Notes de mise à jour de *Message Queue 3.0* à l'adresse suivante :

  `http://docs.sun.com/coll/S1_MessageQueue_30`

Pour obtenir un rapport complet sur un bogue, consultez le site Java Developer Connection à l'adresse

`http://developer.java.sun.com/developer/bugParade`

Résolu dans Message Queue 3.5 SP1

Le Tableau 2 , page 15 présente les bogues résolus dans Message Queue 3.5 SP1. (Le Tableau 3 , page 16 présente les bogues résolus dans Message Queue 3.5).

Tableau 2 Bogues résolus dans Message Queue 3.5 SP1

Référence	Description
4942723	Le courtier peut manquer de mémoire lors de l'envoi de messages volumineux à l'aide de l'option de pool de threads partagés.
4944894	Le courtier peut occasionnellement générer une erreur `CancelledKeyException` lors de l'utilisation du pool de threads partagés.
4947239	La création et la clôture répétées de producteurs augmentent légèrement l'occupation de la mémoire du client.
4947993	Impossible de détruire la destination ou la ressource durable avec un abonné durable actif.
4948525	Des nombres négatifs peuvent s'afficher dans les paramètres de sorties de taille des messages en entrée et sortie. Cela se produit lorsque plus de 2 143 510 810 octets ont été envoyés.
4948563	Conversion de paquets : un message INFO s'affiche sur chaque message 2.0 SP1 envoyé à un courtier 3.5. Chaque fois qu'un client 2.0 SP1 envoie un message à un courtier 3.5, le message de niveau INFO suivant s'affiche : `[04/Nov/2003:10:34:16 PST] Internal Error: Unknown ProducerUID 0`
4949781	Erreur d'impossibilité d'utilisation du diffuseur de cluster au démarrage du courtier.
4952332	Les messages peuvent être livrés de manière erronée si le consommateur principal subit un échec et qu'un consommateur de remplacement prend le relais sur la même connexion.
4956748	Impossible d'utiliser un courtier principal avec la base de données Oracle.
4964703	API C : L'en-tête `MESSAGE_ID` renvoyé par la fonction `MQGetMEssageHeaders()` n'est pas doté du préfixe " ID: "
4964712	API C : L'en-tête `MESSAGE_ID` défini par la fonction `MQSetMEssageHeaders()` n'est pas ignoré lors de l'envoi du message.
4969583	API C : Le même identificateur de message doit pouvoir appeler `MQAcknowledgeMessages()` plusieurs fois.
4983150	L'indicateur `JMSRedlivered` n'est pas défini lors du redémarrage du courtier et de la livraison du message.
4983699	Exceptions de pertes de courtier émises par le magasin lorsqu'il ne parvient pas à stocker un message.

Résolus dans Message Queue 3.5

Le Tableau 3 présente les problèmes résolus dans Message Queue 3.5.

Tableau 3 Problèmes résolus dans Message Queue 3.5

Référence	Description
4449354	Exceptionnellement, l'appel simultané des méthodes `Connection.stop`, `Connection.start`, `Connection.close` et des méthodes `Session.recover` et `Session.rollback` (dans des threads distincts) peut aboutir à une commande de nouvelle livraison de message inattendue.
4630183	La destruction d'une destination laisse des abonnements durables dans le courtier.
4753010	Croissance illimitée du segment de tas natif du processus Java avec un serveur VM.
4761626	Une forte demande de création/suppression avec des files d'attente créées automatiquement peut provoquer la perte de messages.
4855307	Le courtier ne peut pas effectuer d'authentification par rapport à un référentiel LDAP car la configuration par défaut utilise un ancien nom de propriété (`bindDN`)
4883126	La fonction de reconnexion automatique ne fonctionne pas correctement.
4888270	La retransmission d'un message envoyé à l'origine dans une transaction provoque une erreur du courtier.
4431924	`imqadmin`: les boîtes de dialogue modales qui exigent une réponse de l'utilisateur peuvent se bloquer. La console d'administration (`imqadmin`) utilise des boîtes de dialogue qui ont un comportement applicatif. La majorité de ces boîtes de dialogue s'affichent à la suite d'une interaction avec l'interface utilisateur graphique, par exemple en sélectionnant l'option de menu Ajouter courtier. Néanmoins, l'affichage d'une boîte de dialogue peut également résulter de la perte de connexion avec le courtier. Lorsque plusieurs boîtes de dialogue sont ouvertes, la console d'administration est verrouillée. Vous ne pouvez pas faire disparaître des boîtes de dialogue modales exigeant une réponse de l'utilisateur à l'aide du bouton Fermer.
4703406	QueueBrowser doit fonctionner sans appeler au préalable `connection.start()`. Connection.start() doit être appelé au cours d'une connexion avant que QueueBrowser puisse parcourir la file d'attente. En cas d'échec de l'appel de `Connection.start()` , l'énumération QueueBrowser se bloque sur `nextElement()` et peut déclencher une exception `java.util.NoSuchElementException`.
4866814	Sous Solaris, le courtier ne peut pas enregistrer les messages d'erreur et d'avertissement à l'aide de syslog s'il a été démarré avec une JVM 64 bits (le courtier est démarré à l'aide de `-vmargs -d64`). La raison de cet incident est que la version bêta de Message Queue ne contient pas de version 64 bits de notre bibliothèque, `libimqutil.so.1`.
4872121	Le courtier ne démarre pas sur un système non mis en réseau ne possédant pas d'adresse IP autre que 127.0.0.1.
4879902	Augmentation lente de mémoire dans le courtier.
4881968	Impossible de créer des clients de contrôle si la propriété `imq.autocreate.topic` est définie comme étant fausse.

Tableau 3 Problèmes résolus dans Message Queue 3.5 *(suite)*

Référence	Description
4884827	Échec des tests CTS1.3 MDB/EJB CMT avec Message Queue 3.5 & AppServer 7.0
4885654	Les producteurs peuvent échouer si un nouveau message est publié sur une destination créée automatiquement en même temps que le système récolte la destination.
4887506	Au cours d'une reprise après incident d'un consommateur principal vers un consommateur de remplacement, les messages peuvent être livrés de manière incorrecte.
4888939	Les clients C et Java d'une destination avec un comportement de FLOW_CONTROL peuvent cesser de recevoir des messages si la taille maximale d'une destination (`maxNumMsgs`) est très petite (< 5 messages).
4889002	La propriété imq.transaction.autorollback n'est pas prise en charge par la version bêta 3.5.
4891874	Le contrôle de flux consommateur peut provoquer l'arrêt de la livraison des messages aux consommateurs. Ce problème est plus susceptible de survenir avec le problème 4896133 : ConnectionConsumers et les beans pilotés par les messages du serveur d'applications 7 de Sun Java System.
4895262	Les clients HTTPS ne parviennent pas à se connecter au courtier par le biais de `HTTPSTunnelServlet`
4897500	Dans un cluster, lorsqu'un client appelle unsubscribe() pour supprimer un abonnement durable, il est seulement supprimé du courtier auquel le client est connecté. Cela signifie que les messages produits à l'attention des autres courtiers continueront d'être stockés pour cet abonné.
4898020	Impossible d'utiliser les courtiers de Message Queue 3.0.* et Message Queue 3.5 ensemble dans un cluster. Le démarrage d'un cluster mixte engendre une erreur dans le courtier 3.0.1 : Différence de configuration : Interruption de la connexion avec le courtier […] car les propriétés de configuration suivantes ne correspondent pas - null imq.queue.deliverypolicy
4888983	La commande `imqcmd list dur` n'affiche pas d'abonnés durables avec le même nom durable.

Informations importantes

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

- Notes relatives à l'installation
- Problèmes de compatibilité
- Mises à jour de la documentation

Notes relatives à l'installation

Consultez le *Guide d'installation de Message Queue* pour obtenir des informations relatives à la configuration requise, aux plates-formes logicielles et aux produits pris en charge, aux instructions préalables à l'installation, aux procédures de mise à jour, ainsi que d'autres informations pertinentes pour l'installation de Message Queue sur les plates-formes Solaris, Linux et Windows.

Problèmes de compatibilité

Cette section présente les problèmes de compatibilité entre Message Queue 3.5 SP1 et Message Queue 3.5.

Problèmes liés à la version principale à venir de Message Queue

Les changements incompatibles suivants pourraient être introduits dans la version principale à venir de Message Queue. Nous vous communiquons ces informations dès à présent pour que vous puissiez vous préparer à ces changements.

- La prise en charge du client Message Queue pour toutes les versions de J2SE 1.3 sera abandonnée. J2SE 1.4 sera toujours pris en charge.
- Toutes les interfaces de ligne de commande Message Queue seront modifiées pour supprimer l'option vous permettant de fournir un mot de passe en tant qu'argument de ligne de commande. Par exemple :

  ```
  imqbrokerd -ldappassword <passwd> imqcmd -p <passwd>
  ```

  D'autres options de précision du mot de passe seront proposées.
- Le format du fichier journal du courtier changera. Les applications dépendant du format actuel ne fonctionneront plus.
- L'emplacement des fichiers individuels installés avec Message Queue peut changer. Cela pourrait avoir une incidence sur les applications existantes qui dépendent de l'emplacement actuel de certains fichiers Message Queue.
- Le programme `imqkeytool` peut être supprimé du produit. L'utilitaire keytool J2SE le remplacera.
- Les clients Message Queue utilisant une version de Message Queue plus ancienne que la version principale à venir pourraient ne pas avoir accès aux nouvelles fonctions proposées dans cette version du produit.
- Aucune erreur n'est générée lorsque la fonction `MQAcknowledgeMessages()` est appelée par des clients C (à l'aide de `MQ_CLIENT_ACKNOWLEDGE`) sur un message dont il a déjà été accusé réception. Ce comportement pourrait changer.

Problèmes dans Message Queue 3.5

Message Queue 3.5 est généralement compatible avec Message Queue 3.0 (et les versions suivantes 3.0.1, 3.0.1 Service Pack 1, ainsi que 3.0.1 Service Pack 2). Cependant, les changements apportés aux propriétés du courtier, aux objets administrés, au schéma de persistance, aux emplacements de fichiers et aux outils d'administration peuvent avoir un impact sur une mise à niveau à partir des versions deMessage Queue 3.0 vers Message Queue 3.5.

L'opération d'installation de Message Queue 3.5 ne supprime pas ni n'écrase le répertoire `IMQ_VARHOME` de Message Queue 3.0. Ce répertoire contient des fichiers liés à la configuration et à la sécurité. La plupart de ces données sont compatibles avec Message Queue 3.5 et peuvent être conservées. Pour ce faire, suivez les instructions figurant dans le *Guide d'installation de Message Queue*.

Les problèmes auxquels vous pouvez être confronté lors de la mise à niveau de Message Queue 3.0 vers Message Queue 3.5 sont les suivants :

- Compatibilité du courtier
- Changements apportés aux propriétés et aux attributs
- Emplacement des fichiers publics .jar

Pour obtenir des informations concernant la compatibilité des objets administrés, des clients et des outils d'administration, consultez le *Guide d'installation de Message Queue*.

Compatibilité du courtier

Un courtier Message Queue 3.5 pourra interagir avec un courtier Message Queue 3.0. Cependant, des changements ont été apportés aux propriétés du courtier et au schéma du magasin persistant. Certaines données de Message Queue 3.0 restent compatibles avec Message Queue 3.5. Pour plus d'informations, voir le *Guide d'installation de Message Queue*.

Changements apportés aux propriétés et aux attributs

Cette section récapitule les changements apportés aux propriétés du courtier, aux attributs de destination et aux attributs de fabrique de connexion de Message Queue 3.5.

Propriétés du courtier Les tableaux suivants présentent les nouvelles propriétés, les propriétés abandonnées et les changements de nom de propriété dans Message Queue 3.5. Pour plus d'informations, voir le chapitre 2 du *Guide d'administration de Message Queue*.

Tableau 4 Nouvelles propriétés du courtier de Message Queue 3.5

Nom de la propriété	Référence à la fonction
`imq.persist.file.message.max_record.size`	« Meilleures performances du magasin persistant » à la page 11
`imq.persist.file.destination.message.filepool.limit`	« Meilleures performances du magasin persistant » à la page 11
`imq.metrics.topic.enabled`	« API de contrôle à distance (Enterprise Edition) » à la page 8
`imq.metrics.topic.interval`	« API de contrôle à distance (Enterprise Edition) » à la page 8
`imq.metrics.topic.persist`	« API de contrôle à distance (Enterprise Edition) » à la page 8
`imq.metrics.topic.timetolive`	« API de contrôle à distance (Enterprise Edition) » à la page 8
`imq.autocreate.destination.maxNumMsgs`	« Meilleur contrôle du flux de messages du courtier » à la page 5
`imq.autocreate.destination.maxTotalMsgBytes`	« Meilleur contrôle du flux de messages du courtier » à la page 5
`imq.autocreate.destination.maxBytesPerMsg`	« Meilleur contrôle du flux de messages du courtier » à la page 5
`imq.autocreate.destination.maxNumProducers`	« Meilleur contrôle du flux de messages du courtier » à la page 5
`imq.autocreate.queue.maxNumActiveConsumers`	« Meilleures stratégies de livraison de file d'attente (Enterprise Edition) » à la page 9
`imq.autocreate.queue.maxNumBackupConsumers`	« Meilleures stratégies de livraison de file d'attente (Enterprise Edition) » à la page 9
`imq.autocreate.queue.consumerFlowLimit`	« Meilleur contrôle du flux du client Java » à la page 6 et « Meilleures stratégies de livraison de file d'attente (Enterprise Edition) » à la page 9
`imq.autocreate.topic.consumerFlowLimit`	« Meilleur contrôle du flux du client Java » à la page 6
`imq.autocreate.queue.localDeliveryPreferred`	« Amélioration des performances du cluster (Enterprise Edition) » à la page 10
`imq.autocreate.destination.isLocalOnly`	« Destinations locales (Enterprise Edition) » à la page 11

Tableau 5 Propriétés du courtier abandonnées dans Message Queue 3.5

Nom de la propriété
`imq.persist.file.message.fdpool.limit`
`imq.persist.file.message.filepool.limit`
`imq.redelivered.optimization`
`imq.queue.deliverypolicy`

Les tableaux suivants présentent les nouveaux attributs de destination de Message Queue 3.5, ainsi que ceux qui ont été abandonnés. Pour plus d'informations, voir le chapitre 6 du *Guide d'administration de Message Queue*.

Tableau 6 Nouveaux attributs de destination de Message Queue 3.5

Type de destination	Nom d'attribut	Référence à la fonction
File d'attente et rubrique	`maxNumMsgs`	« Meilleur contrôle du flux de messages du courtier » à la page 5
File d'attente et rubrique	`maxTotalMsgBytes`	« Meilleur contrôle du flux de messages du courtier » à la page 5
File d'attente et rubrique	`limitBehavior`	« Meilleur contrôle du flux de messages du courtier » à la page 5
File d'attente et rubrique	`maxBytesPerMsg`	« Meilleur contrôle du flux de messages du courtier » à la page 5
File d'attente et rubrique	`maxNumProducers`	« Meilleur contrôle du flux de messages du courtier » à la page 5
File d'attente uniquement	`maxNumActiveConsumers`	« Meilleures stratégies de livraison de file d'attente (Enterprise Edition) » à la page 9
File d'attente uniquement	`maxNumBackupConsumers`	« Meilleures stratégies de livraison de file d'attente (Enterprise Edition) » à la page 9
File d'attente et rubrique	`consumerFlowLimit`	« Meilleur contrôle du flux du client Java » à la page 6 et « Meilleures stratégies de livraison de file d'attente (Enterprise Edition) » à la page 9
File d'attente uniquement	`localDeliveryPreferred`	« Amélioration des performances du cluster (Enterprise Edition) » à la page 10
File d'attente et rubrique	`isLocalOnly`	« Destinations locales (Enterprise Edition) » à la page 11

Tableau 7 Attributs de destination abandonnés dans Message Queue 3.5

Type de destination	Nom d'attribut
File d'attente	`QueueDeliveryPolicy`

Attributs de fabrique de connexion Les tableaux suivants présentent les nouveaux attributs de fabrique de connexion de Message Queue 3.5. Pour plus d'informations, voir le chapitre 4 du *Guide du développeur Message Queue Client Java*. Notez que les attributs de fabrique de connexion de la version 3.0 de Message Queue sont toujours pris en charge par Message Queue 3.5. Ils continueront de l'être jusqu'à la prochaine version principale du produit Message Queue.

Tableau 8 Nouveaux attributs de fabrique de connexion de Message Queue 3.5

Nom d'attribut	Référence à la fonction
`imqAddressList`	« Reprise après incident de la connexion au client Java (Enterprise Edition) » à la page 4
`imqAddressListBehavior`	« Reprise après incident de la connexion au client Java (Enterprise Edition) » à la page 4
`imqAddressListIterations`	« Reprise après incident de la connexion au client Java (Enterprise Edition) » à la page 4
`imqReconnectEnabled`	« Reprise après incident de la connexion au client Java (Enterprise Edition) » à la page 4
`imqReconnectAttempts`	« Reprise après incident de la connexion au client Java (Enterprise Edition) » à la page 4
`imqReconnectInterval`	« Reprise après incident de la connexion au client Java (Enterprise Edition) » à la page 4
`imqConsumerFlowLimit`	« Meilleur contrôle du flux du client Java » à la page 6
`imqConsumerFlowThreshold`	« Meilleur contrôle du flux du client Java » à la page 6

Tableau 9 Attributs de la fabrique de connexion renommés

Nom précédent	Nom dans Message Queue 3.5
`imqFlowControlCount`	`imqConnectionFlowCount`
`imqFlowControlIsLimited`	`imqConnectionFlowLimitEnabled`
`imqFlowControlLimit`	`imqConnectionFlowLimit`

Emplacement des fichiers publics .jar

Sur la plate-forme Solaris, les fichiers publics `.jar` ont été déplacés de l'ancien emplacement dans Message Queue 3.0 (`/usr/share/lib/imq/`) vers l'emplacement suivant dans Message Queue 3.0.1 : `/usr/share/lib/`. Les liens symboliques introduits dans Message Queue 3.0.1 à l'emplacement `/usr/share/lib/imq/` des fichiers jar déplacés ont été supprimés.

Cela concerne les fichiers `.jar` suivants :

- `jms.jar`
- `imq.jar`
- `imqxm.jar`
- `activation.jar`
- `saaj-api.jar`
- `saaj-impl.jar`
- `mail.jar`
- `commons-logging.jar`
- `jaxm-api.jar`
- `fscontext.jar`

Mises à jour de la documentation

Cette section présente les changements apportés à la documentation Message Queue des versions 3.5 SP1et 3.5.

Changements apportés à la version 3.5 SP1

Les documents Message Queue 3.5 SP1 suivants ont été mis à jour par rapport à la version 3.5 du produit :

Guide d'installation

Le *Guide d'installation de Message Queue* a été mis à jour afin de refléter les changements de noms commerciaux et d'informations de prise en charge de la plate-forme.

Guide d'administration

Le nom du *Guide d'administration de Message Queue* a changé (l'ancien nom était *Guide de l'administrateur de Message Queue*) et le document a été changé pour refléter les changements de noms commerciaux. Ce document contient également un chapitre plus complet et actualisé décrivant le contrôle des performances.

Guide du développeur Java Client

Le *Guide du développeur Message Queue Client Java* a été mis à jour pour refléter les changements de noms commerciaux.

Guide du développeur Client C

Le *Guide du développeur Message Queue Client C* a été mis à jour pour refléter les changements de noms commerciaux.

Changements apportés à la version 3.5

Les documents Message Queue 3.5 suivants ont été mis à jour par rapport à la version 3.0.1 du produit : Ces mises à jour se trouvent sur le site Web de la documentation Message Queue 3.5 : `http://docs.sun.com/coll/S1_MessageQueue_35`.

Guide d'installation

Le produit Message Queue 3.5 contient un *Guide d'installation de Message Queue* actualisé. Celui-ci présente les nouvelles exigences en matière de logiciel, les changements apportés aux instructions relatives à l'installation de Solaris, une nouvelle procédure d'installation de Linux à l'aide de Red Hat Package Manager (RPM) et de la structure de répertoire installés, ainsi que des changements mineurs dans l'installation de Windows.

Correction: En ce qui concerne la prise en charge de la persistance intégrée, le Tableau 1-2 précise que la version 4.5 de PointBase est prise en charge par Message Queue. Il s'agit en fait de PointBase, Version 4.8.

Guide de l'administrateur

Le document *Guide d'administration de Message Queue* a été mis à jour pour refléter les changements apportés à Message Queue 3.5 (consultez la rubrique « Cette section présente les modifications présentes dans Message Queue 3.5 SP1 ainsi que dans la version précédente, Message Queue 3.5. » à la page 2).

Guide du développeur Java Client

Le *Guide du développeur Message Queue Client Java* intègre une grande partie des informations contenues dans Message Queue *Guide du développeur*. Il a été mis à jour pour refléter les changements apportés à Message Queue 3.5 (consultez la rubrique « Cette section présente les modifications présentes dans Message Queue 3.5 SP1 ainsi que dans la version précédente, Message Queue 3.5. » à la page 2).

Guide du développeur Client C

Le *Guide du développeur Message Queue Client C* est un nouveau manuel qui a été ajouté à la documentation Message Queue, en vue de présenter la création d'applications Client C Message Queue.

Problèmes et limites connus

Cette section présente les problèmes, limites et bogues connus de Message Queue 3.5 SP1 et Message Queue 3.5. La version 3.5 SP1 étant simplement une version renommée de Message Queue 3.5, les informations présentées dans cette section concernent les deux versions.

Pour connaître la liste des bogues en cours, leur état et les solutions possibles, les membres de Java Developer Connection™ peuvent consulter la page Bug Parade du site Web Java Developer Connection. Avant de signaler tout nouveau bogue, merci de consulter cette page. Bien que les bogues de Message Queue n'y soient pas tous répertoriés, il est bon de s'y référer pour savoir si un problème a déjà été signalé.

La page en question est la suivante :

http://developer.java.sun.com/developer/bugParade

REMARQUE L'adhésion à Java Developer Connection est gratuite, mais vous devez vous inscrire. 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 courrier électronique à l'adresse imq-feedback@sun.com.

Problèmes connus

Cette section présente les problèmes connus de Message Queue 3.5 SP1. Certains sont apparus avec la version Message Queue 3.5. Cette section regroupe les problèmes en fonction de leur pertinence pour Enterprise Edition et Platform Edition de Message Queue 3.5 ou Enterprise Edition uniquement.

Enterprise Edition et Platform Edition

- À la suite du changement de nom commercial, les API qui auparavant renvoyaient la chaîne suivante :

  ```
  "Sun ONE Message Queue, Sun Microsystems, Inc."
  ```

  renverront désormais :

  ```
  "Sun Java(tm) System Message Queue"
  ```

- À la suite du changement de nom commercial, les programmes Client C compilés à l'aide de MQ 3.5 FCS qui font une comparaison exacte de la valeur associée à la propriété `MQ_NAME_PROPERTY` à partir de `MQGetMetaData()` échoueront s'ils utilisent la bibliothèque partagée `mqcrt` de la version 3.5 SP1 lors de l'exécution.
- Les plate-formes Windows limitent le nombre de connexions à un courtier pouvant être initiées simultanément par TCP/IP, conformément à la taille maximale du backlog. Le backlog correspond au tampon des connexions de la pile TCP : le nombre de connexions TCP simultanées ne peut pas dépasser la taille du backlog. Par exemple, le backlog est limité à 5 sous Windows 2000 Professionnel et à 200 sous Windows 2000 Server.

- Si vous utilisez Windows XP, le nombre de connexions *entrantes* est limité. En ce qui concerne Windows XP Professionnel, lenombre maximal d'autres ordinateurs autorisés à se connecter simultanément sur le réseau est 10. Cette limite inclut tous les protocoles de transport et de partage de ressources combinés. En ce qui concerne Windows XP Édition Familiale, le nombre maximal d'autres ordinateurs autorisés à se connecter simultanément sur le réseau est 5. Cette limite a une incidence sur le nombre de clients pouvant se connecter au courtier fonctionnant sous Windows XP.

  Les fichiers, impressions, tubes nommés ou session de messagerie sans activité sont automatiquement déconnectés après expiration du délai `AutoDisconnect` ; la valeur par défaut du délai de la propriété `AutoDisconnect` est de 15 minutes. À la déconnexion de la session, l'une des dix connexions devient disponible pour qu'un autre utilisateur puisse se connecter au système Windows XP. Par conséquente, la réduction du délai de la propriété `AutoDisconnect` peut permettre de résoudre certains des problèmes liés à la limite de dix connexions ou de cinq connexions sur un système qui n'est pas utilisé principalement en tant que serveur. Pour plus d'informations, reportez-vous aux pages suivantes :

  `http://support.microsoft.com/default.aspx?scid=kb;EN-US;314882`

- Vous ne pouvez pas modifier le fichier de configuration d'une instance de courtier sans avoir démarré, au moins une fois, cette instance. La raison en est que le fichier `config.properties` n'existe pas tant que l'instance du courtier n'a pas été démarrée une première fois. Pour configurer un courtier afin qu'il utilise la permanence enfichable ou pour définir d'autres propriétés de configuration, exécutez une fois le courtier (avec le nom d'instance qui doit être utilisé pour créer le courtier) pour créer le fichier `config.properties` :

Plate-forme	Emplacement
Solaris	`/var/imq/instances/`*instanceName*`/props/config.properties`
Linux	`/var/opt/imq/instances/`*NomInstance*`/props/config.properties`
Windows	`IMQ_VARHOME\instances\`*NomInstance*`\props\config.properties`

  Lorsque le fichier `config.properties` a été créé, modifiez-le pour ajouter d'autres valeurs de propriétés de configuration, puis redémarrez le courtier.

Enterprise Edition uniquement

- Seuls les clusters de courtiers 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 connectez des courtiers par l'intermédiaire de l'argument de ligne de commande `imqbrokerd -cluster`, assurez-vous bien que tous les courtiers du cluster sont pris en compte.
- Si aucun courtier ne fait office de courtier principal dans un cluster, les informations permanentes enregistrées par un courtier ayant été ajouté au cluster ne sont pas communiquées aux autres courtiers du cluster.
- Un service de connexion utilisant SSL ne prend en charge actuellement que les certificats de serveurs auto-signés, c'est-à-dire le mode d'approbation par l'hôte.
- 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 pendant cet intervalle d'une minute et si elle tente d'utiliser le même ID client, le même abonnement durable ou la même file d'attente, elle peut recevoir une exception "L'ID client est déjà utilisé". Il ne s'agit pas véritablement d'un 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.

Bogues connus

Le Tableau 10 répertorie les bogues subsistant dans Message Queue 3.5 SP1.

Tableau 10 Bogues connus dans Message Queue 3.5

Référence	Détails
4683029	L'option `-javahome` dans tous les scripts Solaris/Windows ne fonctionne pas si la valeur comprend un espace. L'option `-javahome` est utilisée par les commandes et les utilitaires de Message Queue pour indiquer un autre exécuteur Java 2 compatible que vous pouvez utiliser. Néanmoins, le chemin d'accès de l'autre exécuteur Java ne doit pas contenir d'espace. Vous trouverez ici des exemples de chemin ayant des espaces : Windows : `C:\jdk 1.4` (sous Windows les espaces sont autorisés dans le chemin si celui-ci est compris entre des guillemets, comme dans l'exemple `"C:\jdk 1.4"`) Solaris : `/work/java 1.4` **Solution** : installez l'exécuteur Java à un emplacement ou dans un chemin ne contenant pas d'espace.
4939923	Le courtier peut provoquer des exceptions `NullPointerException` lors de l'utilisation d'un pool de threads partagés, si la JVM du courtier dispose de peu de mémoire. **Solution** : aucune. Ce bogue est résolu dans J2SE 1.4.2_03.
4941058	Les destinations dont le contrôle de flux est activé peuvent ne pas atteindre la limite maximale. Dans certains cas, les producteurs ne peuvent pas envoyer des messages à une destination avant que celle-ci n'atteigne sa limite maximale configurée. **Solution** : aucune.
4941066	Les destinations peuvent dépasser légèrement les limites attribuées. **Solution** : aucune.
4941127	Une destination ne se charge pas complètement si un message dépasse la limite de taille des messages individuels. Si la limite de la taille des messages autorisée dans une destination est modifiée après le stockage d'un message plus volumineux, la destination ne se charge pas correctement. **Solution** : augmentez la limite de taille des messages jusqu'à ce que le message volumineux soit consommé, puis abaissez-la. Il est possible d'interrompre la production vers la destination pendant ce laps de temps, afin d'empêcher l'acceptation d'autres messages volumineux.
4946531	Des exceptions `NullPointerException` inoffensives peuvent survenir lors de la production de messages, bien que cela soit rare. **Solution** : aucune ; vous pouvez passer outre cette exception sans crainte.

Tableau 10 Bogues connus dans Message Queue 3.5 *(suite)*

Référence	Détails
4949398	Le paramètre `imqcmd query dst` transmet des valeurs incorrectes pour les propriétés `Nombre de messages` et `Taille totale des messages` lorsque la destination correspondante est en cours de chargement. Les valeurs transmises sont correctes avant et après le chargement de la destination. **Solution** : le problème survient uniquement lors du chargement de la destination. Après le chargement, les valeurs renvoyées sont correctes.
4950166	Erreurs alétaoires du courtier lors de son exécution sous les systèmes jdk1.4.2_02 et x86. Pour plus d'informations, reportez-vous au bogue J2SE 4947404. **Solution** : démarrez le courtier à l'aide de `-XX:UseSSE=0`, par exemple. `imqbrokerd -tty -vmargs -XX:UseSSE=0`

4950601	Le paramètre `imqcmd metrics dst` déclenche une erreur interne du courtier qui s'imprime lors de l'utilisation du magasin JDBC persistant.
	
	Les paramètres liés à l'utilisation du disque concernent uniquement le magasin de fichiers. Cependant, lors de l'extraction de ces paramètres, le courtier tente d'obtenir les informations liées à l'utilisation du disque quel que soit le type de magasin. Si vous utilisez une base de données au lieu d'un magasin de fichiers, le courtier imprime le message d'erreur suivant :
	
	`06/Nov/2003:22:57:36 PST] ERROR [B3100]: Erreur interne du courtier inattendue`
	
	`Erreur : [unable to disk usage for destinationT:topic1] :`
	
	`com.sun.messaging.jmq.jmsserver.util.BrokerException:`
	`Cette opération ne s'applique pas au magasin persistant intégré.`
	
	Solution : aucune.

4951010	Dans un cluster de courtiers, un courtier place les messages dans une file d'attente vers une connexion à distance qui n'est peut-être pas démarrée. **Solution** : le consommateur reçoit les messages une fois la connexion démarrée. Les messages seront livrés à un autre consommateur si la connexion du consommateur est désactivée.
4953348	Le paramètre HTTPS `createQueueConnection` déclenche occasionnellement une exception sous Windows 2000. **Solution** : relancez la connexion.
4953354	Le courtier devient inaccessible lorsque le magasin persistant ouvre trop de destinations. **Solution** : ce problème vient du fait que le courtier a atteint la limite des descripteurs de fichiers ouverts. Sous Solaris et Linux, faites appel à la commande `ulimit` pour augmenter le nombre limite de descripteurs de fichiers.
4954974	L'installation par CD ne démarre pas automatiquement sous Windows XP. **Solution** : dans l'Explorateur Windows, double-cliquez sur le dossier Windows du CD, puis double-cliquez sur le fichier `imq3_5-ent-win.exe` pour lancer l'installation.

Tableau 10 Bogues connus dans Message Queue 3.5 *(suite)*

Référence	Détails
4983525	La création d'un producteur de messages pour une destination créée automatiquement peut échouer sous Linux Red Hat Advanced Server 3.0. **Solution** : tentez de créer à nouveau le producteur. Vous devriez y parvenir la seconde fois. Vous pouvez également avoir recours à une destination créée administrativement.

4986318	Le client peut générer un message ACKNOWLEDGE_REPLY de manière inattendue :
	
	`******** Packet: ACKNOWLEDGE_REPLY(25):26-192.18.86.227-42976-1075458056557`
	
	`Magic/Version: 469754818/301Size: 97 Type: ACKNOWLEDGE_REPLY(25)`
	
	`Expiration: 0 Timestamp: 1075458056557`
	
	`Source IP: 192.18.86.227 Source Port: 42976Sequence: 26`
	
	Solution : aucune. Il existe une condition de synchronisation rare dans le courtier qui fait en sorte que le client génère cette erreur. Vous pouvez passer outre cette erreur. Aucun message ne sera perdu.

4991257	L'envoi de messages permanents volumineux à des abonnés durables dans un cluster de courtiers dans lequel le magasin persistant est JDBC peut entraîner un blocage du courtier et/ou générer des erreurs. **Solution** : augmentez le délai d'expiration du protocole de verrouillage du courtier, à l'aide de la propriété suivante du courtier : `imq.cluster.timeout=<expiration-en-secondes>` La valeur par défaut est 60. Si le stockage des messages volumineux s'avère lent, il se peut que vous deviez régler la base de données du magasin persistant ou basculer vers un magasin persistant différent.

5006686	L'exemple ARGS contenu dans le paramètre imqbrokerd.conf est incorrect.
	
	Solution : les valeurs ne doivent pas figurer entre guillemets.
	
	`ARGS="-name newbroker -port 8888"`
	
	La valeur doit apparaître comme suit :
	
	`ARGS=-name newbroker -port 8888`

Fichiers redistribuables

Sun Java System Message Queue 3.5 SP1 contient l'ensemble de fichiers suivants, que vous pouvez utiliser et distribuer librement sous forme binaire :

```
jms.jar
imq.jar
imqxm.jar
fscontext.jar
providerutil.jar
jndi.jar
ldap.jar
ldapbpjar
jaas.jar
jsse.jar
jnet.jar
jcert.jar
```

En outre, vous pouvez également redistribuer les fichiers `LICENSE` et `COPYRIGHT`.

Communication de problèmes et formulation de commentaires

Pour signaler un problème, envoyez un courrier électronique à l'adresse `imq-feedback@sun.com`.

Si vous bénéficiez d'un contrat d'assistance Message Queue, contactez l'assistance client d'une des manières suivantes :

- Services d'assistance logicielle Sun en ligne à l'adresse `http://www.sun.com/service/sunone/software`

  Ce site contient des liens vers la base de connaissances, le centre d'assistance en ligne et ProductTracker, ainsi que vers des programmes de maintenance et les coordonnées des services d'assistance.

- Le numéro de téléphone associé à votre contrat de maintenance.

Afin de nous aider à résoudre vos problèmes, pensez à réunir les informations suivantes lorsque vous contactez l'assistance technique :

- Description du problème, y compris l'endroit où il se produit et son impact sur l'exploitation.
- Type de machine, versions du système d'exploitation et du produit, y compris les correctifs et autres logiciels pouvant avoir un lien avec le problème.
- Procédure détaillée des méthodes utilisées pour reproduire le problème.
- Tous les journaux d'erreur ou vidages de la mémoire.

Sun attend vos commentaires

Afin d'améliorer sa documentation, Sun vous encourage à faire des commentaires et à apporter des suggestions. Pour ce faire, utilisez le formulaire électronique disponible à l'adresse suivante :

`http://www.sun.com/hwdocs/feedback/`

Veuillez indiquer le titre complet du document ainsi que son numéro dans les champs appropriés. Le numéro du document se trouve sur la page de titre du manuel ou en haut du document. Il s'agit généralement d'un nombre à 7 ou 9 chiffres. Par exemple, le numéro de référence des présentes *Message Queue 3.5 SP1 Notes de mise à jour* est 817-7200-10.

Ressources Sun supplémentaires

Outre la documentation relative à Message Queue, vous disposez des ressources d'information suivantes :

Forums de discussion

Sun Java System Forum concernant le logiciel

Il existe un forum Sun Java System Message Queue à l'adresse suivante :

http://softwareforum.sun.com/NASApp/jive/forum.jsp?forum=24

Votre participation y est 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

Base de connaissances SunSolve

Des informations relatives à Sun Java System Message Queue sont disponibles dans la base de connaissances SunSolve Knowledge, à l'adresse suivante :

http://sunsolve.Sun.COM/pub-cgi/search.pl?mode=advanced

Cochez la case « All Free Collections », puis effectuez une recherche sur « Message Queue ».

Informations relatives à Sun Java System

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

- Page relative aux produits Message Queue
  http://wwws.sun.com/software/products/message_queue/index.html
- Documentation relative à Message Queue
  http://docs.sun.com/coll/MessageQueue_35_SP1
- Documentation relative à Sun
  http://docs.sun.com/
- Produits et services Sun Java System Software
  http://www.sun.com/software
- Services d'assistance logicielle de Sun
  http://www.sun.com/service/sunone/software
- Base de connaissance et d'assistance Sun
  http://sunsolve.sun.com
- Services de formation et d'assistance Sun
  http://www.sun.com/supportraining
- Informations destinées aux développeurs Sun
  http://developers.sun.com/
- Services d'assistance pour développeurs Sun
  http://www.sun.com/developers/support
- Fiches techniques sur les logiciels Sun
  http://www.sun.com/software