Versionshinweise zu Sun Java System Message Queue 4.1

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

Teilenr.: 820–3188–10
September 2007

071001@18741

Inhalt

◆ ◆ ◆ KAPITEL 1

Versionshinweise zu Sun Java System Message Queue 4.1

Version 4.1

Teilenr. 820-3188-10

Diese Versionshinweise enthalten wichtige Informationen, die zum Zeitpunkt der Herausgabe von Sun Java™ System Message Queue 4.1 zur Verfügung standen. In diesem Dokument werden neue Funktionen und Verbesserungen, bekannte Probleme und Einschränkungen beschrieben und weitere Informationen bereitgestellt. Lesen Sie dieses Dokument sorgfältig, bevor Sie Message Queue verwenden. Diese Versionshinweise enthalten zudem Informationen zu Version 4.0 von Message Queue; Informationen zu den in dieser Version eingeführten Funktionen finden Sie unter „Informationen zu Message Queue 4.0” auf Seite 18.

Die neueste Ausgabe dieser Versionshinweise werden auf der Dokumentations-Website von Sun Java System Message Queue bereitgestellt. Besuchen Sie diese Website vor der Installation und Konfiguration Ihrer Software und später regelmäßig, um stets die neuesten Versionshinweise und Produktdokumentationen verfügbar zu haben.

In diesen Versionshinweisen werden die folgenden Themen behandelt:

In der vorliegenden Dokumentation wird auf URLs von Drittanbietern verwiesen, über die zusätzliche relevante Informationen zur Verfügung gestellt werden.

Sun ist nicht verantwortlich für die Verfügbarkeit der in diesem Dokument erwähnten Websites anderer Hersteller. Sun ist nicht verantwortlich oder haftbar für die Inhalte, Werbung, Produkte oder andere Materialien, die auf solchen Websites/Ressourcen oder über diese verfügbar sind, und unterstützt diese nicht. Sun lehnt jede Verantwortung oder Haftung für direkte oder indirekte Schäden oder Verluste ab, die durch die bzw. in Verbindung mit der Verwendung von oder der Stützung auf derartige Inhalte, Waren oder Dienstleistungen, die auf oder über diese Sites oder Ressourcen verfügbar sind, entstehen können.

Änderungsprotokoll der Versionshinweise

In der folgenden Tabelle sind die Datumsangaben zu den 4.x-Versionen von Message Queue sowie die wesentlichen Änderungen der einzelnen Versionen aufgeführt.

TABELLE 1–1 Änderungsprotokoll

Datum	Beschreibung der Änderungen
Mai 2006	Ursprüngliche Version dieses Dokuments für Version 4.0 von Message Queue.
Januar 2007	Ursprüngliche Version dieses Dokuments für Beta-Version 4.1 von Message Queue. Fügt eine Beschreibung der JAAS-Unterstützung hinzu.
April 2007	Zweite Version dieses Dokuments für Beta-Version 4.1 von Message Queue. Fügt eine Hochverfügbarkeitsfunktion hinzu.
September 2007	Dritte Version dieses Dokuments, die dem Kunden bereitgestellt wird. Fügt eine Beschreibung für die Unterstützung des Java Enterprise System Monitoring Frameworks, für feste C-Ports, behobene Probleme und andere Funktionen hinzu.

Informationen zu Message Queue 4.1

Sun Java System Message Queue ist ein leistungsfähiger Nachrichtendienst, der ein zuverlässiges, asynchrones Messaging gemäß JMS 1.1 (Java Messaging Specification) bietet. Zusätzlich stellt Message Queue Funktionen bereit, die über die JMS-Spezifikation hinausgehen, um den Anforderungen großer Bereitstellungen im Unternehmensbereich gerecht zu werden.

Version 4.1 von Message Queue fügt Unterstützung für Hochverfügbarkeit, für Java Authentication and Authorization Service (JAAS), für die Verwendung fester C-Ports sowie für das Java Enterprise System Monitoring Framework hinzu. Darüber hinaus wurden kleinere Erweiterungen hinzugefügt und weitere Fehler behoben. Dieser Abschnitt umfasst die folgenden Informationen.

Informationen zu den in Message Queue 4.0 hinzugefügten Funktionen finden Sie unter „Informationen zu Message Queue 4.0” auf Seite 18.

Neuheiten in Version 4.1

Mit Message Queue 4.1 werden hochverfügbare (Daten- und Dienstverfügbarkeit) Broker-Cluster, JAAS-Unterstützung und verschiedene andere kleinere Funktionen eingeführt. In diesem Abschnitt sind diese Funktionen beschrieben, und Sie erhalten weitere Referenzen.

Hochverfügbarkeit

Mit Message Queue 4.1 werden hochverfügbare Cluster eingeführt, die Daten- und Dienstverfügbarkeit bieten. Wenn die Verbindung eines Clients zu einem Hochverfügbarkeits-Broker getrennt wird, wird der Client automatisch mit einem anderen Broker in einem Cluster verbunden. Der Broker, der die neue Verbindung bereitstellt, übernimmt die persistenten Daten und den Status des fehlgeschlagenen Brokers und stellt weiterhin einen unterbrechungsfreien Dienst für die Clients des fehlgeschlagenen Brokers bereit. Hochverfügbarkeits-Broker können über eine sichere Verbindung ausgeführt werden.

Hochverfügbarkeits-Broker erfordern die Verwendung einer hochverfügbaren Datenbank (Highly Available Database, HADB). Wenn Sie nicht über eine solche Datenbank verfügen, oder wenn die Datenverfügbarkeit für Ihre Organisation nicht wichtig ist, können Sie weiterhin konventionelle Cluster verwenden, die eine automatische erneute Verbindungsherstellung und Datenverfügbarkeit bieten.

Die Konfiguration von Hochverfügbarkeits-Brokern ist unkompliziert: Sie geben für jeden Broker innerhalb des Clusters die folgenden Broker-Eigenschaften an.

- *Eigenschaften für die Clustermitgliedschaft* geben an, dass der Broker Teil eines Hochverfügbarkeits-Clusters ist, und legen die ID des Clusters und des Brokers fest.
- *Eigenschaften für die hochverfügbare Datenbank (HADB)* geben das Modell für persistente Meldungen (JDBC), den Namen des HADB-Anbieters sowie herstellerspezifische Konfigurationseigenschaften für die Datenbank an.
- *Eigenschaften für Fehlererkennung und Übernahme* legen fest, wie ein Broker-Ausfall ermittelt und behandelt werden soll.

Um diese Funktion zu verwenden, müssen Sie folgende Schritte ausführen:

1. Installieren einer hochverfügbaren Datenbank.
2. Installieren der .jar-Datei des JDBC-Treibers.
3. Erstellen des Datenbankschemas für den hochverfügbaren persistenten Speicher.
4. Festlegen der Hochverfügbarkeitseigenschaften für jeden Broker im Cluster.
5. Starten der einzelnen Broker im Cluster.

Eine Beschreibung der Konzepte im Zusammenhang mit der Hochverfügbarkeit und einen Vergleich mit konventionellen Clustern finden Sie in Kapitel 4, „Broker Clusters" in *Sun Java System Message Queue 4.1 Technical Overview*. Die Vorgehensweise und Referenzinformationen zu Hochverfügbarkeitslösungen finden Sie in Kapitel 8, „Broker Clusters" in *Sun Java System Message Queue 4.1 Administration Guide* und unter „Cluster Configuration Properties" in *Sun Java System Message Queue 4.1 Administration Guide*.

Wenn Sie eine hochverfügbare Datenbank mit Message Queue 4.0 verwendet haben und nun einen Hochverfügbarkeits-Cluster verwenden möchten, können Sie das Dienstprogramm `dbmgr` einsetzen, um ein Upgrade auf einen gemeinsamen hochverfügbaren Datenbankspeicher durchzuführen. Weitere Informationen finden Sie unter „Broker-Cluster" auf Seite 48.

JAAS-Unterstützung

Neben den dateibasierten und den LDAP-basierten integrierten Authentifizierungsmechanismen unterstützt Message Queue auch Java Authentication and Authorization Service (JAAS), sodass Sie verschiedene Dienste zum Authentifizieren von Message Queue-Clients mit dem Broker verbinden können. In diesem Abschnitt werden die Informationen beschrieben, die der Broker für einen JAAS-konformen Authentifizierungsdienst bereitstellt. Ferner wird die Konfiguration des Brokers für die Verwendung eines solchen Dienstes beschrieben.

Die JAAS-API kann im Rahmen dieses Dokuments nicht beschrieben werden. Wenn Sie mehr zu diesem Thema wissen möchten, finden Sie in den folgenden Quellen weitere Informationen.

- Umfassende Informationen zur JAAS-API finden Sie im *Java Authentication and Authorization Service (JAAS) Reference Guide*.

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

- Informationen zum Schreiben eines Anmeldemoduls finden Sie im *Java Authentication and Authorization Service (JAAS) LoginModule Developer's Guide*.

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

Die JAAS-API ist eine wichtige API in J2SE und daher ein wesentlicher Teil der Message Queue-Laufzeitumgebung. JAAS definiert eine Abstraktionsschicht zwischen einer Anwendung und einem Authentifizierungsmechanismus, der es ermöglicht, den gewünschten Mechanismus einzubinden, ohne den Anwendungscode zu ändern. Beim Message Queue-Dienst befindet sich die Abstraktionsschicht zwischen dem Broker (Anwendung) und einem Authentifizierungsanbieter. Durch die Festlegung einiger Broker-Eigenschaften kann

ein beliebiger JAAS-konformer Authentifizierungsdienst eingebunden und ohne Unterbrechung oder Änderung des Broker-Codes aktualisiert werden.

Sie können JMX-Clients zum Verwalten des Brokers verwenden, wenn Sie die JAAS-basierte Authentifizierung verwenden; vor dem Start des Brokers muss die JAAS-Unterstützung jedoch manuell eingerichtet werden (durch die Festlegung von JAAS-bezogenen Broker-Eigenschaften). Diese Eigenschaften können jedoch nicht über die JMX API geändert werden.

Elemente von JAAS

Abbildung 1–1 zeigt die grundlegenden Elemente von JAAS: ein JAAS-Client, ein JAAS-konformer Authentifizierungsdienst und eine JAAS-Konfigurationsdatei.

- Der JAAS-Client ist eine Anwendung, die eine Authentifizierung unter Verwendung eines JAAS-konformen Authentifizierungsdienstes durchführen möchte. Sie kommuniziert mit diesem Dienst über ein Anmeldemodul (LoginModule) und ist für die Bereitstellung eines Callback-Handlers zuständig, den das Anmeldemodul zum Abrufen von Benutzername, Passwort und anderen relevanten Informationen aufrufen kann.
- Der JAAS-konforme Authentifizierungsdienst umfasst mindestens ein Anmeldemodul sowie Logik zum Durchführen der erforderlichen Authentifizierung. Das Anmeldemodul kann die Authentifizierungslogik umfassen bzw. ein privates Protokoll oder eine API für die Kommunikation mit einem Modul verwenden, welche diese Logik bereitstellt.
- Die JAAS-Konfigurationsdatei ist eine Textdatei, die der JAAS-Client zum Ermitteln der Anmeldemodule verwendet, die für die Kommunikation mit dem JAAS-konformen Dienst benötigt werden.

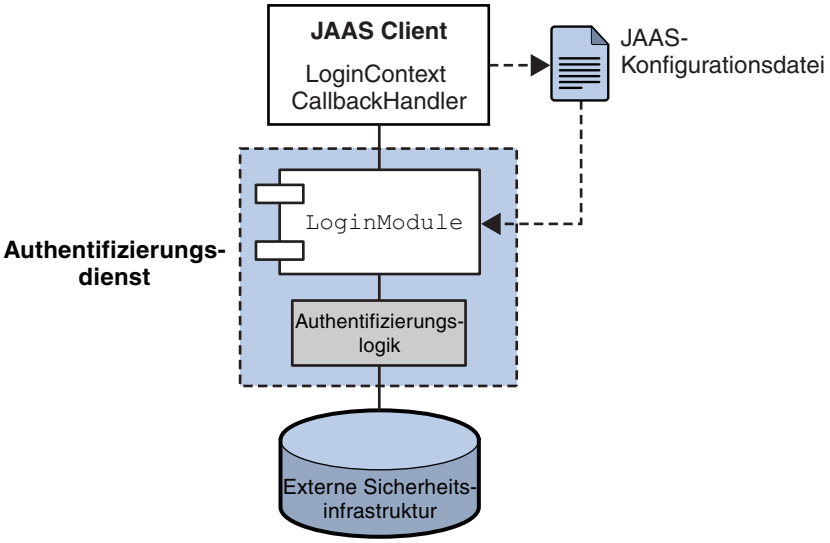

ABBILDUNG 1–1 JAAS-Elemente

Im nächsten Abschnitt wird beschrieben, wie der Message Queue-Dienst diese Elemente für eine JAAS-konforme Authentifizierung verwendet.

JAAS und Message Queue

Die nächste Abbildung zeigt, wie JAAS vom Message Queue-Broker verwendet wird. Sie zeigt eine komplexere Implementierung des JAAS-Modells aus der vorherigen Abbildung.

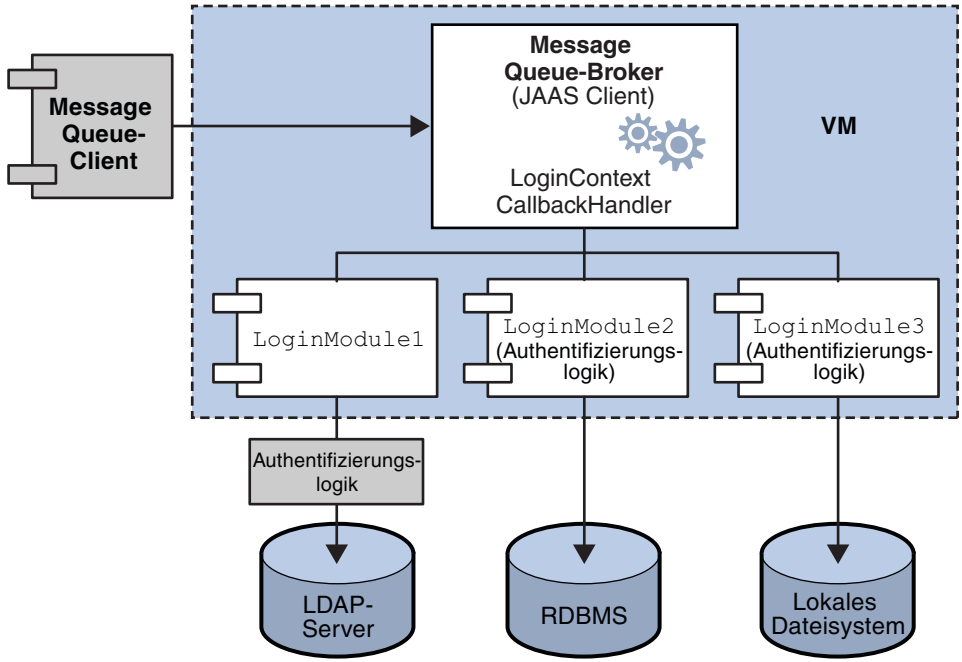

ABBILDUNG 1–2 Verwendung von JAAS durch Message Queue

Wie im einfacheren Beispiel gezeigt, befinden sich Authentifizierungsdienst und Broker nicht in derselben Schicht. Der Authentifizierungsdienst umfasst mindestens ein Anmeldemodul (LoginModule) und bei Bedarf zusätzliche Authentifizierungsmodule. Die Anmeldemodule werden in derselben virtuellen Java-Maschine ausgeführt wie der Broker. Der Message Queue-Broker wird dem Anmeldemodul als `LogInContext` angezeigt und kommuniziert mit dem Anmeldemodul über einen `CallBackHandler`, der Teil des Broker-Laufzeitcodes ist.

Der Authentifizierungsdienst stellt ferner eine JAAS-Konfigurationsdatei bereit, welche Einträge für die Anmeldemodule enthält. Die Konfigurationsdatei gibt die Reihenfolge an, in der die Module verwendet werden sollen, sowie einige Bedingungen für deren Verwendung. Beim Start des Brokers ermittelt JAAS die Konfigurationsdatei über die Java-Systemeigenschaft `java.security.auth.login.config` oder die Java-Sicherheitseigenschaftendatei. Anschließend wird basierend auf dem Wert der Broker-Eigenschaft `imq.user_repository.jaas.name` ein Eintrag in der JAAS-Konfigurationsdatei ausgewählt. Der Eintrag gibt an, welche Anmeldemodule für die Authentifizierung verwendet werden. Wie die Abbildung zeigt, kann der Broker mehrere Anmeldemodule verwenden. (Die Beziehung zwischen der Konfigurationsdatei, dem Anmeldemodul und dem Broker wird in Abbildung 1–3 veranschaulicht.)

Die Tatsache, dass der Broker einen JAAS-Plug-In-Authentifizierungsdienst verwendet, ist für den Message Queue-Client vollständig transparent. Der Client bleibt wie zuvor mit dem Broker verbunden und übergibt einen Benutzernamen und ein Passwort. Der Broker wiederum

verwendet einen Callback-Handler, um diese Informationen an den Authentifizierungsdienst zu übergeben; und der Dienst verwendet diese Informationen, um den Benutzer zu authentifizieren und die Ergebnisse zurückzugeben. Bei erfolgreicher Authentifzierung lässt der Broker die Verbindung zu; wenn die Authentifizierung nicht erfolgreich ist, gibt die Clientlaufzeit eine JMS-Sicherheitsausnahme zurück, die der Client verarbeiten muss.

Nach der Authentifizierung des Message Queue-Clients wird die normale Ausführung des Brokers fortgesetzt (sofern keine weiteren Authentifizierungsvorgänge erforderlich sind); er ermittelt anhand der Zugriffssteuerungsdatei, ob der authentifizierte Client für die ausgeführten Aktionen berechtigt ist: Zugreifen auf ein Ziel, Verwenden einer Nachricht, Durchsuchen einer Warteschlange usw.

Einrichten der JAAS-konformen Authentifizierung

Das Einrichten der JAAS-konformen Authentifizierung umfasst das Festlegen von Broker- und Systemeigenschaften zur Auswahl dieses Authentifizierungstyps, zum Angeben des Speicherorts der Konfigurationsdatei sowie zum Angeben der Einträge für die Anmeldemodule, die verwendet werden sollen.

In diesem Abschnitt wird die Beziehung zwischen dem JAAS-Client, den Anmeldemodulen und der JAAS-Konfigurationsdatei erläutert. Ferner werden die erforderlichen Schritte zum Einrichten der JAAS-konformen Authentifizierung beschrieben. Die folgende Abbildung zeigt die Beziehung zwischen der Konfigurationsdatei, dem Anmeldemodul und dem Broker.

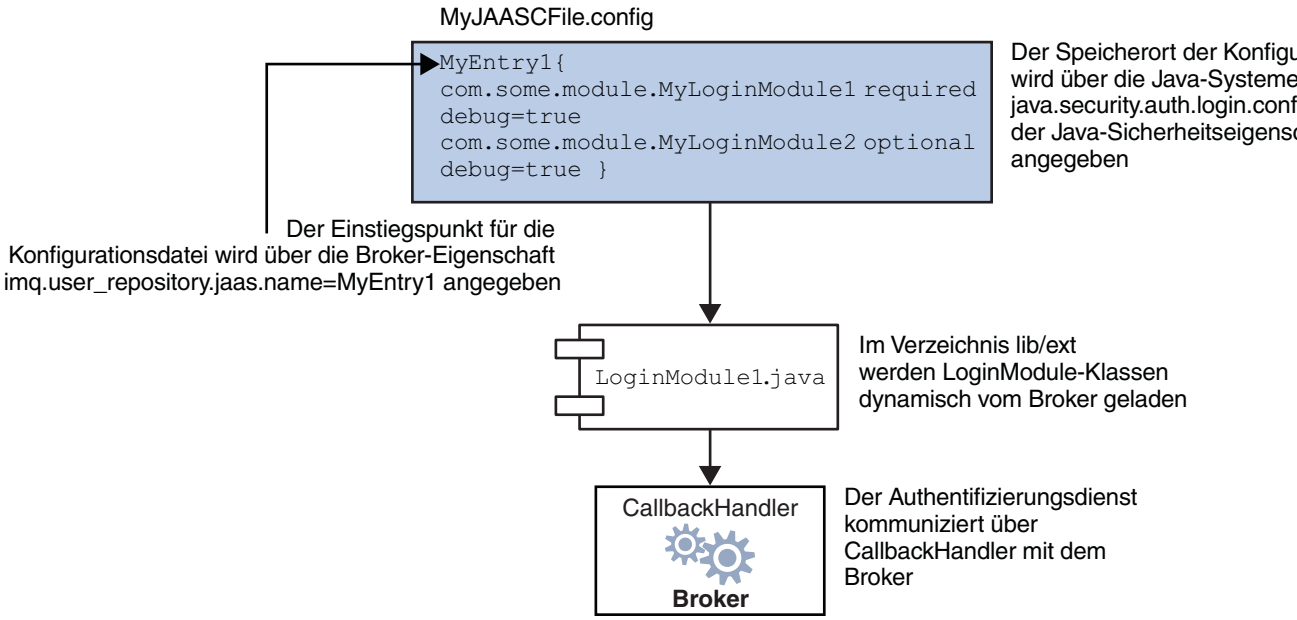

ABBILDUNG 1–3 Einrichten der JAAS-Unterstützung

Wie in der Abbildung gezeigt, enthält die JAAS-Konfigurationsdatei `MyJAASCFile.config` Verweise auf mehrere Anmeldemodule, die unter einem Einstiegspunkt gruppiert sind. Der Broker ermittelt die Konfigurationsdatei anhand der Java-Systemeigenschaft `java.security.auth.login.config` oder der Java-Sicherheitseigenschaftendatei. Die zu verwendenden Anmeldemodule werden anhand der Broker-Eigenschaft `imq.user_repository.jaas.name` ermittelt, welche den gewünschten Eintrag in der Konfigurationsdatei angibt. Die Klassen für diese Module befinden sich im Verzeichnis `lib/ext`.

Zum Einrichten der JAAS-Unterstützung für Message Queue müssen die folgenden Schritte ausgeführt werden. (In einer Entwicklungsumgebung kann der Entwickler all diese Schritte ausführen. In einer Produktionsumgebung würde der Administrator einige dieser Schritte übernehmen.)

1. Erstellen Sie mindestens eine Anmeldemodulklasse, welche den Authentifizierungsdienst implementiert. Im Folgenden sind die JAAS-Callback-Typen aufgelistet, die der Broker unterstützt.

   `javax.security.auth.callback.LanguageCallback`
   Der Broker verwendet diesen Callback-Typ, um das Gebietsschema an den Authentifizierungsdienst zu übergeben, in dem der Broker ausgeführt wird. Dieser Wert kann für die Lokalisierung verwendet werden.

`javax.security.auth.callback.NameCallback`
Der Broker verwendet diesen Callback-Typ, um den Benutzernamen an den Authentifizierungsdienst zu übergeben, den der Message Queue-Client beim Anfordern der Verbindung angegeben hat.

`javax.security.auth.callback.TextInputCallback`
Der Broker verwendet diesen Callback-Typ, um den Wert von `imq.authentication.type` für den Authentifizierungsdienst anzugeben, wenn `TextInputCallback.getPrompt()` `imq.authentication.type` lautet. Gegenwärtig ist für dieses Feld ausschließlich der Wert `basic` zulässig. Mit diesem Wert wird eine Base-64-Passwortverschlüsselung festgelegt.

`javax.security.auth.callback.PasswordCallback`
Der Broker verwendet diesen Callback-Typ, um das Passwort, das der Message Queue-Client beim Anfordern der Verbindung angegeben hat, an den Authentifizierungsdienst zu übergeben.

`javax.security.auth.callback.TextOutputCallback`
Der Broker verwendet diesen Callback-Typ, um Protokollierungsdienste für den Authentifizierungsdienst bereitzustellen, indem die Textausgabe in der Protokolldatei des Brokers protokolliert wird. Die Callback-Meldungstypen `ERROR`, `INFORMATION`, `WARNING` werden den Broker-Protokollebenen `ERROR`, `INFO`, und `WARNING` zugeordnet.

2. Erstellen Sie eine JAAS-Konfigurationsdatei mit Einträgen, welche auf die Anmeldemodulklassen verweisen und den Speicherort dieser Datei für den Message Queue-Administrator angeben. (Die Datei kann sich in einem Remote-Verzeichnis befinden, und der Speicherort kann als URL angegeben werden.)
3. Beachten Sie den Namen des Eintrags (der die Implementierungsklassen für die Anmeldung referenziert) in der JAAS-Konfigurationsdatei.
4. Archivieren Sie die Klassen zur Implementierung der Anmeldemodule in einer .jar-Datei, und platzieren Sie die .jar-Datei im Message Queue-Verzeichnis `lib/ext`.
5. Konfigurieren Sie die Broker-Eigenschaften, die sich auf die JAAS-Unterstützung beziehen. Diese Eigenschaften sind in Tabelle 1–2 beschrieben.
6. Legen Sie die folgende Systemeigenschaft fest, um den Speicherort der JAAS-Konfigurationsdatei anzugeben.

   `java.security.auth.login.config=` *Speicherort der JAAS-Konfigurationsdatei*

   Sie können die Konfigurationsdatei beispielsweise beim Starten des Brokers angeben.

   `imqbrokerd -Djava.security.auth.login.config=`*Speicherort der JAAS-Konfigurationsdatei*

   Der Speicherort der JAAS-Konfigurationsdatei kann auch über andere Methoden angegeben werden. Weitere Informationen finden Sie unter

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

In der folgenden Tabelle sind die Broker-Eigenschaften aufgeführt, die zum Einrichten der JAAS-Unterstützung benötigt werden.

TABELLE 1–2 Broker-Eigenschaften für JAAS-Unterstützung

Eigenschaft	Beschreibung
`imq.authentication.type`	Wählen Sie den Wert `basic`, um die Base-64-Passwortverschlüsselung festzulegen. Dies ist der einzig zulässige Wert für die JAAS-Authentifizierung.
`imq.authentication.basic.user_repository`	Wählen Sie den Wert `jaas`, um die JAAS-Authentifizierung festzulegen.
`imq.accesscontrol.type`	Wählen Sie `file`.
`imq.user_repository.jaas.name`	Wählen Sie den Namen des gewünschten Eintrags (in der JAAS-Konfigurationsdatei), welcher die Anmeldemodule referenziert, die Sie als Authentifizierungsmechanismus verwenden möchten. Dies ist der Name, den Sie in Schritt 3 aufgezeichnet haben.
`imq.user_repository.jaas.userPrincipalClass`	Diese Eigenschaft wird von der Message Queue-Zugriffssteuerung verwendet und gibt die `java.security.Principal`-Implementierungsklasse in den Anmeldemodulen an, die der Broker zum Extrahieren des Prinzipalnamens verwendet, der das Benutzerelement in der Message Queue-Zugriffssteuerungsdatei darstellt. Wenn diese Eigenschaft nicht angegeben wird, wird stattdessen der Benutzername verwendet, der vom Message Queue-Client beim Anfordern einer Verbindung übergeben wurde.
`imq.user_repository.jaas.groupPrincipalClass`	Diese Eigenschaft wird von der Message Queue-Zugriffssteuerung verwendet und gibt die `java.security.Principal`-Implementierungsklasse in den Anmeldemodulen an, die der Broker zum Extrahieren des Prinzipalnamens verwendet, der das Gruppenelement in der Message Queue-Zugriffssteuerungsdatei darstellt. Wenn diese Eigenschaft nicht angegeben wird, werden die Gruppenregeln (sofern vorhanden) in der Message Queue-Zugriffssteuerungsdatei ignoriert.

Formatänderung für persistenten Speicher

In Version 4.1 von Message Queue wurde der JDBC-Speicher geändert, um Hochverfügbarkeit zu unterstützen. Aus diesem Grund lautet die Version des JDBC-Speichers nun 410. Die JDBC-Speicherversionen 350, 370 und 400 werden automatisch in das Format der Version 410 migriert.

Bitte beachten Sie, dass die Version des dateibasierten persistenten Speichers weiterhin 370 lautet, da dieser Speicher nicht geändert wurde.

Broker-Konfiguration

Die Eigenschaft `IMQ_DEFAULT_EXT_JARS` wurde zur Datei `imqenv.conf` hinzugefügt. Sie können diese Eigenschaft festlegen, um die Pfadnamen externer .jar-Dateien anzugeben, die beim Start des Brokers in `CLASSPATH` aufgenommen werden sollen. Wenn Sie diese Eigenschaft verwenden, um den Speicherort von externen `.jar`-Dateien anzugeben, müssen Sie diese Dateien nicht mehr in das Verzeichnis `lib/ext` kopieren. Externe .jar-Dateien können sich auf JDBC-Treiber oder JAAS-Anmeldemodule beziehen. Der folgende Beispielbefehl gibt den Speicherort von JDBC-Treibern an.

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

Unterstützung für das JES Monitoring Framework

Message Queue unterstützt das Sun Java Enterprise System (JES) Monitoring Framework, mit dem Java Enterprise System-Komponenten über eine gemeinsame grafische Oberfläche überwacht werden können. Diese Oberfläche wird durch die webbasierte Sun Java System Monitoring Console implementiert. Wenn Sie Message Queue gemeinsam mit anderen JES-Komponenten ausführen, ist es möglicherweise praktischer, eine einzige Oberfläche zur Verwaltung all dieser Komponenten zu verwenden.

Das JES Monitoring Framework definiert ein gemeinsames Datenmodell (CMM), das von allen JES-Komponenten verwendet wird. Dieses Modell ermöglicht eine zentralisierte und einheitliche Anzeige aller JES-Komponenten. Message Queue stellt die folgenden Objekte im JES Monitoring Framework bereit:

- das installierte Produkt
- den Namen der Broker-Instanz
- den Broker-Port-Mapper
- alle Verbindungsdienste
- alle physischen Ziele
- den persistenten Speicher
- das Benutzer-Repository

Jedes dieser Objekte wird einem CMM-Objekt zugeordnet, dessen Attribute über die JES Monitoring Console überwacht werden können. Administratoren können diese Konsole zur Laufzeit verwenden, um Leistungsstatistiken anzuzeigen, Regeln für eine automatische

Überwachung zu erstellen und Alarme zu bestätigen. Detaillierte Informationen zur Zuordnung von Message Queue-Objekten zu CMM-Objekten finden Sie im *Sun Java Enterprise System-Überwachungshandbuch*.

So können Sie die JES-Überwachung aktivieren

1. Installieren und konfigurieren Sie alle Komponenten in Ihrer Bereitstellung (Message Queue- und andere Komponenten) gemäß den Anweisungen im *Sun Java Enterprise System-Installationshandbuch*.
2. Aktivieren und konfigurieren Sie das Monitoring Framework für alle überwachten Komponenten wie im *Sun Java Enterprise System-Überwachungshandbuch* beschrieben.
3. Installieren Sie die Überwachungskonsole auf einem separaten Host, starten Sie den Master-Agent und anschließend den Webserver wie im *Sun Java Enterprise System-Überwachungshandbuch* beschrieben.

Die Verwendung des JES Monitoring Frameworks hat keine Auswirkungen auf die Broker-Leistung, da die gesamte Erfassung von Metriken vom Monitoring Framework ausgeführt wird, das Daten aus der vorhandenen Überwachungsdateninfrastruktur des Brokers abruft.

Transaktionsverwaltung

Zuvor konnte ein Administrator nur für Transaktionen mit dem Status `PREPARED` ein Rollback ausführen. Das heißt, dass beim nicht ordnungsgemäßen Beenden einer Sitzung, die Teil einer verteilten Transaktion war, für die Transaktion ein Status beibehalten wurde, der nicht durch den Broker-Administrator bereinigt werden konnte. In Message Queue 4.1 können Sie das Dienstprogramm imqcmd zum Bereinigen (für das Rollback) von Transaktionen verwenden, die den folgenden Status aufweisen: `STARTED`, `FAILED`, `INCOMPLETE`, `COMPLETE`, `PREPARED`.

Um zu ermitteln, ob für eine bestimmte Transaktion ein Rollback durchgeführt werden kann (insbesondere, wenn diese nicht den Status `PREPARED` aufweist), bietet das Dienstprogramm imqcmd zusätzliche Daten als Teil der `imqcmd query txn`-Ausgabe: Es stellt die Verbindungs-ID für die Verbindung bereit, welche die Transaktion gestartet hat, und gibt die Uhrzeit an, zu welcher die Transaktion erstellt wurde. Mithilfe dieser Informationen kann der Administrator ermitteln, ob für die Transaktion ein Rollback durchgeführt werden muss. Im Allgemeinen sollte der Administrator ein zu frühes Rollback für eine Transaktion vermeiden.

Feste Ports für C-Clientverbindungen

C-Clients können die `MQ_SERVICE_PORT_PROPERTY`-Verbindungseigenschaft verwenden, um einen festen Port für die Verbindung anzugeben. Dies kann sinnvoll sein, wenn die Kommunikation durch eine Firewall erfolgen soll oder wenn Sie den Port-Mapper-Dienst des Brokers umgehen müssen, der Ports dynamisch zuweist.

Bedenken Sie, dass der JMS-Dienst auch auf Broker-Seite konfiguriert werden muss. Wenn Sie Ihren Client z. B. über `ssljms` mit Port 1756 verbinden möchten, führen Sie die folgenden Schritte aus.

- Auf Client-Seite: Setzen Sie `MQ_SERVICE_PORT_PROPERTY` auf 1756 und `MQ_CONNECTION_TYPE_PROPERTY` auf `SSL`.
- Auf Broker-Seite: Setzen Sie die Eigenschaft `imq.`*serviceNameType.protocol* `.port` wie folgt auf 1756.

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

Hinweis – Die Verbindungseigenschaft `MQ_SERVICE_PORT_PROPERTY` wurde mit Version 3.7 Update 2 von Message Queue eingeführt.

Hardware- und Software-Anforderungen

Die Hardware- und Software-Anforderungen für Version 4.1 finden Sie im *Sun Java System Message Queue 4.1 Installation Guide* .

Informationen zu Message Queue 4.0

Message Queue 4.0 ist auf die Unterstützung von Application Server 9 PE beschränkt. Diese Nebenversion umfasst einige neue Funktionen, kleinere Erweiterungen sowie behobene Fehler. Dieser Abschnitt umfasst die folgenden Informationen.

Neuheiten in Version 4.0

Message Queue 4.0 umfasst die folgenden neuen Funktionen:

Die genannten Themen werden in den folgenden Unterabschnitten näher beschrieben.

Achtung – Eine der eher kleinen, jedoch wichtigen Änderungen, die mit Version 4.0 eingeführt wurden, ist die Tatsache, dass die Befehlszeilenoption nun nicht mehr zur Angabe eines Passworts verwendet werden kann. Daher müssen nun alle Passwörter in einer Datei gespeichert werden, wie unter „Veraltete Passwortoption" auf Seite 45 beschrieben.

Schnittstellenänderungen an der C-API und der C-Clientlaufzeit

Version 4.0 von Message Queue fügt zwei Eigenschaften hinzu, die für alle Nachrichten gesetzt werden, die in einer Warteschlange mit nicht zugestellten Nachrichten platziert wurden.

- `JMS_SUN_DMQ_PRODUCING_BROKER` zeigt dem Broker an, wo die Nachricht generiert wurde.
- `JMS_SUN_DMQ_DEAD_BROKER` zeigt dem Broker an, wer die Nachricht als nicht zugestellt gekennzeichnet hat.

Schnittstellenänderungen an der Java-API und der Java-Clientlaufzeit

Version 4.0 von Message Queue fügt zwei Eigenschaften hinzu, die für alle Nachrichten gesetzt werden, die in einer Warteschlange mit nicht zugestellten Nachrichten platziert wurden.

- `JMS_SUN_DMQ_PRODUCING_BROKER` zeigt dem Broker an, wo die Nachricht generiert wurde.
- `JMS_SUN_DMQ_DEAD_BROKER` zeigt dem Broker an, wer die Nachricht als nicht zugestellt gekennzeichnet hat.

Anzeigen von Informationen zum persistenten Speicher

Der Unterbefehl `query` wurde zum Befehl `imqdbmgr` hinzugefügt. Verwenden Sie diesen Unterbefehl zum Anzeigen von Informationen zum persistenten Speicher (u. a. Speicherversion, Datenbankbenutzer und ob die Datenbanktabellen erstellt wurden).

Im Folgenden finden Sie ein Beispiel für die Informationen, die über diesen Befehl angezeigt werden.

```
imqdbmgr query

[04/Oct/2005:15:30:20 PDT] Verwenden des integrierten Persistenzspeichers:
        Version=400
        Broker-ID=Mozart1756
        Datenbankverbindung url=jdbc:oracle:thin:@Xhome:1521:mqdb
        Datenbankbenutzer=scott
Ausführen im Standalone-Modus.
Datenbanktabellen wurden bereits erstellt.
```

Formatänderungen für die persistente Speicherung

In Version 3.7 UR1 von Message Queue wurden zur Verbesserung der Leistung zwei Änderungen am persistenten Speicherformat eingeführt. Eine Änderung betrifft den Dateispeicher, die andere den JDBC-Speicher.

- Format von Transaktionsdaten im Dateispeicher beibehalten

  Das Format von Transaktionsstatusinformationen, die im dateibasierten persistenten Speicher von Message Queue gespeichert werden, wurde geändert, um die Datenträger-E/A-Vorgänge zu reduzieren und die Leistung von JMS-Transaktionen zu verbessern.

- Oracle JDBC-Speicher

  In vorherigen Versionen von Message Queue hat das Speicherschema für Oracle den Datentyp `LONG RAW` zum Speichern von Nachrichtendaten verwendet. Mit Oracle 8 wurde der Datentyp `BLOB` eingeführt, der den Typ `LONG RAW` ablöste. In Message Queue 3.7 UR1 wurde zum Datentyp `BLOB` gewechselt, um die Leistung zu verbessern und bessere Unterstützungsmöglichkeiten zu bieten.

Da sich diese Änderungen auf die Speicherkompatibilität auswirken, wurde die Speicherversion für Dateispeicher und JDBC-Speicher in Version 3.7 UR1 von Message Queue von 350 in 370 geändert.

In Version 4.0 von Message Queue wurden Änderungen am JDBC-Speicher eingeführt, um die Leistung zu optimieren und zukünftige Erweiterungen zu unterstützen. Aus diesem Grund wurde die Version des JDBC-Speichers in 400 geändert. Beachten Sie, dass in Version 4.0 die Version des dateibasierten persistenten Speichers weiterhin 370 lautet, da hier keine Änderungen durchgeführt wurden.

Message Queue 4.0 unterstützt die automatische Konvertierung des persistenten Speichers in die neuesten Versionen der dateibasierten persistenten Speicher sowie der persistenten JDBC-Speicher. Beim erstmaligen Ausführen von `imqbrokerd` wird ein älterer Speicher (sofern vorhanden) in das neue Format migriert.

- Die Versionen 200 und 350 des dateibasierten Speichers werden in das Format der Version 370 migriert.
- Die Versionen 350 und 370 des JDBC-Speichers werden in das Format der Version 400 migriert. (Wenn Sie einen Speicher der Version 200 aktualisieren müssen, muss zunächst die Version 3.5 oder 3. 6 als Zwischenversion verwendet werden.)

Wenn Sie für dieses Upgrade ein Rollback durchführen müssen, können Sie Message Queue 4.0 deinstallieren und anschließend erneut die Version installieren, die zuvor ausgeführt wurde. Da die ältere Kopie des Speichers beibehalten wird, kann der Broker mit der älteren Kopie des Speichers ausgeführt werden.

Broker-Verwaltung

Im Dienstprogramm Command (`imqcmd`) sind ein neuer Unterbefehl und verschiedene neue Optionen verfügbar, über welche Administratoren den Broker inaktivieren, den Broker nach einem festgelegten Intervall herunterfahren, eine Verbindung löschen oder Java-Systemeigenschaften hinzufügen können (z. B. verbindungsbezogene Eigenschaften).

- Bei der Inaktivierung des Brokers wird dieser in einen inaktiven Zustand versetzt, in dem Nachrichten entfernt werden können, bevor der Broker heruntergefahren oder neu gestartet wird. Zu einem Broker, der inaktiviert wird, können keine neuen Verbindungen erstellt werden. Um den Broker zu inaktivieren, geben Sie einen Befehl ähnlich dem folgenden ein.

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

- Um den Broker nach einem bestimmten Intervall herunterzufahren, geben Sie einen Befehl ähnlich dem folgenden ein. (Das Zeitintervall gibt die Anzahl an Sekunden bis zum Herunterfahren des Brokers an.)

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

  Wenn Sie ein Zeitintervall angeben, protokolliert der Broker eine Meldung, die darauf hinweist, wann er heruntergefahren wird. Beispiel:

  ```
  Broker wird in 29 Sekunden (29996 Millisekunden) heruntergefahren.
  ```

  Während der Broker auf das Herunterfahren wartet, wird sein Verhalten folgendermaßen beeinflusst.

  - Administrative JMS-Verbindungen werden weiterhin akzeptiert.
  - Neue JMS-Verbindungen werden nicht akzeptiert.
  - Bestehende JMS-Verbindungen werden weiterhin funktionieren.
  - Der Broker kann nicht die Funktionen für einen anderen Broker in einem Hochverfügbarkeits-Cluster übernehmen.
  - Das Dienstprogramm imqcmd wird nicht gesperrt, es wird die Anforderung zum Herunterfahren des Brokers senden und umgehend zurückgeben.

- Um eine Verbindung zu löschen, geben Sie einen Befehl ähnlich dem folgenden ein.

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

  Verwenden Sie den Befehl `imqcmd list cxn` oder `imqcmd query cxn`, um die Verbindungs-ID abzurufen.

- Zur Angabe einer Systemeigenschaft über den Befehl `imqcmd` verwenden Sie die neue –D-Option. Dies ist nützlich, um die Werkseinstellungen der JMS-Verbindung oder verbindungsbezogene Java-Systemeigenschaften außer Kraft zu setzen. Beispiel:

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

Vollständige Informationen zur Syntax des Befehls `imqcmd` finden Sie in Kapitel 13, „Command Line Reference” in *Sun Java System Message Queue 4.1 Administration Guide*.

JDBC-Persistenzunterstützung

Apache Derby Version 10.1.1 wird jetzt als JDBC-kompatibler Persistenzspeicheranbieter unterstützt.

SSL-Unterstützung

Ab Version 4.0 ist der Standardwert für die Werkseinstellung der Clientverbindung `imqSSLIsHostTrusted` auf `false` gesetzt. Wenn die Anwendung jedoch von dem vorherigen Standardwert `true` abhängt, müssen Sie die Eigenschaft neu konfigurieren und ausdrücklich auf `true` setzen.

Möglicherweise vertrauen Sie dem Host, wenn der Broker so konfiguriert ist, dass er selbst signierte Zertifikate verwendet. In diesem Fall sollten Sie zusätzlich zur Angabe, dass die Verbindung einen SSL-basierten Verbindungsdienst verwenden soll (über die imqConnectionType-Eigenschaft), die Eigenschaft `imqSSLIsHostTrusted` auf "true" setzen.

Um Clientanwendungen sicher auszuführen, wenn der Broker selbst signierte Zertifikate verwendet, verwenden Sie beispielsweise einen ähnlichen Befehl wie den folgenden.

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

Um das Verwaltungstool `imqcmd` sicher auszuführen, wenn der Broker selbst signierte Zertifikate verwendet, verwenden Sie einen ähnlichen Befehl wie den folgenden.

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

JMX-Unterstützung

Für die Konfiguration und Überwachung von Message Queue-Brokern wurde in Übereinstimmung mit der JMX-Spezifikation (Java Management Extensions) eine neue API hinzugefügt. Mithilfe dieser API haben Sie die Möglichkeit, Broker-Funktionen programmatisch in einer Message Queue-Clientanwendung zu konfigurieren und zu überwachen. In früheren Versionen von Message Queue war der Zugriff auf diese Funktionen ausschließlich über die Befehlszeile oder die Administrationskonsole möglich.

Die API besteht aus mehreren JMX-*MBeans (Managed Beans)* zur Verwaltung der folgenden mit Message Queue verbundenen Ressourcen:

- Nachrichten-Broker
- Verbindungsdienste
- Verbindungen
- Ziele

- Nachrichtenproduzenten
- Nachrichtenkonsumenten
- Transaktionen
- Broker-Cluster
- Protokollierung
- Die JVM (Java Virtual Machine)

Diese MBeans stellen *Attribute* und *Operationen* für das synchrone Abrufen und Manipulieren des Status der zugrunde liegenden Ressourcen bereit, sowie *Benachrichtigungen*, über die eine Clientanwendung Statusänderungen überwachen und asynchron darauf antworten kann, sobald diese auftreten. Mithilfe der JMX-API können Clientanwendungen Konfigurations- und Überwachungsaufgaben wie die folgenden durchführen:

- Festlegen von Portnummern für Broker
- Festlegen der maximalen Nachrichtengröße für Broker
- Anhalten eines Verbindungsdienstes
- Festlegen der maximalen Anzahl an Threads für einen Verbindungsdienst
- Abrufen der aktuellen Anzahl der Verbindungen mit einem Dienst
- Löschen einer Verbindung
- Erstellen eines Zieles
- Löschen eines Zieles
- Aktivieren oder deaktivieren der automatischen Erstellung von Zielen
- Bereinigen aller Nachrichten in einem Ziel
- Abrufen der Gesamtzahl an Nachrichten, die von einem Ziel seit dem Broker-Start empfangen wurden
- Abrufen des aktuellen Status (ausgeführt oder angehalten) einer Warteschlange
- Abrufen der aktuellen Anzahl an Nachrichtenproduzenten für ein Thema
- Bereinigen aller Nachrichten eines dauerhaften Abonnenten
- Abrufen der aktuellen JVM-Heap-Größe

Eine Einführung in die JMX-API und vollständige Referenzinformationen finden Sie im *Sun Java System Message Queue 4.1 Developer's Guide for JMX Clients* .

Broker-Unterstützung: JMX-bezogene Eigenschaften

Zur Unterstützung der JMX-API wurden mehrere neue Broker-Eigenschaften hinzugefügt(siehe Tabelle 1–3). Keine dieser Eigenschaften kann über die Befehlszeile mit dem Message Queue-Dienstprogramm Command (`imqcmd`) festgelegt werden. Stattdessen besteht die Möglichkeit, diese über die `-D`-Option des Broker-Dienstprogramms (`imqbrokerd`) festzulegen oder sie manuell in der Instanzkonfigurationsdatei des Brokers (`config.properties`) zu bearbeiten. Zudem lassen sich einige dieser Eigenschaften

(`imq.jmx.rmiregistry.start`, `imq.jmx.rmiregistry.use`, `imq.jmx.rmiregistry.port`) mit den neuen Broker-Dienstprogrammoptionen festlegen, die in Tabelle 1–4 beschrieben werden. In der Tabelle sind sämtliche Optionen mit Angaben zu Typ und Verwendung enthalten.

TABELLE 1–3 Neue Broker-Eigenschaften für JMX-Unterstützung

Eigenschaft	Typ	Beschreibung
`imq.jmx.rmiregistry.start`	`Boolean`	Gibt an, ob die RMI-Registrierung beim Broker-Start gestartet wird. Wenn der Wert `true` lautet, startet der Broker eine RMI-Registrierung bei dem von `imq.jmx.rmiregistry.port` angegebenen Port und verwendet diesen, um den RMI-Stub für JMX-Connectors zu speichern. Beachten Sie, dass der Wert von `imq.jmx.rmiregistry.use` in diesem Fall ignoriert wird. **Standardwert:** `false`
`imq.jmx.rmiregistry.use`	`Boolean`	Gibt an, ob eine externe RMI-Registrierung verwendet wird. Gilt nur, wenn der Wert für `imq.jmx.rmiregistry.start` `false` lautet. Wenn der Wert `true` lautet, verwendet der Broker eine externe RMI-Registrierung bei dem von `imq.jmx.rmiregistry.port` angegebenen Port, um den RMI-Stub für JMX-Connectors zu speichern. Die externe RMI-Registrierung muss bereits beim Broker-Start ausgeführt werden. **Standardwert:** `false`
`imq.jmx.rmiregistry.port`	`Integer`	Portnummer der RMI-Registrierung Gilt nur, wenn der Wert für `imq.jmx.rmiregistry.start` oder `imq.jmx.rmiregistry.use` `true` lautet. JMX-Connectors können anschließend so konfiguriert werden, dass sie die RMI-Registrierung verwenden, indem sie diese Portnummer im URL-Pfad der JMX-Dienst-URLs einfügen. **Standardwert:** `1099`
`imq.jmx.connector.list`	`String`	Namen von vorkonfigurierten JMX-Connectors (getrennt durch Kommas) **Standardwert:** `jmxrmi,ssljmxrmi`

TABELLE 1–3 Neue Broker-Eigenschaften für JMX-Unterstützung *(Fortsetzung)*

Eigenschaft	Typ	Beschreibung
`imq.jmx.connector.activelist`	`String`	Namen von JMX-Connectors, die beim Broker-Start aktiviert werden sollen (getrennt durch Kommas) **Standardwert:** `jmxrmi`
`imq.jmx.connector.`*Connector-Name*`.urlpath`	`String`	*URL-Pfad*-Komponente der JMX-Dienst-URL für Connector *Connector-Name* Nützlich in Fällen, in denen der URL-Pfad für den JMX-Dienst ausdrücklich festgelegt werden muss (z. B. bei der Verwendung einer externen RMI-Registrierung). **Standardwert:** Wenn eine RMI-Registrierung zum Speichern des RMI-Stub für JMX-Connectors verwendet wird (d. h., wenn `imq.jmx.registry.start` oder `imq.jmx.registry.use` auf `true` gesetzt sind) `/jndi/rmi://`*Broker-Host*`:`*RMI-Port* */Broker-Host/Broker-Port/Connector-Name* Wenn keine RMI-Registrierung verwendet wird (standardmäßig lautet der Wert für `imq.jmx.registry.start` und `imq.jmx.registry.use` `false`): `/stub/`*RMI-Stub* Wobei *RMI-Stub* eine verschlüsselte und serielle Darstellung des RMI-Stub selbst ist
`imq.jmx.connector.`*Connector-Name*`.useSSL`	`Boolean`	Gibt an, ob ein SSL (Secure Socket Layer) für Connector *Connector-Name* verwendet wird. **Standardwert:** `false`

TABELLE 1–3 Neue Broker-Eigenschaften für JMX-Unterstützung *(Fortsetzung)*

Eigenschaft	Typ	Beschreibung
`imq.jmx.connector.`*Connector-Name*`.brokerHostTrusted`	`Boolean`	Gibt an, ob vom Broker für Connector *Connector-Name* bereitgestellte Zertifikate vertraut wird. Gilt nur, wenn der Wert für `imq.jmx.connector.`*Connector-Name*`.useSSL` `true` lautet. Lautet der Wert `false`, überprüft die Message Queue-Clientlaufzeit alle bereitgestellten Zertifikate. Die Validierung schlägt fehl, wenn die Signatur des Zertifikats nicht im Vertrauensspeicher des Clients enthalten ist. Wenn der Wert `true` lautet, wird die Validierung des Zertifikats übersprungen. Dies kann beispielsweise beim Testen von Software hilfreich sein, wenn ein selbst signiertes Zertifikat verwendet wird. **Standardwert:** `false`

Die Eigenschaft `imq.jmx.connector.list` definiert eine Reihe benannter JMX-Connectors, die beim Broker-Start erstellt werden; `imq.jmx.connector.activelist` legt fest, welcher dieser Connectors aktiviert werden soll. Jeder benannte Connector verfügt somit über eigene Eigenschaften:

`imq.jmx.connector.`*Connector-Name* `.urlpath`
`imq.jmx.connector.`*Connector-Name* `.useSSL`
`imq.jmx.connector.`*Connector-Name* `.brokerHostTrusted`

Standardmäßig werden zwei JMX-Connectors namens `jmxrmi` und `ssljmxrmi` erstellt. Entsprechend der Konfiguration verwendet der erste keine SSL-Verschlüsselung (`imq.jmx.connector.jmxrmi.useSSL` = `false` und der zweite verwendet diese (`imq.jmx.connector.ssljmxrmi.useSSL` = `true`). Per Voreinstellung ist nur der `jmxrmi`-Connector beim Broker-Start aktiviert. Weitere Informationen zum Aktivieren des `ssljmxrmi`-Connector für eine sichere Kommunikation finden Sie unter „SSL-Unterstützung für JMX-Clients" auf Seite 27.

Zudem wurden neue Optionen (Tabelle 1–4) zum Befehlszeilendienstprogramm Broker hinzugefügt (`imqbrokerd`), um die Nutzung, den Startvorgang und den Port für die RMI-Registrierung zu steuern. Die Verwendungs- und Wirkungsweise dieser Optionen entspricht der der äquivalenten Broker-Eigenschaften, die in Tabelle 1–3 beschrieben sind. In der Tabelle sind alle Optionen mit der jeweiligen äquivalenten Broker-Eigenschaft und Beschreibung aufgeführt.

TABELLE 1–4 Neue Broker-Dienstprogrammoptionen für JMX-Unterstützung

Option	Äquivalente Broker-Eigenschaft	Beschreibung
`-startRmiRegistry`	`imq.jmx.rmiregistry.start`	Gibt an, ob die RMI-Registrierung beim Broker-Start gestartet wird.
`-useRmiRegistry`	`imq.jmx.rmiregistry.use`	Gibt an, ob eine externe RMI-Registrierung verwendet wird.
`-rmiRegistryPort`	`imq.jmx.rmiregistry.port`	Die Portnummer der RMI-Registrierung

Ein neuer Unterbefehl (Tabelle 1–5) wurde zum Befehlszeilendienstprogramm Command (`imqcmd`) für die Auflistung der JMX-Dienst-URLs von JMX-Connectors hinzugefügt, die beim Broker-Start erstellt und gestartet wurden. Diese Information wird von JMX-Clients benötigt, die nicht die Message Queue-Convenience-Klasse `AdminConnectionFactory` verwenden, um die JMX-Connectors abzurufen. Darüber hinaus kann sie zum Verwalten und Überwachen von Message Queue über einen generischen JMX-Browser, wie Java Monitoring und Management Console (`jconsole`), verwendet werden.

TABELLE 1–5 Neuer Unterbefehl für das Dienstprogramm Command

Unterbefehl	Beschreibung
`list jmx`	Listet JMX-Dienst-URLs von JMX-Connectors auf

SSL-Unterstützung für JMX-Clients

Wie oben bereits erwähnt, ist der Message Queue-Nachrichten-Broker standardmäßig für eine sichere Kommunikation über den vordefinierten JMX-Connector `jmxrmi` konfiguriert. Für Anwendungen muss zur Verwendung von SSL (Secure Socket Layer) für eine sichere Kommunikation der alternative, sichere JMX-Connector `ssljmxrmi` aktiviert werden. Dies erfordert die folgenden Schritte:

1. Rufen Sie ein signiertes Zertifikat ab, und installieren Sie dieses so, wie es für die Verbindungsdienste `ssljms`, `ssladmin` oder `cluster` im *Message Queue Administration Guide* beschrieben wird.
2. Installieren Sie das Root-Zertifizierungsstellenzertifikat im Vertrauensspeicher, falls erforderlich.
3. Fügen Sie den `ssljmxrmi`-Connector zur Liste der JMX-Connectors hinzu, damit dieser beim Broker-Start aktiviert wird:

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

4. Starten Sie den Broker mit dem Message Queue-Dienstprogramm Broker (`imqbrokerd`), indem Sie diesem entweder das Schlüsselspeicher-Passwort in einer Passwortdatei übergeben oder es bei Aufforderung in die Befehlszeile eingeben.

5. Standardmäßig ist der `ssljmxrmi`-Connector (oder jeder andere SSL-basierte Connector) konfiguriert, um sämtliche bereitgestellten Broker-SSL-Zertifikate zu validieren. Um diese Validierung zu verhindern (wenn Sie z. B. selbst signierte Zertifikate beim Testen von Software verwenden), setzen Sie die Broker-Eigenschaft `imq.jmx.connector.ssljmxrmi.brokerHostTrusted` auf `true`.

Auf Clientseite muss das Administrator-Verbindungsfactory (`AdminConnectionFactory`) mit einer URL konfiguriert sein, die `ssljmxrmi` als bevorzugten Connector angibt:

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

Verwenden Sie erforderlichenfalls die Systemeigenschaften `javax.net.ssl.trustStore` und `javax.net.ssl.trustStorePassword`, um den JMX-Client mit dem Vertrauensspeicher zu verknüpfen.

Protokollierung zur Client-Laufzeit

In diesem Abschnitt wird die Message Queue 4.0-Unterstützung für die Protokollierung von verbindungs- und sitzungsbezogenen Ereignissen zur Client-Laufzeit beschrieben.

JDK 1.4 (und höher) enthält die `java.util.logging`-Bibliothek. Diese Bibliothek implementiert eine standardmäßige Protokollschnittstelle, die zur anwendungsspezifischen Protokollierung verwendet werden kann.

Die Message Queue-Client-Laufzeit nutzt die Java-Protokoll-API, um deren Protokollierungsfunktionen zu implementieren. Sie können sämtliche J2SE 1.4-Protokolloptionen zum Konfigurieren der Protokollierungsaktivitäten verwenden. Beispielsweise können von einer Anwendung die folgenden Java-Protokolloptionen verwendet werden, um festzulegen, wie die Protokollinformationen von der Message Queue-Client-Laufzeit ausgegeben werden:

- Protokoll-Handler
- Protokollfilter
- Protokollformatierungen
- Protokollebene

Weitere Informationen zur Java-Protokoll-API finden Sie in der Übersicht über die Java-Protokollierung unter
`http://java.sun.com/j2se/1.4.2/docs/guide/util/logging/overview.html`

Protokollnamensräume, -ebenen und -aktivitäten

Der Message Queue-Anbieter definiert mehrere Protokollnamensräume, die mit Protokollebenen und Protokollaktivitäten verknüpft sind, über die Message Queue-Clients Verbindungs- und Sitzungsereignisse protokollieren können, sofern die Protkollkonfiguration entsprechend festgelegt wurde.

Der Root-Protokollnamensraum für die Message Queue-Client-Laufzeit ist als `javax.jms` definiert. Für alle Protokolle in der Message Queue-Client-Laufzeit wird dieser Name als übergeordneter Namensraum verwendet.

Die für die Message Queue-Client-Laufzeit verwendeten Protokollebenen stimmen mit denen überein, die in der `java.util.logging.Level`-Klasse definiert sind. Diese Klasse definiert sieben Standardprotokollebenen und zwei zusätzliche Einstellungen, über die Sie die Protokollierung aktivieren bzw. deaktivieren können.

`OFF`	Deaktiviert die Protokollierung.
`SEVERE`	Höchste Priorität, höchster Wert. Anwendungsdefiniert.
`WARNING`	Anwendungsdefiniert.
`INFO`	Anwendungsdefiniert.
`CONFIG`	Anwendungsdefiniert.
`FINE`	Anwendungsdefiniert.
`FINER`	Anwendungsdefiniert.
`FINEST`	Niedrigste Priorität, niedrigster Wert. Anwendungsdefiniert.
`ALL`	Aktiviert die Protokollierung aller Nachrichten.

Im Allgemeinen werden Ausnahmen und Fehler, die zur Message Queue-Client-Laufzeit auftreten, im Protokoll mit dem `javax.jms`-Namensraum erfasst.

- Ausnahmefehler, die von der JVM ausgelöst und von der Client-Laufzeit abgefangen werden, z. B. `IOException`, werden bei der Protokollierung mit dem Protokollnamensraum `javax.jms` auf der Ebene `WARNING` erfasst.
- JMS-Ausnahmen, die von der Client-Laufzeit ausgelöst werden, z. B. `IllegalStateException` , werden bei der Protokollierung mit dem Protokollnamensraum `javax.jms` auf der Ebene `FINER` erfasst.
- Fehler, die von der JVM zurückgegeben und von der Client-Laufzeit abgefangen werden, z. B. `OutOfMemoryError`, werden bei der Protokollierung mit dem Protokollnamensraum `javax.jms` auf der Ebene `SEVERE` erfasst.

In den folgenden Tabellen sind die Ereignisse aufgelistet, die protokolliert werden können, sowie die Protokollebene, die festgelegt werden muss, um Ereignisse für JMS-Verbindungen und für Sitzungen zu protokollieren.

Die folgende Tabelle enthält Beschreibungen der Protokollebenen und Ereignisse für Verbindungen.

TABELLE 1–6 Protokollebenen und Ereignisse für den `javax.jms.connection` -Namensraum

Protokollebene	Ereignisse
`FINE`	Verbindung erstellt
`FINE`	Verbindung geöffnet
`FINE`	Verbindung geschlossen
`FINE`	Verbindung unterbrochen
`FINE`	Verbindung wiederhergestellt
`FINER`	Diverse Verbindungsaktivitäten wie z. B. `setClientID`
`FINEST`	Nachrichten, Bestätigungen, Message Queue-Aktion und Steuerungsmeldungen (wie das Durchführen einer Transaktion)

Für Sitzungen werden die folgenden Informationen im Protokolleintrag erfasst.

- Jeder Protokolleintrag für eine an den Konsumenten übermittelte Nachricht umfasst ConnectionID, SessionID und ConsumerID.
- Jeder Protokolleintrag für eine Nachricht, die von einem Produzenten gesendet wurde, umfasst ConnectionID, SessionID, ProducerID und Zielname.

Die folgende Tabelle enthält Beschreibungen der Protokollebenen und Ereignisse für Sitzungen.

TABELLE 1–7 Protokollebenen und Ereignisse für den `javax.jms.session` -Namensraum

Protokollebene	Ereignis
`FINE`	Sitzung erstellt
`FINE`	Sitzung geschlossen
`FINE`	Produzent erstellt
`FINE`	Konsument erstellt
`FINE`	Ziel erstellt
`FINER`	Diverse Sitzungaktivitäten wie z. B. das Durchführen einer Sitzung.
`FINEST`	Produzierte und konsumierte Nachrichten. (Nachrichteneigenschaften und -texte werden in den Protokolleinträgen nicht erfasst.)

Die Ausgabeprotokollebene wird standardmäßig aus der JRE übernommen, in der die Anwendung ausgeführt wird. Überprüfen Sie die Datei `JRE_DIRECTORY/lib/logging.properties` , um diese Ebene zu bestimmen.

Sie haben die Möglichkeit, die Protokollierung programmatisch oder mithilfe von Konfigurationsdateien zu konfigurieren. Zudem können Sie den Umfang der Protokollierung steuern. In den folgenden Abschnitten werden diese Möglichkeiten erläutert.

Verwenden der Konfigurationsdatei für das JRE-Protokoll

Das folgende Beispiel zeigt, wie Protokollnamensräume und -ebenen in der Datei `JRE_DIRECTORY/lib/logging.properties` angegeben werden, die verwendet wird, um die Protokollebene für die Java Runtime Environment festzulegen. Sämtliche Anwendungen, die diese JRE verwenden, verfügen über dieselbe Protokollkonfiguration. In der nachstehenden Beispielkonfiguration wird die Protokollebene für den `javax.jms.connection` -Namensraum auf `INFO` gesetzt, und angegeben, dass die Ausgabe in `java.util.logging.ConsoleHandler` geschrieben werden soll.

```
#logging.properties file.
# "Handler" geben eine durch Kommas getrennte Liste der Protokoll-Handler-Klassen
# an. Diese Handler werden während des VM-Starts installiert.
# Beachten Sie, dass sich diese Klassen im selben Systemklassenpfad befinden müssen.
# Standardmäßig wird nur ein ConsoleHandler konfiguriert, der
# nur Meldungen auf INFO-Ebene und höheren Ebenen anzeigt.

    handlers= java.util.logging.ConsoleHandler

# Standardmäßige globale Protokollebene.
# Dies gibt an, welche Ereignisse in allen Protokollen
# erfasst werden. Für jede vorgegebene Funktion kann diese allgemeine Ebene
# von einer funktionsspezifischen Ebene überschrieben werden.
# Beachten Sie, dass der ConsoleHandler ebenfalls über eine separate Ebeneneinstellung verfügt,
# über die in der Konsole gedruckten Meldungen eingeschänkt werden können.

    .level= INFO

# Grenzen Sie die Meldungen ein, die auf der Konsole für INFO und höher gedruckt werden.

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

# Bei der Protokollierung mit javax.jms.connection-Namensraum
# werden Level.INFO-Nachrichten für die entsprechenden Ausgabe-Handler geschrieben.
# In dieser Konfiguration ist der Ausgabe-Handler auf java.util.logging.ConsoleHandler gesetzt.

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

Verwenden einer Protokollierungskonfigurationsdatei für eine bestimmte Anwendung

Sie können zudem eine Protokollierungskonfigurationsdatei über die Java-Befehlszeile definieren, die Sie verwenden, um eine Anwendung auszuführen. Die Anwendung wird dann die Konfiguration in der angegebenen Protokolldatei verwenden. Im folgenden Beispiel verwendet `configFile` dasselbe Format, das in der Datei `JRE_DIRECTORY/lib/logging.properties` definiert ist.

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

Programmatisches Festlegen der Protokollkonfiguration

Im folgenden Code wird die `java.util.logging`-API zum Protokollieren von Verbindungsereignissen verwendet, indem die Protokollebene für den `javax.jms.connection`-Namensraum auf `FINE` geändert wird. Sie können einen solchen Code in die Anwendung einfügen, um die Protokollkonfiguration programmatisch festzulegen.

```
import java.util.logging.*;
//einen Datei-Handler erzeugen und in der mq.log-Datei
//im Temp-Verzeichnis des Systems ausgeben.

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

//Protokoll für Domäne "javax.jms.connection" abrufen.

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

//javax.jms.connection-Protokollierung würde Aktivitäten
//der Ebene FINE und höher erfassen.

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

Verbindungsereignisbenachrichtigung

Verbindungsereignisbenachrichtigungen ermöglichen einem Message Queue-Client das Schließen und erneute Verbinden von Ereignissen zu überwachen und dem Benachrichtigungstyp und Verbindungsstatus entsprechende Aktionen auszuführen. Wenn es beispielsweise zu einem Failover kommt und der Client mit einem anderen Broker erneut verbunden wird, wird in einer Anwendung möglicherweise der Transaktionsstatus bereinigt, um mit einer neuen Transaktion fortzufahren.

Wenn der Message Queue-Anbieter ein schwerwiegendes Problem mit einer Verbindung ermittelt, ruft dieser den registrierten Ausnahme-Listener des Verbindungsobjekts auf. Er ruft die `onException`-Methode des Listeners auf, und übergibt dieser ein `JMSException`-Argument,

welches das Problem beschreibt. Der Message Queue-Anbieter bietet zudem eine Ereignisbenachrichtigungs-API, die es der Client-Laufzeit ermöglicht, die Anwendung über Änderungen des Verbindungsstatus zu informieren. Die Benachrichtigungs-API wird durch die folgenden Elemente definiert:

- Das `com.sun.messaging.jms.notification`-Paket, das den Ereignis-Listener und die Benachrichtigungsereignisobjekte definiert.
- Die `com.sun.messaging.jms.Connection`-Schnittstelle, die die Erweiterungen der `javax.jms.Connection`-Schnittstelle definiert.

In den folgenden Abschnitten werden die Ereignisse beschrieben, die Benachrichtigungen auslösen können, und wie ein Ereignis-Listener erstellt wird.

Verbindungsereignisse

In der folgenden Tabelle sind die Ereignisse aufgelistet und beschrieben, die vom Ereignis-Listener zurückgegeben werden können.

Beachten Sie, dass der JMS-Ausnahme-Listener nicht aufgerufen wird, wenn ein Verbindungsereignis ausgelöst wird. Der Ausnahme-Listener wird nur dann aufgerufen, wenn die Client-Laufzeit die maximale Anzahl an Versuchen zum Wiederherstellen der Verbindung erreicht hat. Die Client-Laufzeit ruft den Ereignis-Listener immer vor dem Ausnahme-Listener auf.

TABELLE 1–8 Benachrichtigungsereignisse

Ereignistyp	Bedeutung
`ConnectionClosingEvent`	Die Message Queue-Client-Laufzeit generiert dieses Ereignis, sobald sie die Benachrichtigung vom Broker empfängt, dass eine Verbindung aufgrund einer Administratoranforderung zum Herunterfahren geschlossen wird.
`ConnectionClosedEvent`	Die Message Queue-Client-Laufzeit generiert dieses Ereignis, sobald eine Verbindung aufgrund eines Broker-Fehlers geschlossen wird oder diese aufgrund einer Administratoranforderung zum Herunterfahren oder Neustarten geschlossen wird. Wenn ein Ereignis-Listener ein `ConnectionClosedEvent`-Ereignis empfängt, kann die Anwendung mithilfe der `getEventCode()`-Methode des empfangenen Ereignisses einen Ereigniscode abrufen, der die Ursache für den Schließvorgang angibt.

TABELLE 1–8 Benachrichtigungsereignisse *(Fortsetzung)*

Ereignistyp	Bedeutung
`ConnectionReconnectedEvent`	Die Verbindung der Message Queue-Client-Laufzeit und einem Broker wurde wiederhergestellt. Hierbei kann es sich um denselben Broker handeln, mit dem der Client zuvor bereits verbunden war, oder um einen anderen Broker. Eine Anwendung kann mithilfe der `getBrokerAddress`-Methode des empfangenen Ereignisses die Adresse des Brokers abrufen, mit dem diese erneut verbunden wurde.
`ConnectionReconnectFailedEvent`	Die Verbindung zwischen der Message Queue-Client-Laufzeit und einem Broker konnte nicht wiederhergestellt werden. Bei jedem fehlgeschlagenen Verbindungsversuch generiert die Laufzeit ein neues Ereignis und sendet dieses zum Ereignis-Listener. Der JMS-Ausnahme-Listener wird nicht aufgerufen, wenn ein Verbindungsereignis ausgelöst wird. Er wird nur dann aufgerufen, wenn die Client-Laufzeit die maximale Anzahl an Versuchen zum Wiederherstellen der Verbindung erreicht hat. Die Client-Laufzeit ruft den Ereignis-Listener immer vor dem Ausnahme-Listener auf.

Erstellen eines Ereignis-Listeners

Das folgende Codebeispiel zeigt, wie ein Verbindungs-Ereignis-Listener festgelegt wird. Bei jedem Verbindungsereignis wird die `onEvent` -Methode des Ereignis-Listeners von der Client-Laufzeit aufgerufen.

```
//MQ-Verbindungsfactory erstellen.

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

//MQ-Verbindung erstellen.

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

//MQ -Ereignis-Listener erstellen. Der Listener implementiert
//com.sun.messaging.jms.notification.EventListener-Schnittstelle.

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

```
//Ereignis-Listener für MQ-Verbindung festlegen.

connection.setEventListener ( eListener );
```

Beispiele für Ereignis-Listener

In diesem Beispiel erfasst der Ereignis-Listener der Anwendung das Verbindungsereignis im Protokollsystem der Anwendung:

```
public class ApplicationEventListener implementiert
               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();
          //Ereignisinformationen in den Ausgabe-Stream schreiben
         }
}
```

Hardware- und Software-Anforderungen

Informationen zu den Hardware- und Software-Anforderungen für Version 4.0 finden Sie in den Versionshinweisen zur Sun Java System Application Server-Plattform Version 9.

In dieser Version behobene Fehler

Die folgende Tabelle enthält die Fehler, die in Message Queue Version 4.1 behoben wurden.

TABELLE 1–9 In Message Queue 4.1 behobene Fehler

Fehler	Beschreibung
6381703	Transaktionale Remote-Nachrichten können zweifach übermittelt werden, wenn der Broker neu gestartet wird, von dem diese Nachrichten stammen.
6388049	Nicht abgeschlossene verteilte Transaktion kann nicht bereinigt werden.
6401169	Für die Übergabe- und Rollback-Optionen für imqcmd wird keine Bestätigungsaufforderung angezeigt.

TABELLE 1–9 In Message Queue 4.1 behobene Fehler *(Fortsetzung)*

Fehler	Beschreibung
6473052	Standard für automatisch erstellte Warteschlangen sollte Round Robin sein. (`MaxNumberConsumers` = -1).
6474990	Broker-Protokoll zeigt `ConcurrentModificationException` für `imqcmd list dst`-Befehl.
6487413	Speicherleck, wenn Verhalten bei Begrenzungen `REMOVE_OLDEST` oder `REMOVE_LOWER_PRIORITY` ist.
6488340	Broker reagiert nicht, und Client wartet auf Antwort zur Bestätigung.
6502744	Broker beachtet nicht das Standardlimit von 1000 Nachrichten in der Warteschlange für nicht zugestellte Nachrichten.
6517341	Client-Laufzeit muss die Logik für die Verbindungswiederherstellung verbessern, sobald der Client mit einem Hochverfügbarkeits-Cluster verbunden ist, indem der Client die Verbindung unabhängig vom Wert der Eigenschaft *imqReconnectEnabled* wiederherstellen kann.
6528736	Automatischer Windows-Startdienst (*imqbrokersvc*) stürzt während des Startvorgangs ab.
6561494	Nachrichten werden an den falschen Konsumenten zugestellt, wenn beide eine Sitzung teilen.
6567439	Produzierte Nachrichten in einer `PREPARED`-Transaktion werden nicht ordnungsgemäß zugestellt, wenn diese nach dem Neustart des Brokers gesendet werden.

In der folgenden Tabelle sind die in Message Queue 4.0 behobenen Probleme aufgelistet.

TABELLE 1–10 Behobene Probleme in Message Queue 4.0

Fehlernummer	Beschreibung
4986481	In Message Queue 3.5 kann sich die aufgerufene `Session.recover`-Methode im Modus für die automatische Neuverbindung aufhängen.
4987325	Erneut versendetes Flag war für die neu versendeten Nachrichten auf `false` gesetzt, nachdem die `Session.recover`-Methode aufgerufen wurde.
6157073	Ändern der neue Verbindungsmeldung, um die Anzahl an Verbindungen mit dem Dienst zusätzlich zur Gesamtzahl der Verbindungen hinzuzufügen.
6193884	Message Queue gibt unleserliche Nachrichten im syslog von Länderinformationen aus, die für Nachrichten keine ASCII-Zeichen verwenden.
6196233	Nachrichtenauswahl mithilfe von `JMSMessageID` funktioniert nicht.
6251450	`ConcurrentModificationException` für `connectList` während Clusterbeendigung.
6252763	`java.nio.BufferOverflowException` in `java.nio.HeapByteBuffer.putLong/Int .`

TABELLE 1–10 Behobene Probleme in Message Queue 4.0 *(Fortsetzung)*

Fehlernummer	Beschreibung
6260076	Erste Nachrichtenveröffentlichung nach Start bei Verwendung von Oracle-Speicher erfolgt langsam.
6260814	Auswahlverarbeitung auf `JMSXUserID` wird immer als `false` ausgewertet.
6264003	Der Warteschlangenbrowser zeigt Nachrichten an, die Bestandteil von Transaktionen sind, die noch nicht verarbeitet wurden.
6271876	Verbindungsdatensteuerung arbeitet nicht ordnungsgemäß, wenn ein Verbraucher mit nicht verarbeiteten Nachrichten geschlossen wird.
6279833	Message Queue sollte nicht zulassen, dass zwei Broker dieselben JDBC-Tabellen verwenden.
6293053	Master-Broker wird nicht ordnungsgemäß gestartet, wenn die IP-Adresse des Systems geändert wird, es sei denn, der Speicher wurde geleert (mithilfe von `–reset store`).
6294767	Message Queue-Broker muss `SO_REUSEADDR` auf dem Netzwerksocket setzen, der geöffnet wird.
6304949	`ClientID`-Eigenschaft für `TopicConnectionFactory` kann nicht festgelegt werden.
6307056	Dastxn-Protokoll führt zu einem Leistungsengpass.
6320138	Message Queue C-API kann den Namen einer Warteschlange für einen Reply-To-Header nicht ermitteln.
6320325	Der Broker wählt unter Solaris gelegentlich JDK 1.4 anstelle JDK 1.5, selbst wenn beide Versionen installiert sind.
6321117	Die Initialisierung eines Multibroker-Clusters führt zu einer `java.lang.NullPointerException .`
6330053	Der JMS-Client gibt `java.lang.NoClassDefFoundError` zurück, wenn eine Transaktion vom Abonnenten übergeben wird.
6340250	Unterstützung für `MESSAGE`-Typ in C-API.
6351293	Unterstützung für Apache Derby-Datenbank hinzufügen.

Wichtige Informationen

In diesem Abschnitt finden Sie die aktuellsten Informationen, die nicht in der eigentlichen Produktdokumentation enthalten sind. In diesem Abschnitt werden folgende Themen behandelt:

Installationshinweise

Informationen zu Anweisungen für die Vorbereitung der Installation, Upgrade-Verfahren sowie alle weiteren relevanten Informationen für die Installation von Message Queue-Plattformversion unter Solaris, Linux und Windows finden Sie im *Sun Java System Message Queue 4.1 Installation Guide* .

Anweisungen, die vor der Installation auszuführen sind und andere Informationen zur Installation von Message Queue Enterprise Edition zusammen auf Solaris-, Linux- und Windows-Plattformen finden Sie im *Sun Java Enterprise System Installation Guide*.

Weitere Informationen zu Upgrade und Migration auf Message Queue Enterprise Edition unter Solaris, Linux, HP-UX und Windows finden Sie im *Sun Java Enterprise System Upgrade and Migration Guide*.

Kompatibilitätsprobleme

In diesem Abschnitt werden bekannte Kompatibilitätsprobleme in Message Queue 4.1 beschrieben.

Schnittstellenstabilität

Sun Java System Message Queue verwendet zahlreiche Schnittstellen, die sich mit der Zeit ändern können. Eine Klassifizierung der Schnittstellen nach ihrer Stabilität finden Sie in Anhang B, „Stability of Message Queue Interfaces" in *Sun Java System Message Queue 4.1 Administration Guide*. Je stabiler eine Schnittstelle, desto weniger wahrscheinlich ist eine Änderung in den nachfolgenden Produktversionen.

Mögliche Probleme hinsichtlich der nächsten Hauptversion von Message Queue

Die nächste Hauptversion von Message Queue umfasst eventuell Änderungen, die eine Inkompatibilität mit Ihren Clients verursachen können. Wir teilen Ihnen diese Informationen jetzt mit, um Sie auf diese Änderungen vorzubereiten.

- Die Speicherorte einzelner Dateien, die als Bestandteil von Sun Java System Message Queue installiert werden, kann sich ändern. Dadurch können in vorhandenen Anwendungen Fehler auftreten, die vom aktuellen Speicherort bestimmter Message Queue-Dateien abhängen.
- Broker der Version 3.5 oder niedrigere Versionen können in einem Cluster mit neueren Brokern möglicherweise nicht mehr eingesetzt werden.
- In zukünftigen Versionen ist die Verwendung von früheren JDK-Versionen als 1.5 in Message Queue-Clients eventuell nicht mehr möglich.

Dokumentationsaktualisierungen für Message Queue 4.1

Abgesehen von diesem *Versionshinweise*-Dokument, enthält Message Queue 4.1 lediglich ein neues Dokument: *Sun Java System Message Queue 4.1 Developer's Guide for JMX Clients* . Dieses Dokument wurde mit Message Queue Version 4.0 eingeführt. In Version 4.1 wurden konzeptionelle Informationen hinzugefügt, die das JMX-Modell vorstellen.

Die Message Queue-Dokumentation, die für Message Queue 3.6 SP3 (2005Q4) veröffentlicht wurde, ist unter Berücksichtigung der Voraussetzungen für Application Server 9 PE-Clients auf dem neusten Stand. Dieser Dokumentationssatz steht auf folgender Webseite zur Verfügung.

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

Installations- und Upgradeinformationen

Der *Sun Java System Message Queue 4.1 Installation Guide* wurde mit plattformspezifischen Informationen aktualisiert. Dieses Dokument enthält jetzt relevante Installations- und Upgradeinformationen für Message Queue 4.1.

Administration Guide

Der *Administration Guide* wurde aktualisiert und bietet nun Informationen zu Hochverfügbarkeits-Clustern, JAAS-Unterstützung und JMX-Unterstützung.

Developer's Guide for Java Clients

Zum *Developer's Guide for Java Clients* wurden zusätzliche Informationen zur Client-Laufzeit-Protokollunterstützung und zu Verbindungsereignisbenachrichtigungen hinzugefügt.

Developer's Guide for C Clients

Der *Developer's Guide for C Clients* wurde aktualisiert. Es wurden Informationen zu `MQGetDestinationName`-Funktion,`MQ_Message`-Nachrichtentyp und festen Ports hinzugefügt.

Bekannte Probleme und Beschränkungen

In diesem Abschnitt werden die bekannten Probleme mit Message Queue 4.1 aufgeführt. Dies betrifft die folgenden Produktbereiche:

Eine Liste der aktuellen Fehler, deren Status und Umgehungsmöglichkeiten finden Sie als Mitglied der Java Developer Connection™ in der Bug Parade auf der Java Developer Connection-Website. Prüfen Sie die Informationen auf dieser Seite, bevor Sie einen neuen Fehler melden. Auch wenn nicht alle Message Queue-Fehler aufgelistet sind, ist diese Seite ein guter Ausgangspunkt, wenn Sie feststellen möchten, ob ein Problem bekannt gegeben wurde.

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

Hinweis – Die Mitgliedschaft bei der Java Developer Connection ist kostenlos, es ist jedoch eine Registrierung erforderlich. Auf der Sun Java-Webseite wird beschrieben, wie Sie Mitglied bei der Java Developer Connection werden.

Wenn Sie einen neuen Fehler melden oder eine Funktionsanfrage einreichen möchten, senden Sie eine E-Mail an `imq-feedback@sun.com` .

Installationsprobleme

In diesem Abschnitt werden die Installationsprobleme zu Message Queue Version 4.1 beschrieben.

Produktregistrierung und JES

Version 4.1 von Message Queue wird von einen neuen Installationsprogramm installiert, das zudem die gemeinsam genutzten Komponenten, die für Message Queue erforderlich sind, installiert und aktualisiert, z. B. JDK, NSS-Bibliotheken, JavaHelp usw. Diese Installationsprogramm und das JES-Installationsprogramm (Java Enterprise System) verwenden nicht dieselbe Produktregistrierung. Wenn eine Message Queue-Version, die zusammen mit JES installiert wurde, entfernt und durch Message Queue 4.1 mithilfe des Message Queue-Installationsprogramms ersetzt wird, ist der Status der JES-Produktregistrierung möglicherweise inkonsistent. Dies hat zur Folge, dass beim Ausführen des JES-Deinstallationsprogramms Message Queue 4.1 und die zugrunde liegenden gemeinsamen Komponenten, die es nicht installiert hat, eventuell versehentlich entfernt werden.

Zur Aktualisierung der Software, die nicht mit dem JES-Installationsprogramm installiert wurde, gehen Sie am besten wie folgt vor.

1. Entfernen Sie mit dem JES-Deinstallationsprogramm Message Queue sowie die gemeinsam genutzten Komponenten.

2. Installieren Sie mit dem Message Queue-Installationsprogramm Message Queue 4.1.

Auswählen der geeigneten JRE

Auf der Seite für die JDK-Auswahl des Message Queue 4.1-Installationsprogramms können Sie die im System vorhandene JDK/JRE für die Verwendung durch Message Queue auswählen. Leider enthält die angezeigte Liste ebenfalls die JRE, die zum Ausführen der Installationsanwendung verwendet wird. Diese JRE gehört zum Installationspaket und wird nicht tatsächlich auf dem System installiert. *(Fehler 6585911)*

Sie erkennen die vom Installationsprogramm verwendete JRE am Pfad, der innerhalb des entpackten Installationsverzeichnisses liegen und das Unterverzeichnis `mq4_1–installer` enthalten sollte. Beispiel:

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

Wählen Sie diese JRE nicht zur Verwendung durch Message Queue aus. Wählen Sie stattdessen ein anderes JDK im System aus. Wenn eins nicht vorhanden ist, führen Sie die für Ihre Plattform entsprechende Aktion aus.

- Solaris oder Linux: Wählen Sie die Option zum Installieren und verwenden des Standard-JDK aus.
- Windows: Laden Sie ein JDK herunter, und installieren Sie dieses, bevor Sie das Message Queue 4.1-Installationsprogramm ausführen.

Installation unter Windows

Wenn Sie Message Queue unter Windows installieren, sollten Sie die folgenden Einschränkungen beachten.

- Das Installationsprogramm fügt keine neuen Einträge für Message Queue zum Startmenü unter Programme hinzu (*Fehler 6567258*). Um die Administrationskonsole zu starten, verwenden Sie die Befehlszeile, wie in „Starting the Administration Console" in *Sun Java System Message Queue 4.1 Administration Guide* beschrieben.
- Das Installationsprogramm fügt das Verzeichnis `IMQ_HOME\mq\bin` nicht zur `PATH`-Umgebungsvariable hinzu.*(Fehler 6567197)*. Benutzer müssen entweder diesen Eintrag zur `PATH`-Umgebungsvariable hinzufügen oder einen vollständigen Pfadnamen angeben, wenn die Message Queue-Dienstprogramme (`IMQ_HOME\mq\bin\`*Befehl*) aufgerufen werden.
- Das Installationsprogramm fügt keine Einträge zur Windows-Registrierung hinzu, die darauf hinweisen, dass Message Queue installiert ist.
- Beim Ausführen im automatischen Modus wird das Installationsprogramm umgehend zurückgegeben. Die Installation wird zwar durchgeführt, jedoch weiß der Benutzer nicht, wann die automatische Installation tatsächlich abgeschlossen ist. *(Fehler 6586560)*

- Der Textmodus (`installer –t`) wird unter Windows nicht unterstützt. Beim Ausführen des Installationsprogramms im Textmodus unter Windows wird eine Fehlermeldung angezeigt. Diese Meldung wird auf English angezeigt, selbst wenn das Installationsprogramm mit einem anderen Gebietsschema ausgeführt wird. *(Fehler 6594142)*
- Die Zeichenfolge "Install Home" wird auf der entsprechenden Seite des Installationsprogramms auf Englisch angezeigt, selbst wenn das Installationsprogramm mit einem anderen Gebietsschema ausgeführt wird. *(Fehler 6592491)*

Installation unter Solaris

Die Fehlermeldung und der "unvollständige" Status der Zusammenfassung sind für Benutzer irreführend, die versuchen, die Installation über den Befehl `installer-n` aufzuführen. Der Befehl ist tatsächlich erfolgreich. *(Fehler 6594351)*

Installation unter Linux

Die folgenden Probleme betreffen die Installation auf einer Linux-Plattform.

- Auf dem Bildschirm zur JDK-Auswahl wird in der Scroll-Liste nur ein Element angezeigt. Dies macht es schwierig, weitere JDK in der Liste auszuwählen. *(Fehler 6584735)*
- Wenn das JDK aktuell ist, und der Benutzer auf der Seite für die JDK-Auswahl die Option zum Installieren des Standard-JDK wählt, fährt das Installationsprogramm mit den Installationsversuchen fort und gibt anschließend die Meldung aus, dass das Paket nicht installiert werden kann. Die Installation wird trotz dieses Problems erfolgreich abgeschlossen. *(Fehler 6581310)*
- Wenn das Installationsprogramm im Testlaufmodus (`installer –n`) ausgeführt wird, werden auf der Zusammenfassungsseite mehrere Fehlermeldungen angezeigt, und dass der Installationsstatus nicht abgeschlossen wurde. Dies ist falsch und irreführend. Bei einem Testlauf werden keine Komponenten auf dem System installiert, es wird lediglich die Antwortdatei erstellt, die anschließend für die Installation verwendet werden kann.*(Fehler 6594351)*
- Wenn ältere Versionen von Message Queue-Lokalisierungs-RPMs auf dem System vorhanden sind, schlägt die Installation der Lokalisierungs-RPMs von Message Queue 4.1 fehl. Dies geschieht, wenn Sie auf der entsprechenden Seite das Kontrollkästchen zum Installieren der mehrsprachigen Pakete für Message Queue aktivieren. Die Installation schlägt fehl, da es zu einem Konflikt mit Il8-Paketen aus einer früheren 3.7 UR1-Installation kommt. *(Fehler 6594381)*

  Umgehung Entfernen Sie Lokalisierungs-RPMs mithilfe des `rpm –e`-Befehls, bevor Sie das Installationsprogramm für 4.1 ausführen. Um die hierfür relevanten RPMs zu bestimmen, lesen Sie „Message Queue Packages (RPMs)” in *Sun Java System Message Queue 4.1 Installation Guide*.

Installation auf allen Plattformen

Diese Probleme betreffen die Installation auf allen Plattformen.

- Solange das Installationsprogramm die Installation von Message Queue 4.1 ausführt und die Seite mit der Fortschrittsanzeige angezeigt wird, ist die Schaltfläche "Abbrechen" aktiv. Das Klicken auf die Schaltfläche "Abbrechen" führt zu einer unvollständigen oder abgebrochenen Installation. *(Fehler 6595578)*
- Auf der Installationsübersichtsseite werden mehrere Verknüpfungen angezeigt, über die per Mausklick ein Viewer für Protokolle oder Übersichtsseiten angezeigt werden kann. Wenn Sie dieses Viewer-Fenster über die Schaltfläche "X" anstatt über die Schaltfläche "Schließen" schließen, kann dieses Viewer-Fenster nicht erneut geöffnet werden. *(Fehler 6587138)*

  Umgehung Verwenden Sie die Schaltfläche "Schließen", um das Fenster zu schließen.
- Wenn sich auf einem System ältere Versionen von Message Queue und NSS/NSPR befinden, listet das Installationsprogramm lediglich die erforderlichen Upgrades für Message Queue auf. Es wird nicht darauf hingewiesen, dass auch NSS/NSPR aktualisiert werden muss. Dieses Problem betrifft nur die Update-Seite, da die fragliche Software als Teil des Installationsvorgangs aktualisiert wird (auf der Seite "Bereit für die Installation" werden die richtigen Informationen angezeigt). *(Fehler 6580696)*

  Umgehung Keine erforderlich, da die NSS/NSPR-Dateien installiert werden, falls diese nicht aktuell sind, und ältere Versionen deinstalliert werden.
- Wenn das Installations- bzw. Deinstallationsprogramm im Textmodus ausgeführt wird (`installer –t`), wird auf der Zusammenfassungsseite zwar das Verzeichnis mit den Protokoll-/Zusammenfassungsdateien angezeigt, die Namen dieser Dateien werden jedoch nicht aufgelistet. *(Fehler 6581592)*
- Wird der Name einer nicht vorhandenen Datei angegeben, führt dies zu inkonsistenten und nicht eindeutigen Fehlermeldungen. *(Fehler 6587127)*

Versionsinformationen

Die Anzeige der Message Queue-Versionsinformationen ist im Installationsprogramm nicht transparent. *(Fehler 6586507)*

Für die Solaris-Plattform finden Sie in der nachstehenden Tabelle Informationen, um die zu installierende Version zu bestimmen.

TABELLE 1–11 Formate der Versionen

Im Installationsprogramm angezeigte Version	Message Queue-Version
4.1.0.0	4.1
3.7.0.1	3.7 UR1

TABELLE 1–11 Formate der Versionen *(Fortsetzung)*

Im Installationsprogramm angezeigte Version	Message Queue-Version
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
3.6.0.4	3.6 SP4

Hinweis – Für Patch-Versionen für 3.6 SP4 (z. B. 3.6 SP4 Patch 1) werden im Installationsprogramm dieselben Zeichenfolgen für die Versionen angezeigt. Sie müssen den Befehl `imqbrokerd –version` ausführen, um die genaue Version zu bestimmen.

Auf einer Linux-Plattform ist die Bereitstellung einer einfachen Formatüberstetzung nicht möglich. Die Versionsnummer, die im Installationsprogramm unter Linux angezeigt wird, weist folgendes Format auf.

```
<Hauptversionsnummer>.<Nebenversionsnummer>-<beliebige_Nummer>
```

Zum Beispiel könnte sie 3.7–22 lauten. Dies weist zwar darauf hin, dass es sich um eine der 3.7-Versionen, jedoch nicht um welche es sich genau handelt. Um diese zu bestimmen, führen Sie den Befehl `imqbrokerd –version` aus.

Lokalisierungsprobleme

Die folgenden Probleme beziehen sich auf die Lokalisierung.

- Wenn das Installationsprogramm im Textmodus ausgeführt wird (`installer –t`), werden bei anderen Gebietsschemata als Englisch Multibyte-Zeichen unleserlich angezeigt.*(Fehler 6586923)*
- Auf der Seite mit der Installationszusammenfassung wird dem Benutzer ein Zusammenfassungsbericht angezeigt. Dieser Bericht (eine HTML-Seite) enthält unleserliche Zeichen, wenn das Installationsprogramm mit einem Gebietsschema mit Multibyte-Zeichen ausgeführt wird. *(Fehler 6587112)*

  Umgehung Bearbeiten Sie die HTML-Datei, sodass diese den darin angegebenen Zeichensatz korrigiert. In der HTML-Datei sollte ein Inhalt vorhanden sein, der dem folgenden ähnlicht.

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

Ersetzen Sie "UTF-8" durch *Gebietsschema-Name*`.UTF-8`. Zum Beispiel `ja_JA.UTF-8` oder `ko.UTF-8` unter Solaris; `ja_JA.utf8` oder `ko_KO.utf8` unter Linux.

- Auf der Seite mit dem Installationsfortschritt zeigt der Fortschrittbalken merkwürdige Zeichen an. Die QuickInfo bei anderen Gebietsschemata als Englisch ist hartcodiert. *(Fehler 6591632)*
- Der Textmodus (`installer –t`) wird unter Windows nicht unterstützt. Beim Ausführen des Installationsprogramms im Textmodus unter Windows wird eine Fehlermeldung angezeigt. Diese Meldung ist nicht lokalisiert, wenn das Installationsprogramm mit einem anderen Gebietsschema als Englisch ausgeführt wird. *(Fehler 6594142)*
- Auf der Lizenzvereinbarungsseite des Installationsprogramms wird die Lizenzvereinbarung auf Englisch angezeigt, unabhängig davon, mit welchem Gebietsschema das Installationsprogramm ausgeführt wird.*(Fehler 6592399)*

  Umgehung Informationen für den Zugriff auf die lokalisierten Lizenzdateien finden Sie in der Datei `LICENSE_MULTILANGUAGE.pdf`.
- Der Hilfetext zur Verwendung des Installationsprogramms ist nicht lokalisiert. *(Fehler 6592493)*
- Die Zeichenfolge "None" auf der HTML-Zusammenfassungsseite des Installationsprogramms ist in der englischen Version hartcodiert. *(Fehler 6593089)*
- Die Seite mit den Copyright-Informationen ist für andere Gebietsschemata als Frankreich nicht lokalisiert. *(Fehler 6590992)*
- Wenn das Installationsprogramm mit dem deutschen Gebietsschema ausgeführt wird, wird auf der Begrüßungsseite nicht der vollständige Text wie bei anderen Gebietsschemata angezeigt. *(Fehler 6592666)*
- Die Zeichenfolge "Install Home" ist auf der entsprechenden Seite im Installationsprogramm nicht lokalisiert. Sie wird selbst dann auf Englisch angezeigt, wenn das Installationsprogramm mit einem anderen als dem englischen Gebietsschema ausgeführt wird. *(Fehler 6592491)*
- Wenn das Installationsprogramm im Textmodus ausgeführt wird (`installer –t`), lautet die englischen Antwortauswahl "Yes" und "No", unabhängig von dem Gebietsschema, in dem das Installationsprogramm ausgeführt wird. *(Fehler 6593230)*
- Auf der Installationsprogrammseite für die JDK-Auswahl ist die QuickInfo für die Schaltfläche zum Durchsuchen in der englischen Version hartcodiert. *(Fehler 6593085)*

Veraltete Passwortoption

In Vorgängerversionen von Message Queue konnten Sie die `–p`- oder `–password`-Option verwenden, um ein Passwort für die folgenden Befehle interaktiv anzugeben: `imqcmd`, `imqbrokerd` und `imdbmgr`. Mit Version 4.0 wurden diese Optionen verworfen. Sie müssen Passwörter nun wie folgt erzeugen.

1. Setzen Sie die Passworteigenschaft auf den gewünschten Wert in einer Datei, in der ausschließlich Passwörter gespeichert werden.

   Verwenden Sie die folgende Syntax, um Passwörter in der Passwortdatei festzulegen.

   Passworteigenschaftsname=Mein_Passwort
2. Übergeben Sie den Namen der Passwortdatei mithilfe der –`passfile`-Option.

Eine Passwortdatei kann mindestens eins der im Folgenden aufgelisteten Passwörter enthalten.

- Ein Schlüsselspeicherpasswort zum Öffnen des SSL-Schlüsselspeichers. Legen Sie dieses Passwort über die Eigenschaft `imq.keystore.password` fest.
- Ein LDAP-Repository-Passwort für die sichere Verbindung mit einem LDAP-Verzeichnis, wenn die Verbindung nicht anonym ist. Legen Sie dieses Passwort über die Eigenschaft `imq.user_repository.ldap.password` fest.
- Ein JDBC-Datenbankpasswort für die Verbindung zu einer JDBC-kompatiblen Datenbank. Legen Sie dieses Passwort über die Eigenschaft `imq.persist.jdbc.vendorName.password` fest. Die *vendorName*-Komponente des Eigenschaftsnamen ist eine Variable, die den Datenbankanbieter angibt. Zur Auswahl stehen `hadb`, `derby`, `pointbase`, `oracle` oder `mysql`.
- Ein Passwort für den `imqcmd`-Befehl (zum Ausführen von Broker-Administrationsaufgaben). Legen Sie dieses Passwort über die Eigenschaft `imq.imqcmd.password` fest.

Im folgenden Beispiel wird als Passwort für die JDBC-Datenbank `abracadabra` festgelegt.

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

Sie haben folgende Möglichkeiten, um den Broker so zu konfigurieren, dass er die von Ihnen erstelle Passwortdatei verwendet.

- Geben Sie die folgenden Eigenschaften in der `config.properties`-Datei des Brokers an.

  ```
  imq.passfile.enabled=true
  imq.passfile.dirpath=Mein_Dateiverzeichnis
  imq.passfile.name=Mein_Passwortdateiname
  ```
- Verwenden Sie die –`passfile`-Option des `imqbrokerd`-Befehls.

  `imqbrokerd –passfile` *Mein_Passwortdateiname*

Allgemeine Probleme

In diesem Abschnitt werden allgemeine Probleme in Message Queue 4.1 erläutert. Einige Probleme wurden bereits bei Vorgängerversionen von Message Queue bekannt gegeben.

- Wird ein JMS-Client bei Verwendung des HTTP-Transports plötzlich beendet (z.B. über `Strg-C`), benötigt der Broker etwa eine Minute, bevor die Clientverbindung und alle damit zusammenhängenden Ressourcen freigegeben werden.

  Wird innerhalb dieses Zeitraums eine weitere Instanz des Clients gestartet, die versucht, dieselbe Client-ID, Warteschlange oder dasselbe dauerhafte Abonnement zu verwenden, wird möglicherweise ein Ausnahmefehler "Client-ID wird bereits verwendet" ausgegeben. Dies stellt jedoch kein Problem dar, es handelt sich lediglich um eine Nebenwirkung des vorangehend beschriebenen Beendigungsvorgangs. Wenn der Client nach etwa einer Minute gestartet wird, sollte kein Fehler gemeldet werden.
- SOAP-Clients. Vorher musst sich die .jar-Datei für die SAAJ 1.2-Implementierung, die die Dateien `mail.jar` und `mail.jar` betraf, nicht in `CLASSPATH` befinden. In SAAJ 1.3 wurde diese Verknüpfung gelöscht, daher muss sich die Datei `mail.jar` in Message Queue-Clients ausdrücklich in `CLASSPATH` befinden.

Administration/Konfiguration

Die folgenden Probleme beziehen sich auf die Administration und Konfiguration von Message Queue.

- Die Dienstprogramme `imqadmin` und `imqobjmgr` geben einen Fehler aus, wenn `CLASSPATH` auf Windows-Computern doppelte Anführungszeichen enthält (*Fehlernummer 5060769*)

  Umgehung Sie können diese Fehlermeldung ignorieren. Der Broker informiert die Verbraucher ordnungsgemäß über mögliche Fehler. Dieser Fehler hat keine Auswirkungen auf die Zuverlässigkeit des Systems.
- Die Option `-javahome` in Solaris- und Windows-Skripts (alle Versionen) funktioniert nicht, wenn der bereitgestellte Wert ein Leerzeichen enthält (*Fehlernummer 4683029*).

  Die Option `javahome` wird von den Message Queue-Befehlen und -Programmen verwendet, um eine alternative Java 2-kompatible Runtime anzugeben. Der Pfadname zur alternativen Java-Runtime darf jedoch keine Leerzeichen enthalten. Nachfolgend werden einige Beispiele für Pfade mit Leerzeichen genannt:

  Windows: `C:/jdk 1.4`

  Solaris: `/work/java 1.4`

  Umgehung Installieren Sie die Java-Runtime an einem Speicherort oder unter einem Pfad, der keine Leerzeichen enthält.
- Das Attribut `imqQueueBrowserMaxMessagesPerRetrieve` legt die maximale Anzahl an Nachrichten fest, die von der Client-Runtime in einem Schritt abgerufen werden können, wenn die Inhalte eines Warteschlangenziels durchsucht werden. Beachten Sie, dass die Clientanwendung immer alle Nachrichten aus der Warteschlange abruft. Über das Attribut `imqQueueBrowserMaxMessagesPerRetrieve` wird festgelegt, wie die in der Warteschlange enthaltenen Nachrichten aufgeteilt werden, die an die Client-Runtime gesendet werden müssen (einige große Chunks oder viele kleine Chunks), die Gesamtzahl der Nachrichten

bleibt jedoch gleich. Eine Änderung des Attributwerts kann sich auf die Leistung auswirken, führt jedoch nicht dazu, dass die Clientanwendung mehr oder weniger Daten abruft (*Fehlernummer 6387631*).

Broker-Probleme

Die nachfolgend beschriebenen Probleme beziehen sich auf den Message Queue-Broker.

- Es war bisher unklar, wie der Broker für die Round-Robin-Übermittlung konfiguriert werden kann. Die Lösung ist einfach und konfigurierbar.

  1. Setzen Sie das Zielattribut `maxNumActiveConsumers` auf -1. Dadurch wird die Round-Robin-Übermittlung aktiviert.
  2. Setzen Sie das Zielattribut `consumerFlowLimit` auf 1. Dadurch wird die Anzahl an Nachrichten festgelegt, die zu einem einzelnen Konsumenten gesendet werden, bevor mit dem Senden an den nächsten Konsumenten fortgefahren wird. Für eine andere Aufteilung, setzen Sie dieses Attribut auf den gewünschten Wert. Per Voreinstellung werden 100 Nachrichten an jeden Konsumenten gesendet.

- Wenn der Persistenzspeicher zu viele Zielstandorte öffnet, kann auf den Broker nicht mehr zugegriffen werden (*Fehlernummer 4953354*).

  Umgehung Diese Bedingung wird vom Broker verursacht, der das Deskriptor-Limit für die offenen Dateien im System erreicht. Unter Solaris und Linux erhöhen Sie das Dateideskriptor-Limit mit dem Befehl `ulimit`.

- Konsumenten verwaisen, wenn ein Zielstandort gelöscht wird (*Fehlernummer 5060787*).

  Aktive Konsumenten verwaisen, wenn ein Zielstandort gelöscht wird. Ein verwaister Konsument erhält keine Meldungen mehr (auch dann nicht, wenn der Zielstandort neu erstellt wird).

  Umgebung Derzeit gibt es keine Umgehung für dieses Problem.

Broker-Cluster

Die folgenden Punkte beziehen sich auf die Verwendung von Broker-Clustern.

- In dieser Version werden lediglich vollständig verbundene Broker-Cluster unterstützt. Dies bedeutet, dass jeder Broker in einem Cluster direkt mit allen anderen Brokern im Cluster kommunizieren muss. Wenn Sie Broker mithilfe des Befehlszeilenarguments `imqbrokerd -cluster` verbinden, stellen Sie sicher, dass alle Broker im Cluster enthalten sind.
- Ein Broker kann bei Verwendung von HADB ausschließlich Nachrichten bis zu einer Größe von 10 MB verarbeiten. *(Fehler 6531734)*
- Wenn ein Client mit einem Hochverfügbarkeits-Broker verbunden ist, versucht die Client-Laufzeit solange die Verbindung wiederherzustellen, bis sie erfolgreich war (unabhängig davon, auf welchen Wert `imqAddressListIterations` gesetzt ist).

- Ein Client, der mit einem Broker verbunden ist, der wiederum Teil eines Clusters ist, kann `QueueBrowser` nicht zum Durchsuchen von Warteschlangen nutzen, die sich auf Remote-Brokern in diesem Cluster befinden. Der Client kann nur die Warteschlangeninhalte durchsuchen, die sich auf dem Broker befinden, mit dem er direkt verbunden ist. Der Client sendet eventuell noch Meldungen an eine beliebige Warteschlange oder erhält Meldungen von einer Warteschlange oder einem Broker im Cluster. Die Einschränkung betrifft nur das Durchsuchen.
- Wenn Sie in einem konventionellen Cluster einen 4.1-Broker mit einem 3.x-Broker clustern möchten, müssen Sie die Eigenschaft `imq.autocreate.queue.maxNumActiveConsumers=1` für den 4.1-Broker setzen. Anderenfalls werden die Broker keine Clusterverbindung herstellen können.
- Beim Konvertieren in einen Hochverfügbarkeits-Cluster können Sie das Message Queue-Manger-Dienstprogramm (imqdbmgr) verwenden, um einen vorhandenen eigenständigen persistenten HADB-Datenspeicher in einen gemeinsam genutzten HADB-Speicher zu konvertieren. Der Befehl lautet wie folgt.

  ```
  imqdbmgr upgrade hastore
  ```

  Dieses Dienstprogramm können Sie in den folgenden Fällen verwenden.

  - Für den Wechsel von einem eigenständigen HADB-Speichers der Version 4.0 zu einen gemeinsam genutzten HADB-Speicher der Version 4.1. In diesem Fall wird der Broker den Speicher automatisch aktualisieren. Anschließend können Sie den `imqdbmgr`-Befehl ausführen, um den aktualisierten Datenspeicher für die gemeinsame Nutzung zu konvertieren.
  - Für den Wechsel von einem eigenständigen HADB-Speichers der Version 4.1 zu einen gemeinsam genutzten HADB-Speicher. In diesem Fall müssen Sie lediglich den oben gezeigten `imqdbmgr`-Befehl ausführen, um den Datenspeicher für die gemeinsame Nutzung zu konvertieren.

  Da dieser Befehl lediglich die Umwandlung von HADB-Speichern unterstützt, ist es nicht möglich, damit dateibasierte Speicher oder andere JDBC-Speicher in einen gemeinsam genutzten HADB-Speicher zu konvertieren. Wenn Sie vorher eine 3.x-Version von Message Queue ausgeführt haben, müssen Sie einen HADB-Speicher erstellen und anschließend die Daten manuell zu diesem Speicher migrieren, um die Hochverfügbarkeits-Funktion zu nutzen.
- Die Umwandlung in einen HADB-Speicher mit dem Befehl `imqdbmgr upgrade hastore` kann mit der Fehlermeldung "zu viele Sperren gesetzt" fehlschlagen, wenn der Speicher mehr als 10.000 Nachrichten enthält. (*Fehlernummer 6588856)*).

  (*Umgehung*) Verwenden Sie den folgenden Befehl, um die Anzahl an Sperren zu erhöhen.

  ```
  hadbm set NumberOfLocks=<gewünschte_Anzahl>
  ```

  Weitere Informationen finden Sie unter "HADB Problems" im *Sun Java System Application Server 9.1 Enterprise Edition Troubleshooting Guide*.

- Wenn mehr als 500 Remote-Nachrichten in einer Transaktion übermittelt werden, gibt der Broker möglicherweise den Fehler "HADB-E-12815: Kein Tabellen-Arbeitsspeicher mehr verfügbar." aus. *(Fehlernummer 6550483)*

  Weitere Informationen finden Sie unter "HADB Problems" im *Sun Java System Application Server 9.1 Enterprise Edition Troubleshooting Guide*.

- In einem Broker-Cluster fügt ein Broker Nachrichten zu einer Warteschlange für eine Remote-Verbindung hinzu, die noch nicht hergerstellt wurde (*Fehlernummer 4951010*).

  Umgehung Die Meldungen werden vom Konsumenten empfangen, sobald die Verbindung hergestellt ist. Die Nachrichten werden an einen anderen Konsument gesendet, wenn die Verbindung beendet wird.

- Wenn ein Konsument mehr als eine Nachricht von einem Remote-Broker in einer Transaktion empfängt, ist es möglich, dass die folgende Fehlermeldung für den Broker protokolliert wird. Diese Fehlermeldung ist nicht kritisch und kann ignoriert werden:

```
[26/Jul/2007:13:18:27 PDT] WARNING [B2117]:
Nachrichtenbestätigung fehlgeschlagen aus
mq://129.145.130.95:7677/?instName=a&brokerSessionUID=3209681167602264320:
    ackStatus = NOT_FOUND(404)\
    Ursache = Aktualisieren von Remote-Transaktionsstatus auf COMMITED(6):
Transaktion 3534784765719091968 nicht gefunden, die Transaktion
wurde möglicherweise bereits übermittelt.
    AckType = MSG_CONSUMED
    MessageBrokerSession = 3209681167602264320
    TransactionID = 3534784765719091968
        SysMessageID = 8-129.145.130.95(95:fd:93:91:ec:a0)-33220-1185481094690
        ConsumerUID = 353478476571913395 2\par

[26/Jul/2007:13:18:27 PDT] WARNING Benachrichtigung zur Übermittlung von Transaktion
[8-129.145.130.95(95:fd:93:91:ec:a0)-33220-1185481094690,
[consumer:3534784765719133952, type=NONE]]
TUID=3534784765719091968 erhielt Antwort:
com.sun.messaging.jmq.jmsserver.util.BrokerException:
 Aktualisieren von Remote-Transaktionsstatus auf COMMITED(6):
  Transaktion 3534784765719091968 nicht gefunden, die Transaktion wurde möglicherweise
  bereits übermittelt:
com.sun.messaging.jmq.jmsserver.util.BrokerException: Aktualisieren von Remote-Transaktionsstatus
auf COMMITED(6): Transaktion 3534784765719091968 nicht gefunden, die Transaktion wurde möglicherweise
 bereits übermittelt.
```

Diese Meldung wird im Protokoll erfasst, wenn der Nachrichten-Startbroker für spätere Nachrichten in der Transaktion über die Übermittlung benachrichtigt wurde, sobald die `imq.txn.reapLimit`-Eigenschaft im Vergleich zur Anzahl an Remote-Nachrichten in einer Transaktion gering ist. *(Fehler 6585449)*

Umgehung Um diese Meldung zu verhindern, erhöhen Sie den Wert der `imq.txn.reapLimit`-Eigenschaft.

JMX-Probleme

Auf der Windows-Plattform gibt die `getTransactionInfo`-Methode der Überwachungs-MBean für den Transaktionsmanager Transaktionsinformationen zurück, die eine falsche Transaktionserstellungszeit enthalten (*Fehlernummer 6393359*).

Umgehung Verwenden Sie stattdessen die `getTransactionInfoByID`-Methode der Überwachungs-MBean für den Transaktionsmanager.

Unterstützung für SOAP

Sie sollten zwei Probleme in Verbindung mit der SOAP-Unterstützung beachten.

- Seit der Veröffentlichung von Message Queue Version 4.0 werden SOAP-verwaltete Objekte nicht mehr unterstützt.
- Die SOAP-Entwicklung hängt von mehreren Dateien ab: `SUNWjaf`, `SUNWjmail`, `SUNWxsrt` und `SUNWjaxp`. In Message Queue Version 4.1 sind diese Dateien nur verfügbar, wenn Sie Message Queue mit JDK Version 1.6.0 oder höher ausführen.

Dateien für die Neuverteilung

Sun Java System Message Queue 4.1 enthält die folgenden Dateisätze, die Sie verwenden und im Binärformat verteilen können:

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

Außerdem können Sie die Dateien `LICENSE` und `COPYRIGHT` ebenfalls neu verteilen.

Zugriffsfunktionen für Personen mit Behinderungen

Um Eingabehilfen zu erhalten, die seit der Veröffentlichung dieses Dokuments auf den Markt gekommen sind, lesen Sie Abschnitt 508 der Produktbewertungen (bei Sun auf Anfrage erhältlich), um die für Sie geeignete Version zu ermitteln. Die aktualisierten Versionen von Anwendungen finden Sie unter
`http://sun.com/software/javaenterprisesystem/get.html`.

Informationen zum Einsatz von Sun für Eingabehilfen erhalten Sie unter
`http://sun.com/access`.

Problemmeldungen und Feedback

Wenn Sie mit Sun Java System Message Queue Probleme haben, wenden Sie sich an die Kundenunterstützung von Sun. Dazu stehen Ihnen folgende Möglichkeiten zur Verfügung:

- Online-Softwaresupport von Sun unter `http://www.sun.com/service/sunone/software`.

  Diese Site bietet Links zur Knowledge Base, zum Online Support Center und ProductTracker sowie zu Wartungsprogrammen und Supportkontaktnummern.
- Wenden Sie sich per Telefon an Sun. Verwenden Sie hierzu die auf Ihrem Wartungsvertrag angegebene Telefonnummer.

Damit wir Ihnen unmittelbar Hilfe anbieten können, halten Sie die folgenden Informationen bereit, wenn Sie sich an den Support wenden:

- Beschreibung des Problems, einschließlich der Situation, in der das Problem auftrat, sowie seine Auswirkungen auf Ihre Arbeit.
- Computertyp, Betriebssystem- und Produktversion, u. a. Patches und andere Softwareanwendungen, die das Problem verursacht haben könnten.
- Detaillierte Schritte zu den von Ihnen verwendeten Methoden, um das Problem zu reproduzieren.
- Sämtliche Fehlerprotokolle oder Kernspeicherauszüge.

Sun Java System-Software-Forum

Unter der folgenden Adresse steht ein Sun Java System Message Queue-Forum zur Verfügung:

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

Wir freuen uns über Ihre Teilnahme.

Java Technology Forum

Im Java Technology Forum finden Sie möglicherweise ein für Sie interessantes JMS-Forum.

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

Sun freut sich über Ihre Kommentare

Sun ist stets an einer Verbesserung der eigenen Dokumentation interessiert und nimmt Ihre Kommentare und Anregungen gerne entgegen.

Sie können Ihre Kommentare unter `http://docs.sun.com` durch Klicken auf den entsprechenden Link an uns senden. Geben Sie im Online-Formular den Dokumenttitel und die Teilenummer an. Die Teilenummer ist eine 7- oder 9-stellige Zahl, die Sie auf der Titelseite des Handbuchs oder am Anfang des Dokuments finden. Der Titel des vorliegenden Buches lautet beispielsweise Versionshinweise zu Sun Java System Message Queue 4.1, die Teilenummer lautet 820-3188-10.

Weitere Quellen von Sun

Nützliche Informationen über Sun Java System finden Sie unter den folgenden Internetadressen:

- Dokumentation

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

  `http://www.sun.com/service/sunps/sunone`
- Softwareprodukte und Services

  `http://www.sun.com/software`
- Softwaresupport-Services

  `http://www.sun.com/service/sunone/software`
- Support und Knowledge Base

  `http://www.sun.com/service/support/software`
- Sun-Support und -Schulungen

  `http://training.sun.com`
- Beratung und Profi-Services

  `http://www.sun.com/service/sunps/sunone`
- Informationen für Entwickler

  `http://developers.sun.com`
- Sun-Support-Services für Entwickler

  `http://www.sun.com/developers/support`
- Softwareschulungen

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