Notas de la versión de Sun Java System Message Queue 4.1

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

Referencia: 820–3189–10
septiembre 2007

071001@18741

Contenido

◆ ◆ ◆ CAPÍTULO 1

Notas de la versión de Sun Java System Message Queue 4.1

Versión 4.1

Número de referencia 820-3189-10

Estas notas de la versión contienen información importante que está disponible en el momento del lanzamiento de Sun Java™ System Message Queue 4.1. Aquí se tratan nuevas funciones y mejoras, limitaciones y problemas conocidos e información de otro tipo. Lea este documento antes de empezar a utilizar Message Queue. Estas notas de la versión también contienen información sobre la versión 4.0 de Message Queue; consulte "Acerca de Message Queue 4.0" en la página 17 para conseguir información sobre las nuevas funciones de esta versión.

La versión más actualizada de estas notas de la versión se encuentra en el sitio web de documentación de Sun Java System Message Queue Consulte el sitio Web antes de instalar y configurar el software y, después, visítelo de forma periódica para ver los manuales y las notas de la versión más actualizados.

En estas notas de la versión se incluyen los siguientes apartados:

Se hace referencia a las direcciones URL de terceras partes para proporcionar información adicional relacionada.

Sun no se responsabiliza de la disponibilidad de las sedes Web de otras empresas que se mencionan en este documento. Sun no garantiza ni se hace responsable de los contenidos, la publicidad, los productos u otros materiales que puedan estar disponibles a través de dichos sitios o recursos. Sun no será responsable de daños o pérdidas, supuestos o reales, provocados por o a través del uso o confianza del contenido, bienes o servicios disponibles en dichos sitios o recursos, o a través de ellos.

Historial de revisiones de las notas de la versión

La siguiente tabla contiene una lista de las fechas de todas las versiones 4.x del producto Message Queue y describe los principales cambios realizados en cada versión.

TABLA 1–1 Historial de revisiones

Fecha	Descripción de los cambios
Mayo de 2006	Versión inicial de este documento para la versión 4.0 de Message Queue.
Enero de 2007	Versión inicial de este documento para la versión 4.1 beta de Message Queue. Incorpora la descripción de la compatibilidad de JAAS.
Abril de 2007	Segunda versión de este documento para la versión 4.1 beta de Message Queue. Incorpora la función de alta disponibilidad.
Septiembre de 2007	Tercer versión de este documento para enviarla al cliente. Incorpora la descripción de la compatibilidad con Java Enterprise System Monitoring Framework, los puertos C fijos, las soluciones a fallos y otras funciones.

Acerca de Message Queue 4.1

Sun Java System Message Queue es un servicio de mensajería completo que proporciona funciones fiables y asíncronas, conformes con la especificación Java Messaging Specification (JMS) 1.1. Además, Message Queue incluye funciones que superan la especificación JMS para dar satisfacción a las necesidades de las instalaciones en grandes empresas.

La versión 4.1 de Message Queue incorpora compatibilidad con Java Authentication y Authorization Service (JAAS), para el uso de los puertos C fijos y para Java Enterprise System Monitoring Framework. Incorpora algunas mejoras menores y soluciones a errores. Esta sección incluye la siguiente información.

- "Novedades de la versión 4.1" en la página 7

- "Requisitos de hardware y software" en la página 17

Para conseguir información sobre las funciones nuevas de Message Queue 4.0, consulte "Acerca de Message Queue 4.0" en la página 17.

Novedades de la versión 4.1

Message Queue 4.1 incorpora los clúster de agente de alta disponibilidad (disponibilidad de datos y servicios), compatibilidad con JAAS y otras funciones menores. Esta sección describe estas funciones y proporciona más referencias que pueden serle útiles.

- "Alta disponibilidad" en la página 7
- "Compatibilidad con JAAS" en la página 8
- "Cambio del formato del almacén persistente" en la página 14
- "Configuración del agente" en la página 15
- "Compatibilidad con JES Monitoring Framework " en la página 15
- "Gestión de transacciones" en la página 16
- "Puertos fijos para las conexiones de los clientes C" en la página 16

Alta disponibilidad

Message Queue 4.1 presenta los clústeres de alta disponibilidad, que proporcionan disponibilidad de los datos y de servicio. Si un cliente pierde su conexión con un agente de alta disponibilidad, se le vuelve a conectar automáticamente con otro agente de un clúster. El agente que proporciona la nueva conexión se hace cargo de los datos persistentes y del estado del agente desconectado y continúa proporcionando un servicio ininterrumpido a los clientes de este agente. Puede ejecutar los agentes de alta disponibilidad en una conexión segura.

Los agentes de alta disponibilidad requieren el uso de una base de datos también de alta disponibilidad (HADB) Si usted no dispone de esta base de datos o si la disponibilidad de los datos no es importante para usted, puede continuar utilizando clústeres convencionales, que ofrecen una reconexión automática y disponibilidad de servicio.

Configurar agentes de alta disponibilidad es un proceso sencillo: puede especificar los siguientes tipos de propiedades para cada agente del clúster.

- *Propiedades de pertenencia a clúster,* que sirven para especificar si el agente forma parte de un clúster de alta disponibilidad, el ID del clúster y del agente.
- *Propiedades de la base de datos de alta disponibilidad (HADB)*, que sirven para especificar el modelo de los mensajes persistentes (JDBC), el nombre del fabricante HADB y las propiedades de configuración específicas del fabricante de la base de datos.
- *Propiedades de detección de fallos y toma de control*, que sirven para especificar cómo deberían detectarse y gestionarse los fallos de los agentes.

Para usar esta función, debe hacer lo siguiente:

1. Instalar una base de datos de alta disponibilidad.
2. Instalar el archivo .jar del controlador de JDBC.
3. Crear el esquema de la base de datos para el almacén persistente de alta disponibilidad.
4. Configurar aquellas propiedades que estén relacionadas con la alta disponibilidad de cada agente del clúster.
5. Iniciar cada uno de los agentes del clúster.

Si desea consultar una explicación del concepto de alta disponibilidad y una comparación con los clústeres convencionales, consulte el Capítulo 4, "Broker Clusters" de *Sun Java System Message Queue 4.1 Technical Overview*. Si desea información sobre procedimientos y referencias de la alta disponibilidad, consulte el Capítulo 8, "Broker Clusters" de *Sun Java System Message Queue 4.1 Administration Guide* y "Cluster Configuration Properties" de *Sun Java System Message Queue 4.1 Administration Guide*.

Si usa una base de datos HADB con Message Queue versión 4.0 y quiere utilizar un clúster de alta disponibilidad, puede usar la utilidad `dbmgr` para actualizarse a un almacén compartido de HADB. Para más información, consulte "Clústeres de agente" en la página 45.

Compatibilidad con JAAS

Además de los mecanismos de autenticación basados en archivos y en LDAP que integra, Message Queue también es compatible con Java Authentication y Authorization Service (JAAS), que le permite conectar distintos servicios al agente para autenticar los clientes de Message Queue. Esta sección describe la información que el agente pone a disposición de un servicio de autenticación compatible con JAAS, y explica cómo configurar el agente para utilizar este servicio.

No describiremos el JAAS API, por exceder el propósito de este documento. Consulte las siguientes fuentes si necesita más información.

- Para conseguir toda la información sobre el JAAS API, consulte la *Guía de referencia de Java Authentication and Authorization Service (JAAS)*.

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

- Para conseguir información sobre cómo escribir un LoginModule, consulte la *Guía del desarrollador del LoginModule de Java Authentication and Authorization Service (JAAS)*.

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

El JAAS API es un API nuclear de J2SE y por ello forma parte del entorno de tiempo de ejecución de Message Queue. JAAS define la capa de abstracción entre una aplicación y un mecanismo de autenticación, permitiendo conectar el mecanismo deseado sin cambiar el código de aplicación. En el caso del servicio Message Queue, la capa de abstracción se encuentra entre el agente (aplicación) y un proveedor de autenticación. Configurando unas cuantas propiedades de agente, es posible conectar cualquier servicio de autenticación compatible con JAAS y actualizarlo sin interrumpir ni alterar el código del agente.

Si utiliza la autenticación basada en JAAS, puede utilizar clientes JMX para gestionar el agente, pero deberá configurar manualmente la compatibilidad con JAAS (configurando las propiedades de agente relacionadas con JAAS) antes de iniciar el agente. No es posible utilizar el JMX API para cambiar estas propiedades.

Elementos de JAAS

La Figura 1–1 muestra los elementos básicos de JAAS: un cliente JAAS, un servicio de autenticación compatible con JAAS y un archivo de configuración de JAAS.

- El cliente JAAS es una aplicación que busca realizar la autenticación con un servicio compatible con JAAS. Se comunica con este servicio mediante un LoginModule y se encarga de proporcionar un controlador de devolución de llamadas que puede utilizar el LoginModule para obtener el nombre del usuario, su contraseña y otra información importante.
- El servicio de autenticación compatible con JAAS consiste en uno o varios LoginModule y en procesos lógicos que realizan la autenticación necesaria. El LoginModule puede incluir la lógica de autenticación o puede utilizar un protocolo privado o API para comunicarse con un módulo que proporcione el proceso lógico.
- El archivo de configuración de JAAS es un archivo de texto que utiliza el cliente JAAS para encontrar los LoginModules necesarios para comunicarse con el servicio compatible con JAAS.

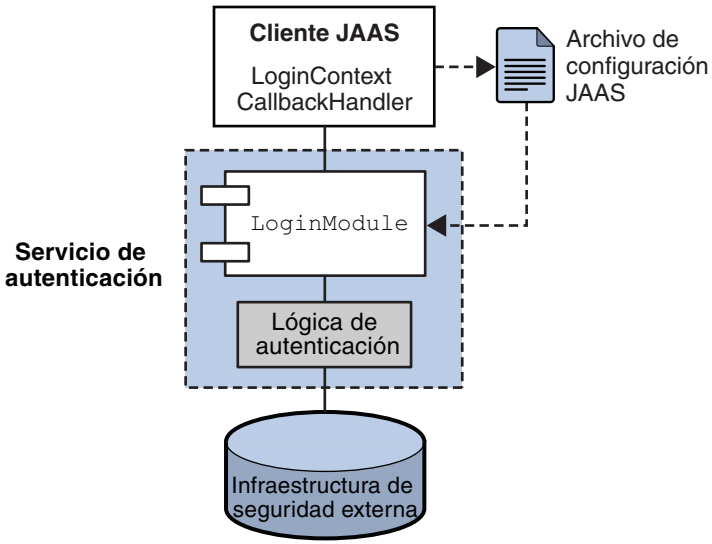

FIGURA 1–1 Elementos JAAS

La siguiente sección explica cómo el servicio Message Queue utiliza estos elementos para proporcionar la autenticación compatible con JAAS.

JAAS y Message Queue

La siguiente figura muestra cómo el agente de Message Queue utiliza JAAS. Muestra una implementación más compleja del modelo JAAS mostrado en la figura anterior.

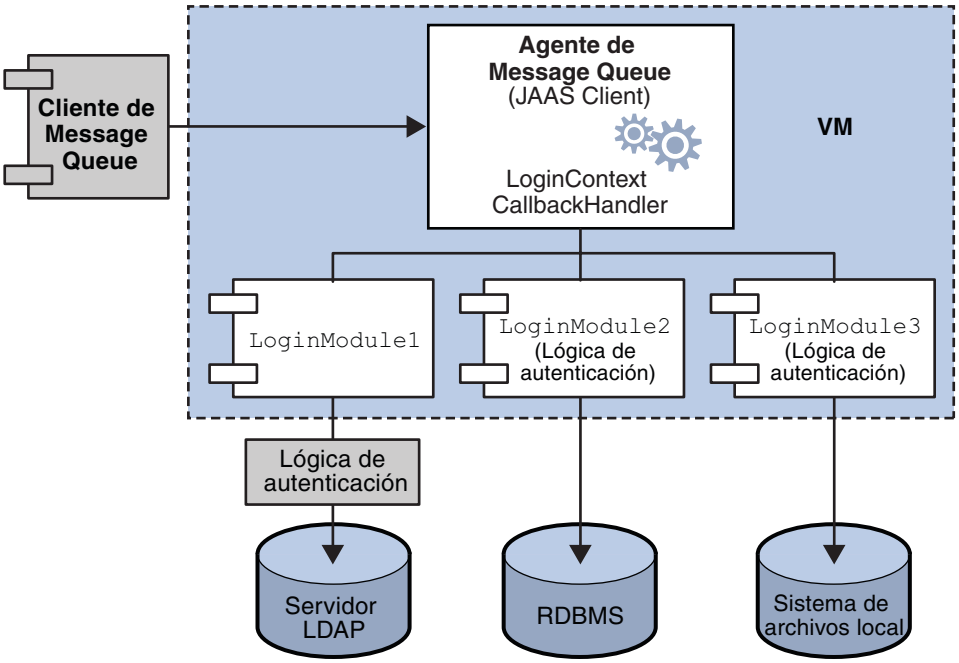

FIGURA 1–2 Cómo utiliza JAAS Message Queue

Tal y como se indicaba en el caso más simple, la capa del servicio de autenticación es independiente del agente. El servicio de autenticación consiste en uno o más módulos de registro (LoginModule) y en los módulos de autenticación adicionales que sean necesarios. Los módulos de registro se ejecutan en la misma máquina virtual Java que el agente. El agente de Message Queue se representa en el módulo de registro como un `LogInContext` y se comunica con él mediante un `CallBackHandler` que forma parte del código de tiempo de ejecución del agente.

El servicio de autenticación también proporciona un archivo de configuración JAAS que contiene entradas a los módulos de registro. El archivo de autenticación especifica el orden en que deben utilizarse los módulos y algunas condiciones para su uso. Al iniciarse el agente, JAAS localiza el archivo de configuración mediante la propiedad del sistema Java `java.security.auth.login.config` o mediante el archivo de propiedades de seguridad de Java. Después selecciona una entrada en el archivo de configuración JAAS, según el valor de la propiedad del agente `imq.user_repository.jaas.name` . Esa entrada especifica cuáles serán los módulos de registro que se utilizarán para la autenticación. Como muestra la figura, el

agente puede utilizar más de un módulo de registro. (La relación entre el archivo de configuración, el módulo de registro y el agente se muestra en la Figura 1–3.)

El hecho de que el agente utilice un servicio de autenticación complemento de JAAS permanece totalmente transparente para el cliente de Message Queue. El cliente continúa conectándose al agente como lo hacía antes, mediante un nombre de usuario y una contraseña. El agente, por su parte, utiliza un controlador de devolución de llamadas para pasar esta información al servicio de autenticación, el cual utiliza esta información para autenticar al usuario y devolver los resultados. Si la autenticación tiene éxito, el agente permitirá la conexión; pero si falla, el tiempo de ejecución del cliente devolverá una excepción de seguridad de JMS que el cliente deberá solucionar.

Una vez autenticado el cliente de Message Queue (y siempre que haya que hacer más tareas de autorización), al agente procede normalmente: consulta el archivo de control de acceso para determinar si el cliente autenticado tiene autorización para realizar las acciones a su cargo: acceder a un destino, consumir un mensaje, examinar una cola, etc.

Configurar la autenticación compatible con JAAS

Para configurar la autenticación compatible con JAAS, es necesario establecer las propiedades del agente y del sistema para seleccionar este tipo de autenticación, especificar la ubicación del archivo de configuración y especificar las entradas a los módulos de registro que van a utilizarse.

Esta sección ilustra la relación existente entre el cliente JAAS, los módulos de registro y el archivo de configuración de JAAS y describe el proceso necesario para configurar la autenticación compatible con JAAS. La siguiente figura muestra la relación entre el archivo de configuracion, el módulo de registro y el agente.

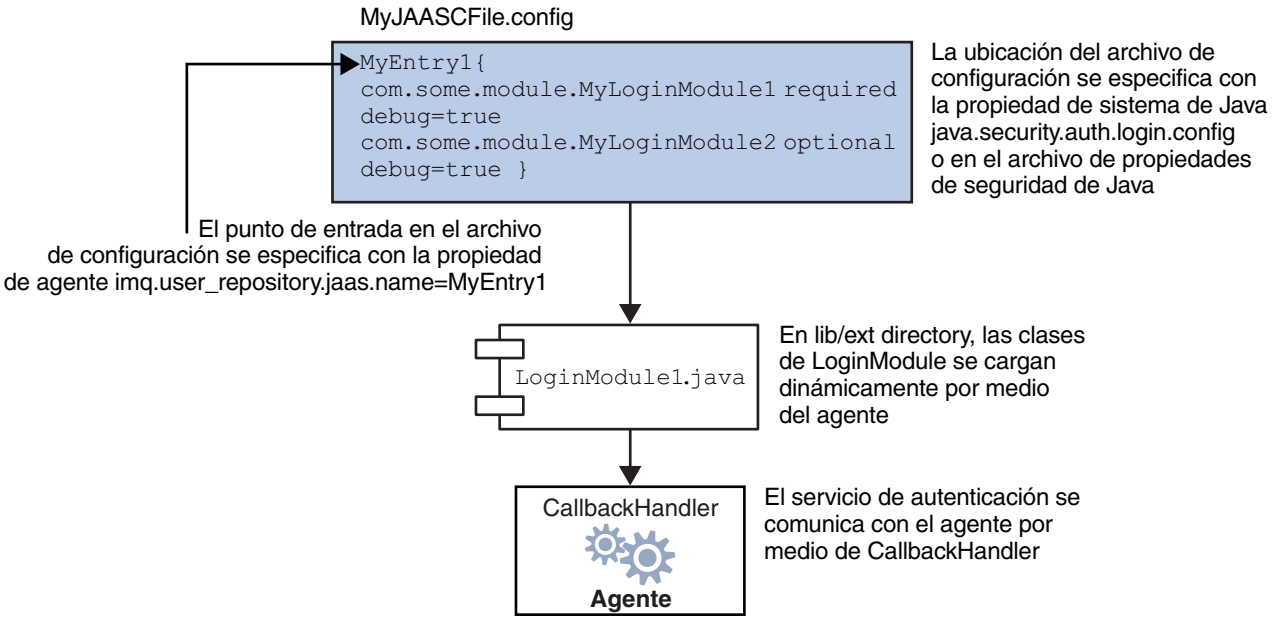

FIGURA 1–3 Configurar la compatibilidad con JAAS

Como muestra la figura, el archivo de configuración de JAAS, `MyJAASCFile.config` contiene referencias a varios módulos de registro, agrupados en un punto de entrada. Para localizar el archivo de configuración, el agente consulta la propiedad de sistema de Java `java.security.auth.login.config` o el archivo de propiedades de seguridad de Java. Para determinar los módulos de registro que van a utilizarse, se consulta la propiedad del agente `imq.user_repository.jaas.name` , que especifica la entrada deseada en el archivo de configuración. Las clases de esos módulos se encuentran en el directorio `lib/ext`.

Para configurar la compatibilidad de JAAS para Message Queue, debe seguir estos pasos: (En un entorno de desarrollo, el encargado de completar estos pasos sería el desarrollador. En un entorno de producción, el administrador podría encargarse de algunas de estas tareas.)

1. Cree una o varias clases del módulo de registro que implementen el servicio de autenticación. A continuación, se enumeran los tipos de devoluciones de llamada JAAS que admite el agente.

   `javax.security.auth.callback.LanguageCallback`
   El agente utiliza esta devolución de llamada para pasar al servicio de autenticación la configuración local en que el agente se está ejecutando. Este valor puede utilizarse para la localización.

`javax.security.auth.callback.NameCallback`
El agente utiliza esta devolución de llamada para pasar al servicio de autenticación el nombre del usuario especificado por el cliente de Message Queue cuando se solicita la conexión.

`javax.security.auth.callback.TextInputCallback`
El agente utiliza esta devolución de llamada para especificar el valor de `imq.authentication.type` al servicio de autenticación cuando el `TextInputCallback.getPrompt()` sea `imq.authentication.type`. En este momento, el único valor posible para este campo es `basic`, que indica que la codificación de contraseña es Base-64.

`javax.security.auth.callback.PasswordCallback`
El agente utiliza esta devolución de llamada para pasar al servicio de autenticación la contraseña especificada por el cliente de Message Queue cuando se solicita la conexión.

`javax.security.auth.callback.TextOutputCallback`
El agente utiliza esta devolución de llamada para proporcionar acceso al servicio de autenticación registrando la salida del texto en el archivo de registro del agente. Los tipos de mensajes de devolución de llamada `ERROR`, `INFORMATION`, `WARNING` se asignan a los niveles de registro del agente `ERROR`, `INFO`, y `WARNING` respectivamente.

2. Cree un archivo de configuración JAAS con entradas que hagan referencia a las clases del módulo de registro y especifique la ubicación de este archivo en el administrador de Message Queue. (El archivo puede localizarse de forma remota, y es posible especificar su ubicación con una URL.)
3. Tome nota del nombre de la entrada (que hace referencia a las clases de implementación de registro) en el archivo de configuración JAAS.
4. Archive las clases que implementan los módulos de registro en un archivo jar y coloque este archivo en el directorio de Message Queue `lib/ext`.
5. Configure las propiedades de agente relacionadas con la compatibilidad con JAAS. Éstas se describen en la Tabla 1–2.
6. Configure la siguiente propiedad del sistema para especificar la ubicación del archivo de configuración JAAS.

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

   Por ejemplo, puede especificar el archivo de configuración al iniciar el agente.

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

   Hay otras formas de especificar la ubicación del archivo de configuración JAAS. Para más información, consulte:

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

La siguiente tabla enumera las propiedades de agente necesarias para configurar la compatibilidad con JAAS.

TABLA 1–2 Propiedades de agente para la compatibilidad con JAAS

Propiedad	Descripción
`imq.authentication.type`	Establézcala en `basic` para indicar que la codificación de contraseña es Base-64. Este es el único valor que se permite para la autenticación JAAS.
`imq.authentication.basic.user_repository`	Establézcala en `jaas` para especificar la autenticación JAAS.
`imq.accesscontrol.type`	Establézcala en `file`.
`imq.user_repository.jaas.name`	Indique el nombre de la entrada deseada (en el archivo de configuración JAAS) que haga referencia a los módulos de acceso que desea utilizar como mecanismo autenticación. Es el nombre que anotó en el paso 3.
`imq.user_repository.jaas.userPrincipalClass`	Esta propiedad, que utiliza el control de acceso de Message Queue, especifica la clase de implementación `java.security.Principal` de los módulos de acceso que el agente utiliza para extraer el nombre principal que representa la entidad del usuario en el archivo de control de acceso de Message Queue. Si no se especifica, se utilizará en su lugar el nombre de usuario que facilitó el cliente de Message Queue al solicitar la conexión.
`imq.user_repository.jaas.groupPrincipalClass`	Esta propiedad, que utiliza el control de acceso de Message Queue, especifica la clase de implementación `java.security.Principal` de los módulos de acceso que el agente utiliza para extraer el nombre principal que representa la entidad del grupo en el archivo de control de acceso de Message Queue. Si no se especifica, se pasarán por alto las reglas de grupo que pudiera haber en el archivo de control de acceso de Message Queue.

Cambio del formato del almacén persistente

La versión 4.1 de Message Queue ha cambiado el almacén JDBC para admitir la alta disponibilidad. Por esta razón, la versión de almacén de JDBC se ha aumentado a la 410. Las versiones de almacén JDBC 350, 370 y 400 cambiarán automáticamente al formato de la versión 410.

Tenga en cuenta que la versión del almacén persistente basado en archivos sigue siendo la 370 porque no se ha efectuado ningún cambio en él.

Configuración del agente

La propiedad `IMQ_DEFAULT_EXT_JARS` se ha añadido al archivo `imqenv.conf`. Puede configurar esta propiedad para especificar el nombre de las rutas de los archivos .jar que deben incluirse en `CLASSPATH` al iniciarse el agente. Si utiliza esta propiedad para especificar la ubicación de los archivos `.jar`, ya no necesitará copiar estos archivos en el directorio `lib/ext`. Los jars externos pueden hacer referencia a los controladores de JDBC o a módulos de registro JAAS. El siguiente comando de ejemplo especifica la ubicación de los controladores jdbc.

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

Compatibilidad con JES Monitoring Framework

Message Queue es es compatible con Sun Java Enterprise System (JES) Monitoring Framework, que permite controlar los componentes de Java Enterprise System con una interfaz gráfica normal. La interfaz se implementa mediante una consola basada en web llamada "Sun Java System Monitoring Console". Si ejecuta Message Queue junto con otros componentes de JES, le resultará más cómodo gestionar todos los componentes con una única interfaz.

La estructura de control JES define el modelo de datos común (CMM) que utilizarán todos los productos de componentes JES. Este modelo ofrece una vista centralizada y uniforme de todos los componentes JES. Message Queue expone los siguientes objetos a la estructura de supervisión JES:

- el producto instalado
- el nombre de la instancia del agente
- el asignador de puertos del agente
- cada uno de los servicios de conexión
- cada uno de los destinos físicos
- el almacén persistente
- el depósito de usuarios

Cada uno de estos objetos se asigna a un objeto CMM cuyos atributos pueden controlarse desde la consola de supervisión de JES. En el tiempo de ejecución, los administradores pueden utilizar la consola para ver las estadísticas de rendimiento, crear reglas de supervisión automáticas y atender alarmas. Si necesita información más detallada sobre la asignación de objetos de Message Queue a objetos CMM, consulte la *Guía de supervisión de Sun Java Enterprise System*.

Para activar la supervisión JES, siga estos pasos:

1. Instale y configure todos los componentes de su implementación (Message Queue y otros componentes) siguiendo las instrucciones de la *Guía de instalación de Sun Java Enterprise System*.
2. Active y configure Monitoring Framework en todos los componentes supervisados, tal y como se describe en la *Guía de supervisión de Sun Java Enterprise System*.

3. Instale la consola de supervisión en un host aparte, inicie el agente maestro y después inicie el servidor web, tal y como se describe en la *Guía de supervisión de Sun Java Enterprise System*.

Si utiliza JES Monitoring Framework, el rendimiento del agente no se verá afectado ya que todo el trabajo de recopilación de datos corre a cargo de la estructura de supervisión, que extrae los datos de la infraestructura actual de gestión de datos del agente.

Gestión de transacciones

Anteriormente, los administradores sólo podían deshacer transacciones que estuvieran en estado `PREPARED` Es decir, si una sesión que formara parte de una transacción distribuida no terminaba correctamente, la transacción permanecía en un estado que era imposible reorganizar para el administrador del agente. En la versión 4.1 de Message Queue, es posible utilizar la utilidad imqcmd para reorganizar (deshacer) las transacciones que se encuentren en los siguientes estados: `STARTED`, `FAILED`, `INCOMPLETE`, `COMPLETE`, `PREPARED`.

Para ayudarle a determinar si es posible deshacer una transacción particular (sobre todo si no se encuentra en estado `PREPARED`), la utilidad imqcmd proporciona datos adicionales como parte de la salida `imqcmd query txn`: proporciona el ID de la conexión que inició la transacción y especifica la hora en la que se creó la transacción. Basándose en esta información, el administrador puede decidir si es preciso deshacer o no la transacción. Por lo general, el administrador debe evitar deshacer una transacción prematuramente.

Puertos fijos para las conexiones de los clientes C

Los clientes C pueden utilizar la propiedad de conexión `MQ_SERVICE_PORT_PROPERTY` para especificar el puerto fijo al que conectarse. Esto puede ser útil si está intentando atravesar un cortafuegos o si necesita evitar el servicio de asignación de puertos del agente (que asigna los puertos dinámicamente).

Recuerde que también debe configurar el puerto de servicio JMS en el lado del agente. Por ejemplo, si desea conectar a su cliente con el puerto 1756 mediante `ssljms`, tendría que hacer lo siguiente:

- En el lado del cliente: Establezca la propiedad `MQ_SERVICE_PORT_PROPERTY` en 1756 y `MQ_CONNECTION_TYPE_PROPERTY` en `SSL`.
- En el lado del agente: Establezca la propiedad `imq.`*serviceNameType.protocol* `.port` en 1756 de la siguiente manera.

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

Nota – La propiedad de conexión `MQ_SERVICE_PORT_PROPERTY` se introdujo por pimera vez en la versión 3.7, actualización 2 de Message Queue.

Requisitos de hardware y software

Para conocer cuáles son los requisitos de hardware y software de la versión 4.1, consulte la *Sun Java System Message Queue 4.1 Installation Guide*.

Acerca de Message Queue 4.0

Message Queue 4.0 es una versión que se limita a ser compatible con Application Server 9 PE. Se trata de una versión menor que incluye unas cuantas funciones nuevas, mejoras menores y soluciones de fallos. Esta sección incluye la siguiente información.

Novedades de la versión 4.0

Message Queue 4.0 incluye las siguientes funciones nuevas:

Se describen en las secciones secundarias siguientes.

Precaución – Uno de los cambios más insignificantes que incorpora la versión 4.0, pero que puede provocar interrupciones, es la desaprobación de la opción command-line para especificar una contraseña. En lo sucesivo, deberá guardar todas las contraseñas en un archivo, tal y como se describe en "Opción de contraseña desaprobada" en la página 42.

Cambios de interfaz en el API C y en el tiempo de ejecución del cliente C

La versión 4.0 de Message Queue incorpora dos nuevas propiedades que se establecerán en todos los mensajes que se hayan colocado en la cola de mensajes inactivos.

- `JMS_SUN_DMQ_PRODUCING_BROKER` indica al agente dónde se generó el mensaje.

- JMS_SUN_DMQ_DEAD_BROKER indica al agente quién marcó el mensaje como inactivo.

Cambios de interfaz en el API de Java y en el tiempo de ejecución del cliente de Java

La versión 4.0 de Message Queue incorpora dos nuevas propiedades que se establecerán en todos los mensajes que se hayan colocado en la cola de mensajes inactivos.

- JMS_SUN_DMQ_PRODUCING_BROKER indica al agente dónde se generó el mensaje.
- JMS_SUN_DMQ_DEAD_BROKER indica al agente quién marcó el mensaje como inactivo.

Muestra información sobre el almacén persistente

El subcomando query se ha añadido al comando imqdbmgr. Utilice este comando para mostrar información sobre el almacén persistente, como la versión del almacén, el usuario de la base de datos y si se han creado las tablas de la bases de datos.

A continuación, se muestra un ejemplo de la información mostrada por este comando.

```
imqdbmgr query

[04/Oct/2005:15:30:20 PDT] Utilizando el almacén persistente conectado:
        versión=400
        Iddebroker=Mozart1756
        url de la conexión de la base de datos=jdbc:oracle:thin:@Xhome:1521:mqdb
        usuario de la base de datos=scott
Ejecutándose en modo individual.
Ya se han creado las tablas de la base de datos.
```

Cambios en el formato del almacén persistente.

La versión 3.7 UR1 de Message Queue introdujo dos cambios en el formato del almacén persistente para mejorar el rendimiento: uno en el almacén del archivo, y el otro en el almacén de JDBC.

- El formato de los datos de transacción se mantuvo en el almacén del archivo.

  El formato de la información del estado de la transacción almacenado en el almacén persistente basado en archivos de Message Queue se cambió para reducir la E/S del disco y mejorar el rendimiento de las transacciones JMS.

- Almacén Oracle JDBC

  En versiones anteriores de Message Queue, el esquema del almacén para Oracle utilizaba el tipo de datos LONG RAW para guardar datos de mensajes. En Oracle 8, Oracle introdujo los tipos de datos BLOB y dejó de utilizar el tipo LONG RAW. BLOBductName; 3.7 UR1 cambió al tipo de datos BLOB para mejorar el rendimiento y la compatibilidad.

Dado que estos cambios afectan a la compatibilidad del almacén, la versión del almacén basado en archivos y en JDBC pasó de la 350 a la 370 en la versión 3.7 UR1 de Message Queue.

La versión 4.0 de Message Queue introdujo algunos cambios en el almacén JDBC para optimizar y admitir futuras mejoras. Por este motivo, se actualizó la versión del almacén JDBC a la 400. También es necesario señalar que, en la versión 4.0, la versión del almacén persistente basado en archivos sigue siendo la 370 porque no se han efectuado cambios en él.

Message Queue 4.0 admite la conversión automática del almacén persistente a las versiones más recientes de los almacenes persistentes basados en archivos y en JDBC. Si la utilidad detecta un almacén más antiguo la primera vez que se inicie `imqbrokerd`, cambiará el almacén al nuevo formato, prescindiendo del almacén antiguo.

- Las versiones 200 y 350 del almacén basado en archivos se convertirán al formato de la versión 370.
- Las versiones 350 y 370 del almacén basado en JDBC se convertirán al formato de la versión 400. (Si necesita actualizar un almacén de la versión 200, tendrá que pasar previamente por alguna de las versiones intermedias: 3.5 ó 3.6.)

Si necesita deshacer esta actualización, puede desinstalar Message Queue 4.0 y después volver a instalar la versión que estaba ejecutando anteriormente. Como la copia antigua del almacén se deja intacta, el agente puede ejecutarse con la copia antigua del almacén.

Administración del agente

La utilidad (`imqcmd`) ha incorporado un subcomando y varias opciones que permiten a los administradores desactivar el agente o cerrarlo después del intervalo especificado, destruir una conexión o establecer las propiedades del sistema java (por ejemplo, las propiedades relacionadas con la conexión.)

- Al desactivar el agente, éste pasa a un estado inactivo, lo cual permite drenar los mensajes antes de cerrar o reiniciar el agente. No es posible crear conexiones nuevas con un agente que está desactivado. Para desactivar el agente, introduzca un comando como el siguiente:

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

- Para que el agente se cierre después del intervalo especificado, introduzca un comando como el siguiente: (El intervalo de tiempo especifica el número de segundos que debe transcurrir para que se cierre el agente.)

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

  Si especifica un intervalo de tiempo, el agente registrará un mensaje indicando cuándo se producirá el cierre. Por ejemplo:

  ```
  El agente se cerrará en 29 segundos (29996 milisegundos)
  ```

  Mientras el agente espera a que se produzca el cierre, su comportamiento se verá afectado de la siguiente forma:

  - Seguirá aceptando las conexiones jms administrativas.
  - No aceptará nuevas conexiones jms.
  - Las conexiones jms existentes continuarán funcionando.

- El agente no podrá sustituir a ningún otro agente de un clúster de alta disponibilidad.
- La utilidad imqcmd no se bloqueará, sino que enviará la solicitud de cerrar el agente e informará inmediatamente.

- Para destruir una conexión, introduzca un comando como el siguiente:

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

  Utilice los comandos `imqcmd list cxn` o `imqcmd query cxn` para obtener el ID de conexión.

- Para establecer una propiedad con `imqcmd`, utilice la nueva opción –D. Es útil para establecer o anular las propiedades de fábrica de la conexión JMS o las propiedades del sistema java relacionadas con la conexión. Por ejemplo:

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

Encontrará toda la información sobre la sintaxis del comando `imqcmd` en el Capítulo 13, "Command Line Reference" de *Sun Java System Message Queue 4.1 Administration Guide*.

Compatibilidad con la persistencia de JDBC

Ahora se admite la versión 10.1.1 de Apache Derby como proveedor de almacenes persistentes compatibles con JDBC.

Compatibilidad con SSL

Desde la versión 4.0, el valor predeterminado de la propiedad de fábrica de la conexión del cliente `imqSSLIsHostTrusted` es `false`. Si su aplicación depende del valor predeterminado anterior (`true`), deberá cambiar la configuración y establecer la propiedad explícitamente a `true`.

Puede elegir si desea confiar en el host cuando el agente está configurado para utilizar certificados autofirmados. En este caso, además de especificar que la conexión debe usar un servicio de conexión basado en SSL (con la propiedad imqConnectionType), debería establecer la propiedad `imqSSLIsHostTrusted` en verdadera (true).

Por ejemplo, para ejecutar de forma segura las aplicaciones del cliente cuando el agente utiliza certificados autofirmados, utilice un comando como el siguiente:

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

Para ejecutar la herramienta de administración `imqcmd` de forma segura cuando el agente utiliza certificados autofirmados, utilice un comando como el siguiente:

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

Compatibilidad con JMX

Se ha añadido un nuevo API para configurar y supervisar los agentes de Message Queue de conformidad con la especificación de Java Management Extensions (JMX). Con este API, puede configurar y supervisar las funciones de agente de forma programática desde una aplicación de cliente de Message Queue. En versiones anteriores de Message Queue, sólo era posible acceder a estas funciones desde la línea de comando o desde la consola de administración.

beansI consiste en un conjunto de *beans gestionadas (MBeans)* que sirven para administrar los siguientes recursos relacionados con Message Queue:

- Agentes de mensaje
- Servicios de conexión
- Conexiones
- Destinos
- Productores de mensajes
- Consumidores de mensajes
- Transacciones
- Clústeres de agentes
- Registros
- La máquina virtual de Java (Java Virtual Machine, JVM)

Estas MBeans proporcionan *atributos* y *operaciones* para interrogar y manipular de forma síncrona el estado de los recursos subyacentes, y también *notificaciones* que permiten a una aplicación del cliente escuchar y responder de forma asíncrona a las cambios de estado a medida que se producen. Utilizando JMX API, las aplicaciones del cliente pueden realizar taras de configuración y supervisión como:

- Establecer el número de puerto de un agente
- Establecer el tamaño máximo de los mensajes de un agente
- Detener un servicio de conexión
- Establecer el número máximo de subprocesos para un servicio de conexión
- Averiguar el número de conexiones actuales que hay en un servicio
- Destruir una conexión
- Crear un destino
- Destruir un destino
- Activar o desactivar la creación automática de destinos
- Depurar todos los mensajes de un destino
- Obtener el número acumulativo de mensajes recibidos por un destino desde que se inició el agente.
- Averiguar el estado actual (en ejecución o en pausa) de una cola
- Averiguar el número actual de productores de mensajes de un tema

- Depurar todos los mensajes de un suscriptor duradero
- Averiguar el tamaño de un montón JVM

Encontrará una introducción del API de JMX e información completa de referencia en la *Sun Java System Message Queue 4.1 Developer's Guide for JMX Clients*.

Propiedades de compatibilidad del agente relacionadas con JMX

Se han añadido varias propiedades de agente nuevas para admitir el JMX API (consulte la Tabla 1–3). Ninguna de estas propiedades puede configurarse desde la línea de comando con la utilidad Command de Message Queue (`imqcmd`). Sólo es posible configurarlas con la opción `-D` de la utilidad Agente (`imqbrokerd`) o editarlas manualmente en el archivo de configuración de instancias de agente (`config.properties`). Además, algunas de estas propiedades (`imq.jmx.rmiregistry.start`, `imq.jmx.rmiregistry.use`, `imq.jmx.rmiregistry.port`) pueden configurarse con las opciones de la utilidad Agente que se describe en la Tabla 1–4. En esta tabla se enumeran todas las opciones, se especifica su tipo y se describe cómo utilizarlas

TABLA 1–3 Nuevas propiedades de agente para la compatibilidad con JMX

Propiedad	Tipo	Descripción
`imq.jmx.rmiregistry.start`	`Boolean`	Especifica si debe iniciarse el registro RMI al iniciarse el agente. truel valor es `true`, el agente iniciará un registro RMI en el puerto especificado por `imq.jmx.rmiregistry.port` y lo utilizará para guardar el cabo RMI de los conectores JMX. imqgjmxnrmiregistry el valor de `imq.jmx.rmiregistry.use` se ignora en este caso. **Valor predeterminado:** `false`
`imq.jmx.rmiregistry.use`	`Boolean`	Especifica si debe utilizarse un registro RMI. Sólo se cumple si `imq.jmx.rmiregistry.start` es `false`. Si el valor es `true`, el agente utilizará un registro RMI externo en el puerto especificado por `imq.jmx.rmiregistry.port` para guardar el cabo RMI de los conectores JMX. El registro RMI deberá estar ejecutándose ya al iniciarse el agente. **Valor predeterminado:** `false`

TABLA 1–3 Nuevas propiedades de agente para la compatibilidad con JMX *(Continuación)*

Propiedad	Tipo	Descripción
`imq.jmx.rmiregistry.port`	`Integer`	Número de puerto del registro RMI Sólo se cumple si `imq.jmx.rmiregistry.start` o `imq.jmx.rmiregistry.use` son `true`. URL conectores de JMXpueden configurarse después para que utilicen el registro RMI incluyendo este número de puerto en la ruta URL de las URL del servicio JMX. **Valor predeterminado:** `1099`
`imq.jmx.connector.list`	`String`	Nombres de conectores JMX preconfigurados y separados por comas **Valor predeterminado:** `jmxrmi,ssljmxrmi`
`imq.jmx.connector.activelist`	`String`	Nombres de conectores JMX que se activarán al iniciarse el agente, separados por comas **Valor predeterminado:** `jmxrmi`
`imq.jmx.connector.`*Nombre del conector* `.urlpath`	`String`	El componente*RutaURL* de la URL del servicio JMX para el conector *NombreDelConector* Esto es útil cuando la ruta de la URL del servicio JMX debe configurarse explícitamente (cuando se utiliza un registro externo compartido RMI). **Valor predeterminado:** Si se utiliza un registro RMI para almacenar el cabo RMI de los conectores de JMX (es decir, si `imq.jmx.registry.start` o `imq.jmx.registry.use` son `true`) `/jndi/rmi://`*HostDelAgente*`:`*Puertormi*`/`*HostDelAgente*`/`*PuertoDelAgente*`/`*NombreDelConector* Si no se utiliza un registro RMI (el caso predeterminado `imq.jmx.registry.start` y `imq.jmx.registry.use` ambos `false`): `/stub/`*Cabormi* donde *Cabormi* es una representacion codificada y serializada del mismo cabo RMI
`imq.jmx.connector.`*Nombre del conector* `.useSSL`	`Boolean`	Especifica si debe usarse una capa de zócalos seguros (SSL) para el conector *NombreDelConector*. **Valor predeterminado:** `false`

TABLA 1–3 Nuevas propiedades de agente para la compatibilidad con JMX *(Continuación)*

Propiedad	Tipo	Descripción
`imq.jmx.connector.`*Nombre del conector*`.brokerHostTrusted`	`Boolean`	Especifica si debe confiarse en alguno de los certificados presentados por el agente del conector *NombreDelConector*. Sólo se cumple cuando `imq.jmx.connector.`*NombreDelConector*`.useSSL` es `true`. Si es `false`, el tiempo de ejecución del cliente de Message Queue validará todos los certificados que se le presenten. La validación fallará si el firmante del certificado no se encuentra en el almacén de confianza del cliente. Si es `true`, se omitirá la validación del certificado. Esto puede ser útil, por ejemplo, durante la fase de pruebas de un software o cuando se utiliza un certificado autofirmado. **Valor predeterminado:** `false`

La propiedad `imq.jmx.connector.list` define el conjunto de conectores de JMX con nombre que se crearán al iniciarse el agente, mientras que `imq.jmx.connector.activelist` especifica cuál de ellos debe activarse. Cada conector con nombre tendrá su propio conjunto de propiedades:

`imq.jmx.connector.`*NombreDelConector* `.urlpath`
`imq.jmx.connector.`*NombreDelConector* `.useSSL`
`imq.jmx.connector.`*NombreDelConector* `.brokerHostTrusted`

Por defecto, se crean dos conectores JMX llamados `jmxrmi` y `ssljmxrmi`; el primero está configurado para no utilizar el cifrado SSL (`imq.jmx.connector.jmxrmi.useSSL` = `false`, y el segundo, para utilizarlo (`imq.jmx.connector.ssljmxrmi.useSSL` = `true`). Por defecto, sólo se activa el conector `jmxrmi` al iniciarse el agente; consulte "Compatibilidad con SSL para los clientes de JMX" en la página 25 para conseguir información sobre cómo activar el conector `ssljmxrmi` y establecer comunicaciones seguras.

Para mayor comodidad, también se han añadido nuevas opciones (Tabla 1–4) a la utilidad Agente de la línea de comando (`imqbrokerd`) que permiten controlar el uso, el inicio y el puerto del registro RMI. El uso y los efectos de estas opciones son los mismos que los de las propiedades equivalentes del agente, que se describen en la Tabla 1–3. Esta tabla incluye una lista de todas las opciones, especifica la propiedad equivalente del agente y describe cómo se utilizan.

TABLA 1–4 Nuevas opciones de la utilidad Agente para la compatibilidad con JMX

Opción	Propiedad equivalente del agente	Descripción
`-startRmiRegistry`	`imq.jmx.rmiregistry.start`	Especifica si debe iniciarse el registro RMI al iniciarse el agente.
`-useRmiRegistry`	`imq.jmx.rmiregistry.use`	Especifica si debe utilizarse un registro RMI.
`-rmiRegistryPort`	`imq.jmx.rmiregistry.port`	El número de puerto del registroRMI

Se ha añadido un nuevo subcomando (Tabla 1–5) a la utilidad Command de la línea de comando (`imqcmd`) que enumera las URL del servicio JMX de los conectores JMX creados e iniciados al iniciarse el agente. Esta información la necesitan los clientes de JMX que no utilizan la clase de conveniencia de Message Queue convenience class `AdminConnectionFactory` para obtener sus conectores JMX; también puede utilizarse pra gestionar o supervisar Message Queue mediante un navegador de JMX genérico como Monitoring and Management Console (`jconsole`).

TABLA 1–5 Nuevo subcomando de la utilidad Command

Subcomando	Descripción
`list jmx`	Ofrece una lista de las URL de los conectores JMX del servicio JMX

Compatibilidad con SSL para los clientes de JMX

Como se mencionó más arriba, un agente de mensajes d eMessage Queue está configurado por defecto para establecer comunicaciones no seguras con el conector JMX preconfigurado `jmxrmi`. Las aplicaciones que deseen utilizar la capa de zócalos seguros (SSL) para establecer comunicaciones seguras, deberán activar el otro conector JMX seguro: `ssljmxrmi`. Para ello, debe seguir estos pasos:

1. Obtenga e instale un certificado firmado de la misma manera que para los servicios de conexión `ssljms`, `ssladmin` o `cluster`, tal y como se describe en la *Message Queue Administration Guide*.
2. Instale en el almacén de confianza el certificado raíz de la autoridad de certificación, si es necesario.
3. Añada el conector `ssljmxrmi` a la lista de conectores JMX que deben activarse al iniciar el agente:

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

4. Inicie el agente con la utilidad Agente de Message Queue (`imqbrokerd`), bien facilitándole la contraseña key-store de un archivo de claves, bien escribiéndola desde la línea de comando cuando se le indique.

5. ssljmxrmito, el conector `ssljmxrmi` (o cualquier otro conector basado en SSL) está configurado para validar todos los certificados SSL del agente que se le presenten. Si desea evitar esta validación (por ejemplo, cuando utilice certificados autofirmados o durante la prueba de un software), establezca la propiedad del agente `imq.jmx.connector.ssljmxrmi.brokerHostTrusted` en `true`.

En el lado del cliente, la fábrica de conexión del administrador (`AdminConnectionFactory`) deberá configurarse con una URL que especifique `ssljmxrmi` como el conector preferido:

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

Si es preciso, utilice las propiedades del sistema `javax.net.ssl.trustStore` y `javax.net.ssl.trustStorePassword` para indicar al cliente de JMX el almacén de confianza.

Registro del tiempo de ejecución del cliente

Esta sección describe la compatibilidad de Message Queue 4.0 con registros de tiempo de ejecución del cliente de sucesos de conexión y eventos relacionados con la sesión

JDK 1.4 (y versiones superiores) incluye la biblioteca `java.util.logging`. Esta biblioteca implementa una interfaz de registro estándar que puede utilizarse para registros específicos de la aplicación.

El tiempo de ejecución del cliente de Message Queue utiliza Java Logging API para implementar sus funciones de registro. Puede utilizar todos los servicios de registro de J2SE 1.4 para configurar las actividades de registro. Por ejemplo, una aplicación puede utilizar los siguientes servicios de registro de Java para establecer cómo el tiempo de ejecución del cliente de Message Queue debe presentar su información de registro:

- Controladores de registro
- Filtros de registro
- Formateadores de registro
- Nivel de registro

Para más información sobre el Java Logging API, consulte la presentación de este producto en `http://java.sun.com/j2se/1.4.2/docs/guide/util/logging/overview.html`

Espacios de nombre de registro, niveles y actividades

El proveedor de Message Queue define un conjunto de espacios de nombre de registro asociados a los niveles de registro y a actividades de registro que permiten a los clientes de Message Queue crear un registro de los sucesos de conexión y de sesión siempre que se haya establecido correctamente la configuración de registro.

El espacio del nombre de registro raíz del tiempo de ejecución de Message Queue está definido como `javax.jms`. Todos los registros del tiempo de ejecución del cliente de Message Queue utilizan este nombre como espacio de nombres padre.

Los niveles de registro del tiempo de ejecución del cliente de Message Queue son los mismos que los definidos en la clase `java.util.logging.Level`. Esta clase define siete niveles de registro estándar y dos ajustes adicionales que puede utilizar para desactivar y activar el proceso de registro.

`OFF`	Desactiva el registro.
`SEVERE`	Cuanto mayor es la prioridad, más alto es el valor. Definido por la aplicación.
`WARNING`	Definido por la aplicación.
`INFO`	Definido por la aplicación.
`CONFIG`	Definido por la aplicación.
`FINE`	Definido por la aplicación.
`FINER`	Definido por la aplicación.
`FINEST`	Cuanto menor es la prioridad, menor es el valor. Definido por la aplicación.
`ALL`	Activa el registro de todos los mensajes.

En general, las excepciones y los errores que se producen en el tiempo de ejecución del cliente de Message Queue son registrados con el espacio de nombre `javax.jms`.

- Las excepciones que devuelve JVM y que capta el tiempo de ejecución del cliente, como `IOException`, quedan registradas con el espacio de nombre de registro `javax.jms` en el nivel `WARNING`.
- Las excepciones de JMS que devuelve el tiempo de ejecución del cliente, como `IllegalStateException` , quedan registradas con el espacio de nombre de registro `javax.jms` en el nivel `FINER`.
- Los errores que devuelve JVM y que capta el tiempo de ejecución del cliente, como `OutOfMemoryError`, quedan registrados con el espacio de nombre de registro `javax.jms` en el nivel `SEVERE`.

La siguiente tabla muestra una lista de los sucesos que pueden registrarse y el nivel de registro que debe establecerse en los sucesos de registro para las conexiones de JMS y las sesiones.

La siguiente tabla describe los niveles de registro y los sucesos de conexiones.

TABLA 1–6 Niveles de registro y sucesos para el espacio de nombre `javax.jms.connection`

Nivel de registro	Sucesos
`FINE`	Conexión creada
`FINE`	Conexión iniciada
`FINE`	Conexión cerrada

TABLA 1–6 Niveles de registro y sucesos para el espacio de nombre `javax.jms.connection` *(Continuación)*

Nivel de registro	Sucesos
`FINE`	Conexión interrumpida
`FINE`	Conexión reanudada
`FINER`	Diversas actividades de conexión como `setClientID`
`FINEST`	Mensajes, reconocimientos, mensajes de acción y control de Message Queue (como confirmar una transacción)

Para las sesiones, se deja constancia de la siguiente información en el registro.

- Cada registro de un mensaje entregado a un consumidor incluye el ID de la conexión, el ID de la sesión y el ID del consumidor.
- Cada registro de un mensaje enviado por un productor incluye el ID de la conexión, el ID de la sesión y el ID del productor, además del nombre del destino.

La siguiente tabla describe los niveles de registro y los sucesos de las sesiones.

TABLA 1–7 Niveles de registro y sucesos para el espacio de nombre `javax.jms.session`

Nivel de registro	Suceso
`FINE`	Sesión creada
`FINE`	Sesión cerrada
`FINE`	Productor creado
`FINE`	Consumidor creado
`FINE`	Destino creado
`FINER`	Diversas actividades de sesión como confirmar una sesión.
`FINEST`	Mensajes producidos y consumidos. (Las propiedades y los cuerpos de los mensajes no se consignan en los registros.)

Por defecto, el nivel de registro de salida se hereda del JRE en que se esté ejecutando la aplicación. Compruebe el archivo `JRE_DIRECTORY/lib/logging.properties` para determinar cuál es el nivel.

Puede configurar el registro de forma programática o mediante archivos de configuración, de forma que pueda controlar el ámbito en que se produce el registro. En las siguientes secciones se describen estas posibilidades.

Mediante el archivo de configuración de registro JRE

El siguiente ejemplo muestra cómo establecer espacios de nombre y niveles de registro en el archivo `JRE_DIRECTORY/lib/logging.properties`, que se utiliza para establecer el nivel de registro del entorno en tiempo de ejecución de Java. Todas las aplicaciones que utilicen este JRE tendrán la misma configuración de registro. El ejemplo de configuración que se muestra abajo establece el nivel de registro en `INFO` para el espacio de nombre `javax.jms.connection` y especifica que la salida debe escribirse en `java.util.logging.ConsoleHandler` .

```
#registro.archivo de propiedades.
# "controladores" especifica una lista separada por comas de clases
# de controladores de registro. Estos controladores se instalan durante el inicio de VM.
# Tenga en cuenta que estas clases deberán estar en la ruta de clase del sistema.
# Por defecto, solo configuramos un ConsoleHandler, que sólo mostrará
# mensajes en el nivel INFO y superiores.

    controladores= java.util.logging.ConsoleHandler

# Nivel de registro global predeterminado.
# Especifica qué tipo de sucesos se registran en todos los
# registros. Es posible anular este nivel global para un recurso
# determinado y sustituirlo por un nivel específico para el recurso.
# Observe que el ConsoleHandler también tiene una configuración de nivel
# independiente para limitar los mensajes que se imprimen en la consola.

    .nivel= INFO

#Limitar los mensajes que se imprimen en la consola a INFO y niveles superiores.

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

# Este registro con espacio de nombre de conexión javax.jms. escribirá
# mensajes de nivel INFO en sus controladores de salida. En esta configuración
# el controlador de entrada está establecido en java.util.logging.ConsoleHandler.

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

Utilizar un archivo de configuración de registro para una aplicación específica

También puede definir un archivo de configuración de registro desde la línea de comandos de Java que utiliza para ejecutar una aplicación. La aplicación utilizará la configuración definida en el archivo de registro especificado. En el siguiente ejemplo, `configFile` utiliza el mismo formato definido en el archivo `JRE_DIRECTORY/lib/logging.properties` .

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

Establecer la configuración de registro de forma programática

El siguiente código utiliza el API `java.util.logging` para registrar sucesos de conexión y lo hace cambiando el nivel del espacio de nombre `javax.jms.connection` a `FINE`. Puede incluir este código en su aplicación para establecer la configuración de registro de forma programática.

```
import java.util.logging.*;
//construye un controlador de archivo y una salida en el archivo mq.log
//del directorio temp del sistema.

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

//Obtener registrador para el dominio  "javax.jms.connection".

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

//el registrador javax.jms.connection registraría las actividades
//de nivel FINE y superior.

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

Notificación de sucesos de conexión

Las notificaciones de los sucesos de conexión permiten al cliente de Message Queue escuchar los sucesos de cierre y reconexión, y tomar las medidas oportunas en función del tipo de notificación y del estado de conexión. Por ejemplo, si se produce un error y el cliente se vuelve a conectar a otro agente, es posible que una aplicación quiera limpiar el estado de su transacción y proceder con una nueva.

Si el proveedor de Message Queue detecta un problema grave con una conexión, invocará a la escucha de excepción registrada del objeto de conexión. Llama al método `onException` y le envía un argumento `JMSException` que describe el problema. El proveedor de Message Queue también ofrece un API de notificación de sucesos que permite al tiempo de ejecución del cliente informar a la aplicación sobre los cambios que se produzcan en el estado de la conexión. El API de notificación se define por los siguientes elementos:

- El paquete `com.sun.messaging.jms.notification`, que define la escucha de sucesos y los objetos de suceso de notificaciones.
- La interfaz `com.sun.messaging.jms.Connection`, que define las extensiones de la interfaz `javax.jms.Connection`.

Las siguientes secciones describen los eventos que pueden desencadenar una notificación y explican cómo crear una escucha de sucesos.

Sucesos de conexión

La siguiente tabla muestra una lista de los sucesos que puede devolver la escucha de sucesos.

Tenga en cuenta que la escucha de excepción de JMS no se invoca cuando se produce un suceso de conexión. La escucha de excepciones sólo se invoca cuando el tiempo de ejecución del cliente ha agotado sus intentos de reconexión. El tiempo de ejecución del cliente siempre invoca la escucha de sucesos antes que la escucha de excepciones.

TABLA 1–8 Sucesos de notificación

Tipo de suceso	Significado
`ConnectionClosingEvent`	El tiempo de ejecución del cliente de Message Queue genera este suceso cuando el agente le notifica que está a punto de cerrarse una conexión debido a que el administrador ha solicitado el cierre.
`ConnectionClosedEvent`	El tiempo de ejecución del cliente de Message Queue genera este suceso cuando se cierra una conexión debido a un error del agente o porque el administrador haya solicitado cerrarla o reiniciarla. Cuando una escucha de sucesos recibe un `ConnectionClosedEvent`,, la aplicación puede utilizar el método `getEventCode()` del suceso recibido para obtener un código de sucesos que especifique la causa del cierre.
`ConnectionReconnectedEvent`	El tiempo de ejecución del cliente de Message Queue ha vuelto a conectar con un agente. Este agente podría ser el mismo con el que el cliente estaba conectado antes u otro diferente. Una aplicación puede utilizar el método `getBrokerAddress` del suceso recibido para obtener la dirección del agente con el que ha vuelto a establecer conexión.
`ConnectionReconnectFailedEvent`	El tiempo de ejecución del cliente de Message Queue no ha conseguido reconectar con un agente. Siempre que un intento de reconexión falla, el tiempo de conexión genera un nuevo suceso y lo transmite a la escucha de sucesos. La escucha de excepción de JMS no se invoca cuando se produce un suceso de conexión. Sólo se invoca cuando el tiempo de ejecución del cliente ha agotado sus intentos de reconexión. El tiempo de ejecución del cliente siempre invoca la escucha de sucesos antes que la escucha de excepciones.

Crear una escucha de sucesos

El siguiente código de ejemplo muestra cómo establecer una escucha de sucesos de conexión. Siempre que se produce un suceso de conexión, el tiempo de ejecución del cliente invocará el método `onEvent` de la escucha del suceso.

```
//crear una fábrica de conexión de MQ.

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

//crear una conexión de MQ.

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

//construir una escucha de sucesos de MQ.  La escucha implementa
//com.sun.messaging.jms.notification.EventListener interface.

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

//establecer una escucha de sucesos en la conexión MQ.

connection.setEventListener ( eListener );
```

Ejemplos de escuchas de sucesos

En este ejemplo, una aplicación elige ordenar a su escucha de sucesos que registre el evento de conexión en el sistema de registro de la aplicación:

```
public class ApplicationEventListener implements
                com.sun.messaging.jms.notification.EventListener {

vacío público onEvent ( com.sun.messaging.jms.notification.Event connEvent ) {
          registro (connEvent);
}
registro de vacío privado ( com.sun.messaging.jms.notification.Event connEvent ) {
          String eventCode = connEvent.getEventCode();
          String eventMessage = connEvent.getEventMessage();
          //escribir información de eventos en el flujo de salida.
         }
}
```

Requisitos de hardware y software

Para conocer los requisitos de hardware y software de la versión 4.0, consulte las notas de la versión de Sun Java System Application Server Platform Edition 9.

Errores solucionados en esta versión

La siguiente tabla muestra los errores que se han solucionado en la versión 4.1 de Message Queue.

TABLA 1–9 Errores solucionados en Message Queue 4.1

Error	Descripción
6381703	Los mensajes remotos tramitados pueden confirmarse dos veces si se reinicia el agente que lo ha originado.
6388049	No es posible deshacer una transacción distribuido incompleta
6401169	Las opciones "confirmar" y "deshacer" de imqcmd no muestran un mensaje de confirmación.
6473052	El valor predeterminado de las colas autocreadas debería ser "round robin" (operación por turnos). (`MaxNumberConsumers` = -1).
6474990	El registro del agente muestra `ConcurrentModificationException` para el comando `imqcmd list dst`.
6487413	Se produce una pérdida de memoria cuando el comportamiento límite es `REMOVE_OLDEST` o `REMOVE_LOWER_PRIORITY`.
6488340	El agente hace un "giro" y el cliente espera a que se atienda la respuesta.
6502744	El agente no cumple el límite predeterminado de 1000 mensajes de la cola de mensajes inactivos.
6517341	El tiempo de ejecución del cliente tiene que mejorar la lógica de reconexión cuando el cliente se conecta con un clúster de alta disponibilidad y permitir al cliente volver a conectarse con independencia de cual sea el valor de la propiedad *imqReconnectEnabled*.
6528736	El servicio de inicio automático de Windows (*imqbrokersvc*) se bloquea durante el inicio.
6561494	Los mensajes se entregan al consumidor equivocado cuando ambos comparten una sesión.
6567439	Los mensajes generados en una transacción `PREPARED` se entregan sin orden si se confirman después de que se reinicie el agente.

La siguiente tabla muestra los errores solucionados en Message Queue 4.0.

TABLA 1–10 Errores solucionados en Message Queue 4.0

Número de error	Descripción
4986481	En Message Queue 3.5, la invocación de `Session.recover` podría bloquearse en el modo de conexión automática.
4987325	El indicador de nueva entrega se establecía en `false` en mensajes entregados de nuevo después de invocar a `Session.recover`.
6157073	Cambiar el mensaje de nueva conexión para que incluya el número de conexiones que se encuentran en el servicio además del número total de conexiones.
6193884	Message Queue muestra un mensasyslogura en syslog en configuraciones regionales que utilizan caracteres no ASCII en los mensajes.
6196233	No funciona la selección de mensajes con `JMSMessageID`.
6251450	`ConcurrentModificationException` en `connectList` durante apagado de clúster.
6252763	`java.nio.BufferOverflowException` en `java.nio.HeapByteBuffer.putLong/Int` .
6260076	El primer mensaje publicado tras el inicio es lento cuando utiliza almacenamiento Oracle.
6260814	El procesamiento del Selector en `JMSXUserID` siempre evalúa como `falso`.
6264003	El explorador de colas muestra mensajes que son parte de transacciones no confirmadas.
6271876	El control de flujo de la conexión no funciona correctamente cuando se cierra un consumidor que tiene mensajes no consumidos.
6279833	Message Queue no debería permitir que dos agentes utilicen las mismas tablas jdbc.
6293053	El agente maestro no se inicia correctamente cuando cambia la dirección IP del sistema, a menos que se borre el almacén (con `–reset store`.)
6294767	El agente de Message Queue tiene que establecer `SO_REUSEADDR` en los sockets de la red que abra.
6304949	No es posible configurar la propiedad `ClientID` de `TopicConnectionFactory.`
6307056	El registro txn tiene un efecto adverso sobre el rendimiento.
6320138	La API de C de Message Queue no tiene la capacidad de determinar el nombre de una cola de una cabecera de respuesta.
6320325	El agente a veces utiliza JDK 1.4 en vez de JDK 1.5 en Solaris, incluso si ambas versiones se encuentran instaladas.
6321117	El clúster de múltiples agentes lanza la excepción `java.lang.NullPointerException` .
6330053	El cliente de jms devuelve `java.lang.NoClassDefFoundError` al confirmar una transacción desde el suscriptor.
6340250	Compatibilidad con el tipo `MESSAGE` en la API de C.

TABLA 1–10 Errores solucionados en Message Queue 4.0 *(Continuación)*

Número de error	Descripción
6351293	Se incluye compatibilidad con la base de datos Apache Derby.

Información importante

Este apartado contiene la información más reciente que no se incluye en la documentación principal del producto. En este capítulo se tratan los siguientes temas:

Notas de la instalación

Consulte la *Sun Java System Message Queue 4.1 Installation Guide* para obtener información sobre las instrucciones previas a la instalación, los procedimientos de actualización y las demás informaciones importantes para instalar Message Queue, Platform Edition en las plataformas Solaris, Linux y Windows.

Consulte la *Guía de instalación de Sun Java Enterprise System* para obtener información sobre las instrucciones previas a la instalación y el resto de informaciones importantes para instalar Message Queue, Enterprise Edition en las plataformas Solaris, Linux y HPUX.

Consulte la *Guía de actualización y migración de Sun Java Enterprise System* para obtener información relevante sobre la actualización y la migración de Message Queue Enterprise Edition en las plataformas Solaris, Linux, HPUX y Windows.

Problemas de compatibilidad

En esta sección se tratan los problemas de compatibilidad de Message Queue 4.1.

Estabilidad de interfaz

Sun Java System Message Queue utiliza muchas interfaces que pueden cambiar con el tiempo. El Apéndice B, "Stability of Message Queue Interfaces" de *Sun Java System Message Queue 4.1 Administration Guide* incluye una clasificación de las interfaces según su estabilidad. Cuanto más estable sea la interfaz, menos probable es que cambie en las siguientes versiones del producto.

Problemas en relación a la próxima versión importante de Message Queue

La próxima versión importante de Message Queue puede introducir cambios que hagan que sus clientes dejen de ser compatibles con el producto. Esta información se proporciona ahora para permitirle anticipar dichos cambios.

- Las ubicaciones de los archivos instalados como parte de Sun Java System Message Queue pueden cambiar. Esto puede hacer que dejen de funcionar algunas de las aplicaciones que dependen de la ubicación actual de archivos de Message Queue.
- El agente 3.5 y anteriores puede que dejen de poder funcionar en un clúster con agentes más recientes.
- En futuras versiones, es posible que los clientes de Message Queue no puedan utilizar versiones de JDK anteriores a la 1.5.

Actualizaciones de documentación para Message Queue 4.1

Además de este documento de *Notas de la versión*, Message Queue 4.1 incluye un único documento nuevo: *Sun Java System Message Queue 4.1 Developer's Guide for JMX Clients*. Este documento se incluyó por primera vez en la versión 4.0 de Message Queue. En la versión 4.1, se ha añadido información conceptual para presentar el modelo JMX.

La documentación de Message Queue, que se publicó para Message Queue 3.6 SP3, 2005Q4, se ha actualizado para ajustarse a las necesidades de los clientes de Application Server 9 PE. Todos estos documentos pueden encontrarse en la siguiente dirección:

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

Información de instalación y de actualización

La *Sun Java System Message Queue 4.1 Installation Guide* ha sido actualizada parar reflejar información específica de cada plataforma. Esta documento contiene ahora información importante sobre la instalación y la actualización de Message Queue 4.1.

Guía de administración

La *Guía de administración* se ha actualizado para ofrecer información sobre los clústeres de alta disponibilidad y la compatibilidad con JAAS y con JMX.

Guía del desarrollador para clientes Java

La *Guía del desarrollador para los clientes de Java* se ha actualizado para reflejar la nueva compatibilidad con los registros de tiempo del ejecución y de notificación de sucesos de conexión.

Guía del desarrollador para clientes de C

La *Guía del desarrollador para clientes de C* se ha actualizado para reflejar la incorporación de la función `MQGetDestinationName` del tipo de mensaje `MQ_Message` y de los puertos fijos.

Limitaciones y problemas conocidos

Esta sección contiene una lista de los problemas conocidos de Message Queue 4.1. Se describen las siguientes áreas del producto:

Para encontrar una lista de los fallos actuales, sus estados y sus soluciones alternativas, invitamos a los miembros de Java Developer Connection™ a visitar la página Bug Parade del sitio web Java Developer Connection. Compruebe la página antes de informar de un nuevo error. A pesar de que no se muestran todos los problemas de Message Queue la página es un buen punto de partida para ver si un problema ha sido comunicado.

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

Nota – Aunque la suscripción a Java Developer Connection es gratuita, es necesario registrarse. Encontrará información sobre cómo ser miembro de Java Developer Connection en la página web "For Developers" de Sun.

Para informar de un nuevo problema o enviar una petición de sobre nuevas funcionalidades, envíe un mensaje a `imq-feedback@sun.com` .

Problemas de instalación

Esta sección describe los problemas relacionados con la instalación de la versión 4.1 de Message Queue.

Registro del producto y JES

La versión 4.1 de Message Queue se instala mediante un nuevo programa de instalación, que también instala y actualiza los componentes compartidos que necesita Message Queue; por ejemplo, JDK, las bibliotecas de NSS, JavaHelp, etc. Este programa de instalación y el instalador

de Java Enterprise System (JES) no comparten el mismo registro del producto. Si una versión de Message Queue que se instaló con JES se elimina y se actualiza a Message Queue 4.1 con el instalador de Message Queue, es posible que el registro del producto de JES se encuentre en un estado inconsistente. Esto significa que, al ejecutar el desinstalador de JES, es posible que se elimine sin querer Message Queue 4.1 y los componentes compartidos de los que depende y que no instaló él.

La mejor manera de actualizar el software que instaló el instalador de JES es la siguiente:

1. Desinstale Message Queue y sus componentes compartidos con el desinstalador de JES.
2. Vuelva a instalar Message Queue 4.1. con el instalador de Message Queue.

Seleccione el JRE adecuado

La pantalla de selección de JDK del instalador de Message Queue 4. 1 le permite seleccionar el JDK y JRE que exista en el sistema para que lo utilice Message Queue. Desafortunadamente, la lista que aparece también incluye el JRE utilizado para ejecutar la aplicación del instalador. Este JRE forma parte del paquete del instalador y no se instala realmente en el sistema. *(Fallo 6585911)*

El JRE que utiliza el instalador es reconocible por su ruta, que debería encontrarse en el directorio comprimido del instalador y debería incluir el subdirectorio `mq4_1–`. Por ejemplo:

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

No seleccione este JRE para que lo use Message Queue. Seleccione en su lugar otro JDK del sistema. Si no existe ninguno, siga la indicación recomendada para su plataforma.

- Solaris o Linux: Seleccione "Instalar y utilizar el JDK predeterminado".
- Windows: Descargue e instale un JDK antes de ejecutar el instalador de Message Queue 4.1.

Instalación en Windows

Al instalar Message Queue en Windows, tenga presente las siguientes limitaciones:

- El instalador no añade entradas de Message Queue en el menú Inicio>Programas (*Fallo 6567258*). Para iniciar la consola de administración, utilice la línea de comando tal y como se indica en “Starting the Administration Console” de *Sun Java System Message Queue 4.1 Administration Guide*.
- El instalador no añade el directorio `IMQ_HOME\mq\bin` al entorno variable `PATH`.*(Fallo 6567197)*. Los usuarios deben añadir esta entrada a su entorno variable `PATH` o proprocionar un nombre de ruta completa al invocar las utilidades de Message Queue (`IMQ_HOME\mq\bin\`*comando*).
- El instalador no agrega entradas al registro de Windows para indicar que Message Queue está instalado.

- Al ejecutarlo en modo silencioso, el instalador retorna inmediatamente. La instalación se lleva a cabo, pero el usuario no tiene forma de saber cuándo se ha completado la instalación silenciosa. *(Fallo 6586560)*
- El modo de texto (`installer –t`) no se admite en Windows. Al ejecutar el instalador en modo de texto en Windows, aparece un mensaje de error. Este mensaje se muestra en inglés aunque el instalador se haya ejecutado en configuraciones regionales de lengua no inglesa. *(Fallo 6594142)*
- La cadena "Install Home" (Inicio de instalación) aparece en inglés en la pantalla del instalador aunque éste se ejecute en configuraciones regionales de lengua no inglesa. *(Fallo 6592491)*

Instalación en Solaris

Un mensaje de error y el estado "incompleto" del resumen confunde al usuario que intenta hacer la instalación con el comando `installer-n`. En realidad, el comando funciona. *(Fallo 6594351)*

Instalación en Linux

Estos son los problemas que se producen al hacer la instalación en la plataforma Linux

- En el panel de selección de JDK, la lista desplazable sólo muestra un elemento. Esto dificulta la selección de otro JDK de la lista. *(Fallo 6584735)*
- Si el JDK es actual y el usuario selecciona "Instalar JDK predeterminado" en la pantalla de selección de JDK, el instalador seguirá intentando instalarlo e informará de que no puede instalar el paquete. La instalación se realiza correctamente a pesar de este problema. *(Fallo 6581310)*
- Si el instalador se ejecuta en modo de ejecución seca (`installer –n`), la pantalla de resumen muestra algunos mensajes de error y el estado de instalación como "incompleto": Esto es incorrecto y confunde al usuario; una ejecución seca no instala nada en el sistema, sino que crea sólo un archivo de respuesta que puede utilizarse después para hacer la instalación.*(Fallo 6594351)*
- Si en su sistema existen versiones anteriores de localización RPM de Message Queue, la instalación de la versión 4.1 de este producto (que se produce al marcar la casilla "Instalar paquetes multilingües de Message Queue" en la pantalla Paquetes multilingües) fallará. La instalación fracasa debido a que se produce un conflicto con los paquetes Il8 de una instalación anterior de 3.7 UR1. *(Fallo 6594381)*

  Solución alternativa Elimine manualmente la localización RPM con el comando `rpm –e` antes de ejecutar el instalador 4.1. Para determinar qué RPM es relevante aquí, consulte “Message Queue Packages (RPMs)” de *Sun Java System Message Queue 4.1 Installation Guide*.

Instalación en todas las plataformas

Los siguientes problemas afectan a la instalación en todas las plataformas:

- Cuando el instalador está en proceso de instalar Message Queue 4.1 y aparece la pantalla de progreso, el botón Cancelar está activo. Si en ese momento se pulsara el botón Cancelar, la instalación quedaría incompleta o interrumpida. *(Fallo 6595578)*
- La pantalla resumen del instalador contiene una serie de vínculos que, al pulsar sobre ellos, activan un registro o un visor de página resumen. Al cerrar la ventana de este visor con el botón X de la ventana en lugar de con el botón marcado con la etiqueta "cerrar", ya no podrá volver a abrir la ventana del visor. *(Fallo 6587138)*

  Solución alternativa Cierre la ventana con el botón de la etiqueta Cerrar.
- Cuando un sistema tiene versiones anteriores de Message Queue y NSS/NSPR, la actualización del instalador sólo enumera los Message Queue que necesitan actualizarse, pero no indica si NSS/NSPR también lo necesita. Este problema sólo existe con la pantalla de actualización, ya que todo el software relevante se actualizará como parte del proceso de instalación (tal y como se indica en la pantalla PreparadoParaInstalarse que muestra la información correcta). *(Fallo 6580696)*

  Solución alternativa No hace falta ninguna, ya que los archivos NSS/NSPR se instalan si no están actualizados y se desinstalan las versiones más antiguas.
- Cuando el instalador o el desinstalador se ejecutan en modo texto (`installer –t`), la pantalla Resumen muestra el directorio que contiene los archivos de registro y de resumen pero no enumera los nombres de estos archivos. *(Fallo 6581592)*
- Si se especifica el nombre de un archivo que no existe, aparece un mensaje de error incoherente y confuso. *(Error 6587127)*

Información de versión

El instalador muestra la información de la versión de Message Queue de una manera poco clara. *(Fallo 6586507)*

En la plataforma Solaris, consulte la tabla siguiente para determinar la versión que se está instando.

TABLA 1–11 Formatos de versión

La versión como se muestra en el instalador	Versión Message Queue
4.1.0.0	4.1
3.7.0.1	3.7 UR1
3.7.0.2	3.7 UR2

TABLA 1–11 Formatos de versión *(Continuación)*

La versión como se muestra en el instalador	Versión Message Queue
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

Nota – Para versiones con parche de 3.6 SP4 (por ejemplo, 3.6 SP4 Patch 1), la cadena de las versiones que muestra el instalador sigue siendo la misma. Debe ejecutar el comando `imqbrokerd –version` para determinar la versión exacta.

En la plataforma Linux, no es posible ofrecer una traducción sencilla del formato. El número de versión que muestra el instalador en Linux es de la siguiente forma:

```
<NúmVersionMayor>.<NúmVersionMenor>-<unNúm.>
```

Por ejemplo, 3.7–22. Esto quiere decir que se trata de una de las versiones 3.7, pero no especifica cuál. Para saber cuál es, ejecute el comando `imqbrokerd –version` .

Problemas de localización

Estos son los problemas relacionados con la localización.

- Cuando el instalador se ejecuta en modo texto (`installer –t`) en una configuración regional de lengua no inglesa, los caracteres de varios bytes aparecen como texto basura. *(Fallo 6586923)*
- La pantalla de resumen de la instalación permite al usuario ver un informe resumido. Desafortunadamente, este informe (una página HTML) muestra texto basura cuando el instalador se ejecuta en configuraciones regionales de varios bytes. *(Error 6587112)*

  Solución alternativa Edite el archivo HTML para corregir el conjunto de caracteres específicos. El archivo HTML debería incluir algo parecido a esto:

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

  Sustituya "UTF-8" por *locale_name* `.UTF-8`. Por ejemplo, `ja_JA.UTF-8` o `ko.UTF-8` en Solaris; `ja_JA.utf8` o `ko_KO.utf8` en Linux.

- En la pantalla de progreso del instalador, la barra de progreso muestra caracteres extraños. La información sobre herramientas no es modificable en configuraciones regionales de lengua no inglesa. *(Fallo 6591632)*
- El modo de texto (`installer –t`) no se admite en Windows. Al ejecutar el instalador en modo de texto en Windows, aparece un mensaje de error. Este mensaje se muestra en inglés aunque el instalador se haya ejecutado en configuraciones regionales de lengua no inglesa. *(Fallo 6594142)*
- La pantalla de licencia del instalador muestra el texto de licencia en inglés, con independencia de la configuración regional en la que se haya ejecutado. *(Fallo 6592399)*

  Solución alternativa Para acceder a los archivos de licencia traducidos, busque el archivo `LICENSE_MULTILANGUAGE.pdf`.
- El texto de ayuda sobre el uso del instalador no está traducido. *(Fallo 6592493)*
- La cadena "Ninguno" que aparece en la página resumen del instalador no es modificable en inglés. *(Fallo 6593089)*
- La página de copyright sólo aparece traducida para la configuración regional francesa. *(Fallo 6590992)*
- Cuando el instalador se ejecuta en una configuración regional alemana, la pantalla de Bienvenida no muestra el texto completo que se ve en otras configuraciones. *(Fallo 6592666)*
- La cadena "Inicio de instalación" que se ve en la página del instalador del mismo nombre no está traducida. Aparece en inglés aunque el instalador se haya ejecutado en configuraciones regionales de lengua no inglesa. *(Fallo 6592491)*
- Cuando el instalador se ejecuta en modo texto (`installer –t`), aparecen las opciones de respuesta inglesas "Yes" y "No", aunque el instalador se haya ejecutado en otra configuración regional. *(Error 6593230)*
- La información sobre herramientas del botón Examinar de la pantalla de selección de JDK del instalador no es modificable en inglés. *(Fallo 6593085)*

Opción de contraseña desaprobada

En versiones anteriores de Message Queue, podía utilizarse la opción `–p` o `–password` para especificar interactivamente una contraseña para los siguientes comandos: `imqcmd`, `imqbrokerd` y `imdbmgr`. Al comenzar con la versión 4.0., estas opciones se han desaprobado. Debe facilitar las contraseñas de la siguiente manera.

1. Establezca el valor que desee para la propiedad de la contraseña en un archivo utilizado para guardar sólo contraseñas.

   Utilice la siguiente sintaxis para especificar contraseñas en el archivo de la contraseña.

   ContraseñaPropiedadNombre= MiContraseña
2. Transmita el nombre del archivo de la contraseña con la opción `–passfile` .

Una nueva contraseña puede contener una o varias de las contraseñas enumeradas más abajo.

- Una contraseña keystore utilizada para abrir el keystore de SSL. Utilice la propiedad `imq.keystore.password` para especificar la contraseña.
- Una contraseña de depósito LDAP utilizada para conectar de forma segura con un directorio LDAP si la conexión no es anónima. Utilice la propiedad `imq.user_repository.ldap.password` para especificar esta contraseña.
- Una contraseña de base de datos JDBC utilizada para conectarse con la base de datos compatible con JDBC. Utilice la propiedad `imq.persist.jdbc.vendorName.password` para especificar esta contraseña. Este componente de *NombreDel Proveedor* del nombre de la propiedad es una variable que especifica el proveedor de la base de datos. Las opciones son `hadb`, `derby`, `pointbase`, `oracle` o `mysql`.
- Una contraseña del comando `imqcmd` (para realizar tareas administrativas del agente). Utilice la propiedad `imq.imqcmd.password` para especificar esta contraseña.

En el siguiente ejemplo, la contraseña de la base de datos JDBC se ha establecido como `abracadabra`.

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

Puede configurar el agente para que utilice el archivo de contraseña creado por usted de las siguientes formas:

- Establezca las siguientes propiedades en el archivo `config.properties` del agente.

  ```
  imq.passfile.enabled=true
  imq.passfile.dirpath=MiDirectorioDeArchivo
  imq.passfile.name=NombreDeMiArchivoContraseñas
  ```

- Utilice la opción `–passfile` del comando `imqbrokerd`.

  `imqbrokerd –passfile` *NombreDeMiArchivoContraseñas*

Problemas generales

Esta sección trata de los problemas generales de Message Queue 4.1. Algunos de ellos se introdujeron ya en versiones anteriores de Message Queue.

- Cuando un cliente JMS que utiliza transporte HTTP finaliza de forma abrupta (por ejemplo por el uso de `Ctrl-C`), el agente tarda aproximadamente un minuto a liberar la conexión del cliente y todos los recursos asociados.

  Si se inicia otra instancia del cliente dentro de este periodo de un minuto y si ésta intenta utilizar el mismo ClientID, suscripción duradera o cola, es posible que obtenga la excepción "ID de cliente ya en uso". No se trata de un problema real, es simplemente un efecto secundario del proceso de finalización descrito anteriormente. Si el cliente se inicia después de un retraso de aproximadamente un minuto, todo debería funcionar correctamente.

- Clientes de SOAP. Antes, la implementación jar de SAAJ 1.2 solía referirse a `mail.jar` y `mail.jar` no necesitaba estar en `CLASSPATH`. En SAAJ 1.3 ya no existe esta referencia; por lo que los clientes de Message Queue deben poner `mail.jar` explícitamente en `CLASSPATH`.

Problemas de administración y configuración

Los siguientes problemas son generados por la administración y la configuración de Message Queue

- The `imqadmin` and `imqobjmgr` utilities throw an error when the `CLASSPATH` contains double quotes on Windows machines (*Bug ID 5060769*).

  Solución alternativa Puede hacer caso omiso de este mensaje de error; el agente se encargará de notificar correctamente a los usuarios de cualquier error. Este error no afecta a la fiabilidad del sistema.

- La opción `-javahome` en todos los scripts de Solaris y Windows no funciona si el valor proporcionado contiene un espacio (*Bug ID 4683029*).

  La opción `javahome` es utilizada por las órdenes y utilidades de Message Queue para especificar un entorno de tiempo de ejecución alternativo compatible con Java 2. Sin embargo, el nombre de la ruta del entorno de tiempo de ejecución de Java alternativo no debe contener espacios. A continuación le mostramos ejemplos de rutas que incluyen espacios.

  Windows: `C:/jdk 1.4`

  Solaris: `/work/java 1.4`

  Solución temporal Instale el entorno de tiempo de ejecución de Java en un lugar cuya ruta no contenga espacios.

- El atributo `imqQueueBrowserMaxMessagesPerRetrieve` especifica el número máximo de mensajes que el tiempo de ejecución del cliente recupera a la vez cuando examina el contenido de una cola. Tenga en cuenta que la aplicación del cliente siempre obtendrá todos los mensajes de la cola. El atributo `imqQueueBrowserMaxMessagesPerRetrieve` afecta la forma en la que los mensajes de la cola son empaquetados para entregarlos al tiempo de ejecución del cliente (menos paquetes grandes o más paquetes pequeños), pero no afecta la cantidad total de mensajes examinados. Si cambia el valor de este atributo, podría empeorar el rendimiento, pero no influirá en el número de datos que obtendrá la aplicación del cliente (*Fallo ID 6387631*).

Problemas de agentes

Los problemas siguientes afectan al agente de Message Queue.

- Ha habido alguna confusión sobre cómo configurar el agente para la entrega round-robin (operación por turnos). Esta solución es sencilla y configurable.

1. Establezca el atributo `maxNumActiveConsumers` del destino en -1. De esta forma, se activa la entrega round-robin.
2. Establezca el atributo `consumerFlowLimit` del destino en 1. Esto especifica el número de mensajes entregados a un solo consumidor antes de que la entrega pase al siguiente. Si desea un empaquetado distinto, establezca este atributo en el valor que desee. Por defecto, se entregan a cada consumidor cien mensajes.

- No se puede tener acceso al agente cuando un almacén persistente abre demasiados destinos (*Bug ID 4953354*).

  Solución temporal Esta condición está causada por el hecho de que el agente ha llegado al límite de archivos abiertos del descriptor del sistema. En Solaris y Linux utilice la orden `ulimit` para incrementar el límite del descriptor de archivos.

- Los consumidores se quedan sin referencia cuando se destruye un destino (*Bug ID 5060787*).

  Los consumidores activos se quedan sin referencia cuando se destruye un destino. En ese caso, dejan de recibir mensajes (aunque se vuelva a crear el destino).

  Solución temporal No existe ninguna solución temporal para este problema.

Clústeres de agente

Estos son los problemas que afectan a los agentes de clúster.

- Sólo se admiten en esta versión los clústeres de agentes totalmente conectados. Esto significa que todos los agentes de un clúster deben establecer comunicación directamente con todos los demás agentes del clúster. Si se conecta a agentes mediante el argumento de orden `imqbrokerd -cluster`, asegúrese de que se incluyen todos los agentes del clúster.
- Un agente que utilice HADB no puede gestionar mensajes superiores a 10 MB. *(Fallo 6531734)*
- Si un cliente está conectado a un agente de alta disponibilidad, el tiempo de ejecución del cliente intentará volver a establecer la conexión hasta que lo consiga (con independencia de cuál sea el valor de `imqAddressListIterations`).)
- Un cliente conectado a un agente que es parte de un clúster por ahora no puede utilizar `QueueBrowser` para ver las colas que están ubicadas en agentes remotos del clúster. El cliente sólo puede examinar el contenido de las colas que se encuentran en el agente al que está directamente conectado. El cliente puede continuar enviando mensajes a cualquier cola o consumiendo mensajes desde una cola de cualquier agente del clúster; esta limitación sólo afecta a las funciones de exploración.
- En un clúster convencional, si desea "clustear" un agente de 4.1 con un agente 3.x, debe establecer la propiedad `imq.autocreate.queue.maxNumActiveConsumers=1` del clúster 4.1. De lo contrario, los agentes no serán capaces de establecer una conexión de clúster.

- Al convertir un clúster de alta disponibilidad, puede utilizar la utilidad Message Queue Manager (imqdbmgr) para convertir un almacén de datos persistentes de un HADB independiente existente en un almacén HADB. El comando es el siguiente:

  ```
  imqdbmgr upgrade hastore
  ```

  Puede utilizar esta utilidad en los siguientes casos:

  - Para mover un almacén HADB independiente 4.0 a otro compatido 4.1. En este caso, el agente actualizará automáticamente el almacén. Puede ejecutar el comando `imqdbmgr` para convertir el almacén de datos actualizado para uso compartido.
  - Para mover un almacén HADB independiente 4.1 a otro compatido. En este caso, sólo tendrá que ejecutar el comando `imqdbmgr` de arriba para convertir el almacén de datos para uso compartido.

  Dado que este comando sólo admite la conversión de los almacenes HADB, no es posible utilizarlo para convertir los almacenes basados en archivos u otros almacenes de JDBC a un almacén HADB compartido. Si ejecutaba antes una versión 3.x de Message Queue, deberá crear un almacén HADB y después migrar manualmente sus datos a ese almacén para utilizar la función de alta disponibilidad.

- La conversión a un almacén HADB con este comando `imqdbmgr upgrade hastore` puede fallar con el mensaje "hay demasiados bloqueos establecidos" si el almacén contiene más de 10.000 mensajes. (*Fallo ID 6588856)*).

  (*Solución alternativa*) Utilice el siguiente comando para aumentar el número de bloqueos.

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

  Para más información, consulte "Problemas de HADB" en la *Guía de resolución de problemas de Sun Java System Application Server 9.1 Enterprise Edition*.

- Si se confirman más de 500 mensajes remotos en una transacción, el agente podría devolver el error "HADB-E-12815: Agotado el espacio de memoria de la tabla." *(Fallo ID 6550483)*

  Para más información, consulte "Problemas de HADB" en la *Guía de resolución de problemas de Sun Java System Application Server 9.1 Enterprise Edition*.

- En un clúster de agente, el agente coloca mensajes en la cola para una conexión remota que no se ha iniciado (*Fallo ID 4951010*).

  *Solución alternativa*El consumidor recibirá los mensajes cuando se establezca la conexión. Los mensajes se enviarán a otro consumidro si se cierra la conexión del consumidor.

- Al consumir más de un mensaje de un agente remoto de una transacción, es posible que se registre en el agente el siguiente mensaje de error. El mensaje es benigno y puede omitirse:

```
[26/Jul/2007:13:18:27 PDT] WARNING [B2117]:
Error al reconocer el mensaje de
mq://129.145.130.95:7677/?instName=a&brokerSessionUID=3209681167602264320:
    ackStatus = NOT_FOUND(404)\
    Motivo = Actualización del estado de la transacción remota a CONFIRMADO(6):
transacción 3534784765719091968 no encontrada, la transacción
```

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

[26/Jul/2007:13:18:27 PDT] AVISO Notifi. conf. transacc.
[8-129.145.130.95(95:fd:93:91:ec:a0)-33220-1185481094690,
[consumer:3534784765719133952, type=NONE]]
TUID=3534784765719091968 got response:
com.sun.messaging.jmq.jmsserver.util.BrokerException:
Actualizar estado de transacción remota a CONFIRMADO(6):
  transacción 3534784765719091968 no encontrada, la transacción
puede haber sido ya confirmada:
com.sun.messaging.jmq.jmsserver.util.BrokerException: Actualizar estado de
transacción remota a CONFIRMADO(6): transacción 3534784765719091968 no encontrada, la transacción
puede haber sido ya confirmada.r
```

El mensaje se registra al notificar la confirmación al agente de inicio del mensaje para posteriores mensajes de la transacción cuando la propiedad `imq.txn.reapLimit` es baja comparada con el número de mesajes remotos de una transacción. *(Fallo 6585449)*

Solución alternativa Para evitar este mensaje, aumente el valor de la propiedad `imq.txn.reapLimit`.

Problemas con JMX

En la plataforma Windows, el método `getTransactionInfo` de Transaction Manager Monitor MBean devuelve información de la transacción con un tiempo de creación de transacción incorrecto (*Fallo ID 6393359*).

Solución alternativa Utilice el método `getTransactionInfoByID` de Transaction Manager Monitor MBean en su lugar.

Compatibilidad para SOAP

Es necesario que tenga en cuenta dos cuestiones relacionadas con la compatibilidad con SOAP

- Al comenzar con esta edición de la versión 4.0 de Message Queue, la compatibilidad con los objetos administrados por SOAP era discontinua.
- El desarrollo de SOAP depende de varios archivos: `SUNWjaf`, `SUNWjmail`, `SUNWxsrt` y `SUNWjaxp`. En la versión 4.1 de Message Queue, estos archivos sólo están disponibles si ejecuta Message Queue con JDK versión 1.6.0 ó superior.

Archivos que se pueden distribuir

Sun Java System Message Queue 4.1 contiene el siguiente conjunto de archivos que puede utilizar y distribuir libremente un formato binario:

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

Además, puede también redistribuir los archivos `LICENSE` y `COPYRIGHT`.

Funciones de accesibilidad para usuarios con discapacidades

Para obtener funciones de accesibilidad que han aparecido desde la publicación de este software, consulte las evaluaciones de producto de la Sección 508 (se pueden pedir a Sun) para determinar qué versiones son las mejores para utilizar soluciones accesibles. Puede encontrar versiones actualizadas de las aplicaciones en `http://sun.com/software/javaenterprisesystem/get.html`.

Para obtener información sobre el compromiso de Sun con la accesibilidad, visite `http://sun.com/access`.

Información sobre problemas y respuestas de los clientes

Si experimenta problemas con Sun Java System Message Queue, póngase en contacto con el servicio de atención al cliente de Sun usando uno de estos procedimientos:

- Servicios en línea de asistencia técnica de software de Sun en `http://www.sun.com/service/sunone/software`

  Este sitio dispone de vínculos a la base de datos de soluciones, al centro de asistencia en línea y al rastreador de productos, así como a programas de mantenimiento y números de contacto de asistencia técnica.
- El número de teléfono del distribuidor asociado al contrato de mantenimiento.

Para que podamos ayudarle de forma óptima en la resolución de problemas, tenga a mano la siguiente información cuando se ponga en contacto con el servicio de asistencia técnica:

- Descripción del problema, incluida la situación en la que éste se produce y la forma en que afecta al funcionamiento.
- Tipo de equipo, versión del sistema operativo y versión del producto, incluida cualquier revisión del producto y otro software que pudiera influir en el problema.

- Pasos detallados de los métodos que haya usado para reproducir el problema.
- Cualquier registro de errores o volcados del núcleo.

Sun Java System Software Forum

Existe un foro de Sun Java System Message Queue en la ubicación siguiente:

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

Apreciamos su participación.

Foro de tecnología de Java

Existe un foro de JMS en los foros de tecnología de Java que puede interesarle.

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

Sun valora sus comentarios

Deseamos mejorar nuestra documentación y agradecemos sus comentarios y sugerencias.

Para compartir con nosotros sus comentarios, vaya a `http://docs.sun.com` y haga clic en Enviar comentarios (Send Comments). Se mostrará un formulario en línea en el que deberá indicar el título del documento y el número de referencia. El número de referencia consta de siete o de nueve dígitos y se encuentra en la página que contiene el título de la guía o al principio del documento. Por ejemplo, el título de este libro es Notas de la versión de Sun Java System Message Queue 4.1, y el nombre de la pieza 820-3189-10.

Recursos adicionales de Sun

Puede encontrar información útil de Sun Java System en las siguientes ubicaciones de Internet:

- Documentación

  `http://docs.sun.com/prod/java.sys`
- Servicios profesionales

  `http://www.sun.com/service/sunps/sunone`
- Servicio y productos de software de

  `http://www.sun.com/software`
- Servicios de asistencia técnica para software

http://www.sun.com/service/sunone/software

- Base de datos de soluciones y asistencia al cliente de
  http://www.sun.com/service/support/software
- Servicios de formación y asistencia al cliente de Sun
  http://training.sun.com
- Servicios profesionales y de consultoría de
  http://www.sun.com/service/sunps/sunone
- Información para programadores
  http://developers.sun.com
- Servicios de asistencia para programadores de Sun
  http://www.sun.com/developers/support
- Formación sobre el software de
  http://www.sun.com/software/training