MQSeries® Adapter Kernel for Multiplatforms

Guía rápida de iniciación

Versión 1 Release 1

GC10-3637-01

MQSeries® Adapter Kernel for Multiplatforms

Guía rápida de iniciación

Versión 1 Release 1

GC10-3637-01

Aviso: antes de utilizar esta información y el producto al que da soporte, lea la información del apartado "Avisos" en la página 127.

Segunda edición (abril de 2001)

Esta edición se aplica a la versión 1, release 1, nivel de modificación 1 de MQSeries Adapter Kernel for Multiplatforms (número de producto 5648-D75) y a todos los releases y modificaciones posteriores hasta que se indique lo contrario en nuevas ediciones.

IBM agradece sus comentarios. Puede hacer los comentarios que desee relacionados con esta información, enviando un mensaje de correo electrónico a la dirección siguiente: idrcf@hursley.ibm.com.

Al enviar información a IBM, se concede a IBM un derecho no exclusivo de utilización o distribución de la misma en la forma que considere adecuada y sin incurrir por ello en ninguna obligación con el remitente.

© Copyright International Business Machines Corporation 2000, 2001. Reservados todos los derechos.

Contenido

© Copyright IBM Corp. 2000, 2001

Figuras

© Copyright IBM Corp. 2000, 2001

Tablas

© Copyright IBM Corp. 2000, 2001

Bienvenidos a MQSeries Adapter Kernel: Guía rápida de iniciación

En este documento se describe MQSeries® Adapter Kernel y se explica cómo planificarlo, instalarlo y utilizarlo.

Para preparar el kernel para su utilización, lleve a cabo los pasos generales siguientes:

1. Lea el "Capítulo 1. Acerca de la Oferta MQSeries Adapter" en la página 1.
2. Lleve a cabo la preparación para la instalación. Si desea obtener más información, consulte el apartado "Preparación para la instalación" en la página 40.
3. Instale el kernel. Si desea obtener más información, consulte el apartado "Instalación del kernel" en la página 41.
4. Verifique la instalación. Si desea obtener más información, consulte el apartado "Verificación de la instalación" en la página 47.
5. Configure el kernel. Si desea obtener más información, consulte el apartado "Configuración del kernel" en la página 58.
6. Si lo desea, configure software opcional para que funcione con el kernel. Si desea obtener más información, consulte el apartado "Configuración de MQSeries y MQSeries Integrator" en la página 92.
7. Genere los adaptadores utilizando MQSeries Adapter Builder y, a continuación, pruébelos y difúndalos. Para obtener información más detallada, consulte la documentación de MQSeries Adapter Builder.
8. Inicie el kernel. Si desea obtener más información, consulte el apartado "Inicio del kernel" en la página 92.

Para utilizar esta información, también debe conocer los productos de requisito previo y opcionales. Consulte el "Capítulo 2. Planificación de la instalación del kernel" en la página 31 y el apartado "Consultas" en la página 101.

A quién va dirigida esta información

Esta información va dirigida a quienes necesitan planificar, instalar o utilizar MQSeries Adapter Kernel.

Información relacionada

Para obtener más información, consulte lo siguiente:

© Copyright IBM Corp. 2000, 2001

- El archivo `readme.txt`. Es posible que este archivo contenga información nueva, disponible después de finalizar esta publicación. Antes de la instalación, el archivo `readme.txt` está situado en el directorio raíz del CD-ROM del producto. Después de la instalación, el archivo `readme.txt` está situado en el directorio raíz de la instalación de MQSeries Adapter Kernel.
- La publicación *Problem Determination Guide*, número GC34-5897, en la que se describen herramientas, entre las que se incluye el rastreo, para resolver problemas específicos con MQSeries Adapter Kernel. La publicación *Problem Determination Guide* está disponible en el Centro de información de MQSeries Adapter Kernel, que se instala con el producto.
- La documentación en línea de API (interfaz de programación de aplicaciones) que se facilita en el Centro de información de MQSeries Adapter Kernel. Esta información sólo se proporciona como ayuda para comprender el funcionamiento del kernel y para los diagnósticos. Consulte el "Capítulo 5. Utilización de API de MQSeries Adapter Kernel" en la página 99.
- Información sobre MQSeries Adapter Builder, incluidas las publicaciones y el sistema de ayuda.
- El sitio web de la familia de productos MQSeries que se encuentra en www.ibm.com/software/ts/mqseries/.

  Siguiendo los enlaces de este sitio web, puede:

  – Obtener la información más actualizada acerca de la familia de productos MQSeries, entre la que se incluye la Oferta MQSeries Adapter.

  – Acceder a las publicaciones de MQSeries en formatos HTML y PDF, que pueden incluir una edición más reciente de esta publicación.

  – Bajar SupportPacs de MQSeries.

Convenios tipográficos

En la documentación de MQSeries Adapter Kernel se utilizan los convenios tipográficos y de tecleo siguientes.

Tabla 1. Convenios utilizados en esta publicación

Convenio	Significado
Negrita	Indica nombres de mandatos. Cuando hace referencia a interfaces gráficas de usuario (GUI), indica menús, elementos de menú, etiquetas y botones.
`Monoespaciado`	Indica el texto que debe entrar en un indicador de mandatos y los valores que debe utilizar exactamente tal como se especifica, por ejemplo, nombres de archivo y vías de acceso, así como elementos de lenguajes de programación, por ejemplo, funciones, clases y métodos. El Monoespaciado también indica el texto de las pantallas y los ejemplos de código.
Cursiva	Indica los valores de variables que deben proporcionarse (por ejemplo, al suministrar el nombre de un archivo en *NombreArchivo*). La cursiva también se utiliza para enfatizar y para indicar los títulos de las publicaciones.
`%`	Representa el indicador de shell de mandatos de UNIX para un mandato que no requiere privilegios de root.
`#`	Representa el indicador de shell de mandatos de UNIX para un mandato que requiere privilegios de root.
`C:\>`	Representa los indicadores de mandatos en los sistemas Windows®.
>	Cuando se utiliza para describir un menú, muestra una serie de selecciones del menú. Por ejemplo, "Pulse **Archivo > Nuevo**" significa "En el menú **Archivo**, pulse el mandato **Nuevo**."
Entrar mandatos	Cuando se le pida que "entre" o "especifique" un mandato, escriba el mandato y pulse la tecla Intro. Por ejemplo, la instrucción "Entre el mandato **ls**" significa que debe escribir **ls** en un indicador de mandatos y pulsar la tecla Intro.
[ ]	Se utiliza para delimitar los elementos opcionales en descripciones de sintaxis.
{ }	Se utiliza para delimitar listas de las que debe seleccionar un elemento en las descripciones de sintaxis.
|	Separa elementos de una lista de opciones entre llaves ({ }) en las descripciones de sintaxis.
...	Los puntos suspensivos en las descripciones de sintaxis indican que puede repetir una o varias veces el elemento anterior. Los puntos suspensivos en los ejemplos indican que se ha omitido información del ejemplo para mayor brevedad.

© Copyright IBM Corp. 2000, 2001

Nota: El término Epic aparece en determinados valores y nombres del software del kernel y en esta publicación. Con respecto a la Oferta MQSeries Adapter, este término no tiene ningún significado en sí mismo.

Resumen de cambios

La sexta edición (la edición actual) de la versión en inglés incluye las adiciones y modificaciones siguientes con respecto a la quinta edición:

- Actualización del tema sobre el flujo de ejecución para reflejar varios cambios. Consulte el apartado "Flujo de ejecución" en la página 14.
- Información sobre la utilización de MQSeries Adapter Kernel con WebSphere® Business Integrator. Para obtener información más detallada, consulte el apartado "Utilización del MQSeries Adapter Kernel con WebSphere Business Integrator y WebSphere Application Server" en la página 29.
- Información sobre el nivel de soporte al idioma nacional que se proporciona con diferentes tipos de adaptadores. Para obtener información más detallada, consulte el apartado "Soporte del idioma nacional" en la página 30.
- Aclaración de las instrucciones de instalación. Consulte el apartado "Instalación del kernel" en la página 41.
- Información sobre la instalación silenciosa. Para obtener información más detallada, consulte el apartado "Utilización de la instalación silenciosa" en la página 52.
- Una visión general conceptual de la configuración para ayudarle a configurar el kernel. Para obtener información más detallada, consulte el apartado "Visión general de la configuración" en la página 58.
- Información sobre los nuevos valores de cabecera. Para obtener información más detallada, consulte el apartado "Cabecera del descriptor de mensaje de MQSeries Adapter Kernel" en la página 111.

La quinta edición de la versión en inglés incluye las adiciones y modificaciones siguientes con respecto a la cuarta edición:

- Información acerca del uso del kernel en las plataformas Windows® 2000, OS/400®, HP-UX y Solaris. El soporte para estas plataformas era nuevo en MQSeries Adapter Kernel versión 1.1. Anteriormente, el kernel sólo estaba disponible en Windows NT® y AIX®.
- Actualizaciones de todas las instrucciones de instalación para reflejar MQSeries Adapter Kernel, versión 1.1.
- Información acerca de la utilización del archivo `aqmconfig.xml` para configurar MQSeries Adapter Kernel. Anteriormente, el kernel se configuraba con el archivo `aqmconfig.properties`. Para obtener información más detallada, consulte el apartado "El archivo de configuración" en la página 65.

© Copyright IBM Corp. 2000, 2001

- Información acerca de las nuevas modalidades de comunicación MQ y JMS (Java Message Service). Para obtener información más detallada, consulte el "Apéndice A. Modalidades de comunicación" en la página 103.
- La información acerca del rastreo se ha trasladado de esta publicación al nuevo documento titulado *Problem Determination Guide*. Para obtener más información, consulte la publicación *Problem Determination Guide*.

Capítulo 1. Acerca de la Oferta MQSeries Adapter

IBM MQSeries Adapter Kernel forma parte de un conjunto de productos de integración de aplicaciones que, juntos, se denominan Oferta IBM MQSeries Adapter. La Oferta MQSeries Adapter funciona con los servicios de mensajería de MQSeries y con otros servicios de envío de mensajes con la finalidad de reducir el riesgo, la complejidad y el coste de gestión de la integración punto a punto de los procesos de negocio.

En la integración *punto a punto*, cada aplicación se comunica de modo individual con cada una de las demás aplicaciones. Todas las interfaces son diferentes y hay muchas interfaces distintas. Generalmente, un cambio en una aplicación requiere modificaciones en muchas interfaces. A medida que aumenta el número de aplicaciones, también se incrementa rápidamente el coste de la integración punto a punto. Por lo general, la integración de cada nueva aplicación requiere más trabajo que la integración de la anterior.

Con la Oferta MQSeries Adapter, puede migrar de la utilización de la integración punto a punto a la integración *múltiple*. La integración múltiple ofrece muchas ventajas, entre las que se encuentran:

- Todas las aplicaciones pueden utilizar una interfaz común.
- Los datos de una *aplicación de origen*, en formato de *mensaje*, se *direccionan* a una o más *aplicaciones de destino*.
- Generalmente, un cambio en una aplicación sólo afecta a la interfaz específica.
- Utilizar una interfaz común, neutra de la aplicación, por ejemplo, un estándar del sector como XML (Extensible Markup Language), puede ser incluso más rentable. Se puede ofrecer soporte para más aplicaciones con un esfuerzo menor.
- La integración múltiple puede ser incluso más rentable a medida que aumenta el número de aplicaciones. Generalmente, al añadir cada nueva aplicación no es necesario realizar cambios importantes en las interfaces del resto de aplicaciones.
- El trabajo de la integración se puede automatizar y basar en plantillas.

La Oferta MQSeries Adapter se puede difundir sin necesidad de realizar cambios en las aplicaciones o en los procesos de negocio. Normalmente, todo el trabajo de integración se realiza en la Oferta MQSeries Adapter, por lo que ya no es necesario crear código personalizado.

© Copyright IBM Corp. 2000, 2001

En la Oferta MQSeries Adapter, un *adaptador* proporciona la interfaz para o desde una aplicación. Todas las aplicaciones necesitan al menos un adaptador para facilitar la interfaz entre el entorno de aplicaciones y el entorno de mensajería. Cada adaptador es específico de una aplicación y un tipo de mensaje.

Si se desea, MQSeries Adapter Kernel se puede difundir con MQSeries Integrator para llevar a cabo la intermediación y la transformación de mensajes. La Oferta MQSeries Adapter se puede complementar con las ofertas de servicio de IBM y de otros proveedores.

Entre los usos de ejemplo de los adaptadores se incluyen los siguientes:

- Añadir un pedido de venta.
- Sincronizar un registro de cliente.
- Sincronizar un registro de inventario.
- Sincronizar un elemento.
- Sincronizar un pedido de venta.

Generación y ejecución

La Oferta MQSeries Adapter consta de dos componentes primarios, Adapter Builder (también denominado generador) y Adapter Kernel (también denominado kernel). En este apartado se describen estos componentes, así como los adaptadores que genera y ejecuta la Oferta Adapter.

adaptador

Software que proporciona una interfaz para o desde una aplicación. Los adaptadores se generan utilizando MQSeries Adapter Builder. Generalmente, cada adaptador se genera para que sea específico para un tipo de mensaje que se envía a o desde una aplicación. Los adaptadores en sí mismos no forman parte de la Oferta MQSeries Adapter.

Un adaptador consta de código fuente C o Java™ que se compila en una biblioteca compartida. Cuando se ejecutan al mismo tiempo los adaptadores y MQSeries Adapter Kernel, realizan la función de ejecución de la Oferta MQSeries Adapter.

Dependiendo del modo en que se ha modelado en MQSeries Adapter Builder, el adaptador puede contener una gran variedad de funciones como, por ejemplo, flujo de control; flujo de datos; navegación secuencial; ramificación condicional, que incluye decisión e iteración; escritura de datos; almacenamiento de contexto de datos; transformación de elementos de datos; control transaccional; operaciones lógicas y código personalizado.

Los adaptadores se pueden volver a utilizar.

Hay dos tipos principales de adaptadores:

- Adaptadores origen, para las aplicaciones que envían los datos.
- Adaptadores destino, para las aplicaciones que reciben los datos.

El envío de un tipo de mensaje de una aplicación a una segunda aplicación requiere, por lo general, un adaptador origen y un adaptador destino. Si la segunda aplicación debe enviar un tipo de mensaje a la primera aplicación, se necesita otro adaptador origen y otro adaptador destino. Así, para enviar un tipo de mensaje de la primera aplicación a la segunda y, a continuación, para devolver otro tipo de mensaje de la segunda aplicación a la primera, generalmente se deben difundir cuatro adaptadores.

Se necesita un adaptador independiente para cada tipo de mensaje.

Se utiliza un tercer tipo de adaptador, el adaptador bean de sesión de servicio Java, cuando IBM WebSphere Application Server y los Enterprise Beans se utilizan en la parte destino del kernel. La implementación de WebSphere Application Server de la especificación Sun Microsystems Enterprise JavaBeans (EJB) permite utilizar los adaptadores bean de sesión de servicio Java y otros Enterprise Beans. Consulte el apartado "Utilización del MQSeries Adapter Kernel con WebSphere Business Integrator y WebSphere Application Server" en la página 29 y la documentación de MQSeries Adapter Builder si desea obtener más información.

MQSeries Adapter Builder
Interfaz gráfica de usuario (GUI) que permite generar un adaptador prácticamente para cualquier aplicación. La interfaz de usuario es similar a la interfaz de usuario de MQSeries Integrator. Si desea obtener más información, consulte el Centro de información de MQSeries Adapter Builder.

MQSeries Adapter Kernel
Conjunto de API (interfaces de programación de aplicaciones), numerosos programas ejecutables en C y Java, y varios archivos de configuración. El kernel permite difundir y ejecutar adaptadores. Además de ofrecer soporte directamente para los adaptadores, el kernel realiza funciones relacionadas, entre las que se incluyen el direccionamiento simple de mensajes. También ofrece control transaccional, rastreo e interfaz con MQSeries u otro software de mensajería.

El kernel se instala en todos los sistemas en los que se ejecuta un adaptador origen o un adaptador destino.

Con la Oferta MQSeries Adapter, los procesos de negocio y cada una de las aplicaciones pueden permanecer aislados de los datos específicos de middleware, de los detalles del mensaje y de otras aplicaciones. Una interfaz común para mensajería permite añadir nuevas aplicaciones sin necesidad de cambiar las aplicaciones existentes o los procesos de negocio.

MQSeries Adapter Kernel se puede difundir en dos capas. Una capa es la parte origen de la ejecución y la otra es la parte destino de la ejecución. La difusión en dos capas posibilita un funcionamiento eficaz y permite reducir la actividad general administrativa. No es necesario que una tercera capa resida entre las dos partes de la ejecución para direccionamiento y entrega. No obstante, se puede añadir MQSeries Integrator para realizar la intermediación como, por ejemplo, para direccionamientos complejos, transformación y mediación de datos.

Excepto cuando se especifique de otro modo, el resto de este documento hace referencia a MQSeries Adapter Kernel. Para obtener información más detallada acerca de MQSeries Adapter Builder, consulte el Centro de información del producto.

Acerca del kernel

En el sentido más simple, la ejecución, es decir, el kernel y los adaptadores que se generan, tiene las finalidades siguientes:

1. Transferir datos de una aplicación de origen a una aplicación de destino.
2. Convertir los datos de la aplicación de origen en un mensaje, generalmente, en un formato neutro de aplicación, que se direcciona a través del kernel utilizando MQSeries u otro software de mensajería.
3. Direccionar el mensaje a la aplicación de destino.
4. Determinar el modo de hacer llegar los datos a la aplicación de destino.
5. Convertir los datos del formato del mensaje que se direcciona a través del kernel mediante un adaptador al formato de la aplicación de destino.

En este apartado, se trata la funcionalidad del kernel a alto nivel. Para obtener información más detallada acerca de las funciones, consulte el apartado “Flujo de ejecución” en la página 14.

El kernel consta de dos partes:

- La *parte origen*, que se inicia cuando se recibe el mensaje de la aplicación origen y finaliza cuando se coloca el mensaje en una cola de mensajes.
- La *parte destino*, que se inicia cuando se recupera el mensaje de la cola de mensajes y finaliza cuando se envía el mensaje al destino.

Generalmente, cada parte reside en un sistema diferente, pero pueden residir ambas en el mismo sistema.

Consulte la Figura 1 en la página 6. Presenta la secuencia siguiente:

Parte origen del kernel

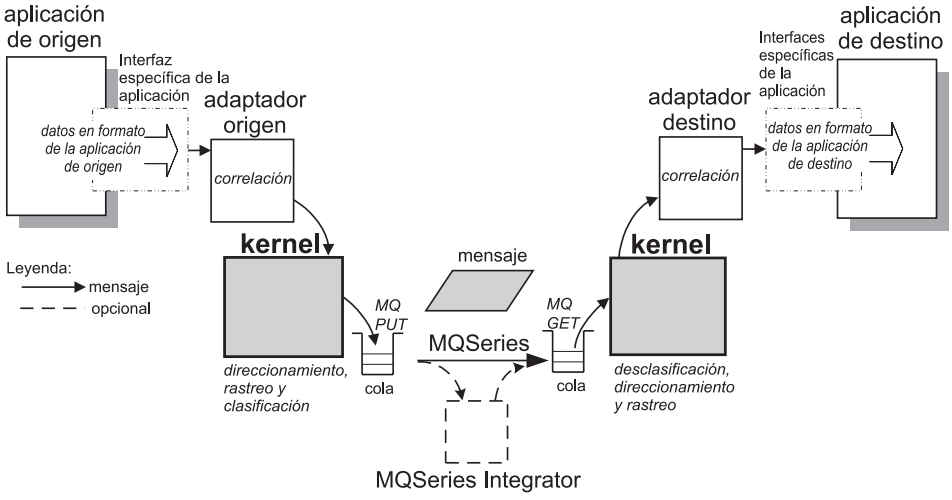
1. En la parte origen del kernel, la aplicación de origen envía los datos en el *formato de la aplicación de origen* utilizando una *interfaz específica de la aplicación* a un adaptador origen que se ha generado en MQSeries Adapter Builder. Se necesita un adaptador origen diferente para cada tipo de mensaje, por ejemplo, para "añadir un pedido de venta" o para "sincronizar un pedido de cliente".

   La interfaz específica de la aplicación se debe desarrollar fuera de la Oferta MQSeries Adapter. La naturaleza exacta de la interfaz específica de la aplicación depende de las características de la aplicación de origen o de la aplicación de destino. Algunos ejemplos pueden ser llamadas API y salidas de usuario, lecturas y grabaciones de archivo, activaciones de base de datos y colas de mensajes.

   Se debe tener en cuenta que el adaptador origen se ejecuta en el proceso de la aplicación de origen; por este motivo, se deben iniciar todos los daemons o servidores que contengan el adaptador origen para que éste funcione.
2. El adaptador origen lleva a cabo su función según el modo en que se ha generado. Una de sus funciones habituales consiste en transformar elementos de datos, es decir, correlacionar elementos del formato de la aplicación de origen con un *formato de datos de mensajería de integración* para los datos del cuerpo. Los datos del cuerpo y los metadatos adicionales que representan valores de control se colocan en un *objeto de soporte de mensajes* del kernel.
3. Cuando el adaptador origen pasa el objeto de soporte de mensajes al kernel utilizando el *adaptador nativo*, el kernel utiliza los valores de control del objeto de soporte de mensajes (*valores de control de mensaje*) para controlar la clasificación del objeto de soporte de mensajes en un formato de mensaje de comunicaciones, así como el direccionamiento de los mensajes de comunicación.

   Si el mensaje no contiene valores de control de mensaje determinados, el kernel puede utilizar los valores por omisión o los valores de control de mensaje obtenidos del archivo de configuración. En el apartado "Valores de control de mensaje" en la página 16 se incluyen las definiciones de los valores de control de mensaje.
4. El kernel realiza sus funciones, que incluyen *clasificación de mensajes*, *direccionamiento* simple y, si se desea, *rastreo*. Consulte los apartados "Mensaje y formato del mensaje" en la página 12, "Direccionamiento y entrega" en la página 13 y "Rastreo" en la página 29.

Figura 1. Visión general de la Oferta MQSeries Adapter.



Entrega de la parte origen a la parte destino del kernel

5. Utilizando su adaptador nativo, el kernel coloca el mensaje en la cola de mensajes adecuada.

   En la parte origen se utilizan dos métodos de envío (send):

   - `sendMsg`, que envía el mensaje y vuelve inmediatamente. El método `sendMsg` también se puede utilizar con los métodos `begin`, `commit` y `rollback` para enviar mensajes de modo *transaccional*, es decir, se pueden enviar mensajes si (y sólo si) otras operaciones han finalizado satisfactoriamente. Consulte el apartado "Posibilidades transaccionales" en la página 28 para obtener más información.
   - `sendRequestResponse`, que envía el mensaje y espera una respuesta. El método `sendRequestResponse` no se puede emitir transaccionalmente.

     Cuando el emisor solicita una respuesta, se utiliza un tercer método, `sendResponse`, en la parte destino.

   MQSeries u otro software de mensajería transporta el mensaje. Consulte el apartado "Función de MQSeries u otro software de mensajería" en la página 8. Se debe haber configurado previamente el software de mensajería para ofrecer soporte para la Oferta MQSeries Adapter.

   Además, si se ha configurado MQSeries Integrator en el kernel como destino, puede llevar a cabo funciones de intermediación. Consulte el apartado "Función de MQSeries Integrator" en la página 8. Si se ha configurado el destino final, una cola de mensaje, en los flujos de mensajes o las normas de MQSeries Integrator, éste envía el mensaje a la cola de mensajes.

El mensaje llega a la cola de mensajes adecuada.

Parte destino del kernel

6. En la parte destino del kernel, hay dos *modelos de entrega* posibles para la interfaz entre la ejecución y la aplicación de destino.
   - El modelo más común es el de *inserción*, en el que el kernel se encarga de iniciar y gestionar la entrega del mensaje a la aplicación de destino. Generalmente, este modelo no requiere que se realice ninguna modificación en la aplicación de destino para ofrecer soporte para la Oferta MQSeries Adapter.
   - En el modelo de *extracción*, la aplicación de destino se encarga de gestionar la recepción del mensaje. Este modelo requiere llevar a cabo modificaciones en la aplicación de destino para ofrecer soporte para la Oferta MQSeries Adapter. La aplicación de destino debe gestionar la interfaz del kernel para la aplicación de destino.

   En el modelo de inserción, se debe tener en cuenta que el usuario debe iniciar previamente los procesos del kernel de la parte de destino para obtener y entregar el mensaje. Consulte el apartado "Inicio del kernel" en la página 92.

   En el modelo de inserción, el kernel obtiene el mensaje de la cola de mensajes, realiza el rastreo, si está habilitado, y sigue direccionando el mensaje mediante la selección del adaptador destino adecuado. Por lo general, se necesita un adaptador destino diferente para cada tipo de mensaje.
7. El kernel entrega el mensaje al adaptador destino adecuado. El adaptador destino lleva a cabo las funciones que se le han incorporado. Una función habitual consiste en correlacionar elementos del formato de mensajería de integración con el *formato de la aplicación de destino*.

   Los adaptadores destino se pueden alojar en un daemon del adaptador de MQSeries Adapter Kernel o en WebSphere Application Server. En el apartado "Utilización del MQSeries Adapter Kernel con WebSphere Business Integrator y WebSphere Application Server" en la página 29 se trata este último con más detalle.
8. El adaptador destino envía los datos a la aplicación de destino en el formato de ésta utilizando una interfaz específica de la aplicación, desarrollada fuera de la Oferta MQSeries Adapter.
9. Cuando el adaptador destino entrega el mensaje, la cola de mensajes lo confirma. De este modo, se suprime el mensaje de la cola.
10. Si el adaptador origen tiene un valor de control de mensaje establecido para solicitar un acuse de recibo, el kernel entrega un acuse de recibo de la entrega del mensaje o una salida del adaptador destino al adaptador origen utilizando el método `sendResponse`.

11. En caso de que se produzca algún error, el kernel coloca el mensaje original en la cola de errores. Si el kernel no puede poner el mensaje original en la cola de errores, no se produce la confirmación.

Función de MQSeries u otro software de mensajería

Los mensajes de comunicación de la Oferta MQSeries Adapter se transportan a través de colas de mensajes. El software de mensajería como, por ejemplo, MQSeries o JMS (Java Message Service) proporciona las colas de mensajes. Los mensajes que transporta la Oferta MQSeries Adapter utilizan los tipos de colas siguientes:

- *Colas de recepción*, en la terminología de la Oferta MQSeries Adapter. Se utilizan como las colas principales de entrada para recibir mensajes. Puede haber varias colas de recepción por aplicación de destino.
- *Colas de error*, en la terminología de la Oferta MQSeries Adapter. Se utilizan cuando no se puede procesar un mensaje que se ha obtenido de una cola de recepción.
- Opcionalmente, *colas de respuesta*. Se utilizan con el método `sendRequestResponse`.

La Oferta MQSeries Adapter utiliza posibilidades específicas de MQSeries como, por ejemplo, los tipos de mensajes siguientes:

- Datagramas, que utilizan el método `sendMsg`.
- Peticiones, que utilizan el método `sendRequestResponse`.
- Respuestas, que utilizan los métodos `sendRequestResponse` y `sendResponse`.

Si se desea, MQSeries puede actuar como una interfaz específica de la aplicación.

Consulte el “Apéndice B. Configuraciones validadas” en la página 109, en el que se ha incluido una lista de las configuraciones validadas de MQSeries y la Oferta MQSeries Adapter. En el apartado “Software” en la página 32 puede encontrar una lista de las versiones de MQSeries y otros tipos de software para los que se ofrece soporte.

Función de MQSeries Integrator

Si se desea, MQSeries Integrator se puede difundir con MQSeries Adapter Kernel. Se puede utilizar para satisfacer muchos requisitos potenciales de intermediación:

- Direccionamiento complejo, es decir, el direccionamiento basado en el contenido de la cabecera o el cuerpo del mensaje. El direccionamiento puede cambiar dinámicamente a medida que se modifica el contenido del cuerpo del mensaje. Consulte el apartado “Direccionamiento y entrega” en la página 13 para obtener más información acerca del direccionamiento simple y complejo.

- Transformación de datos, es decir, cambiar a un tipo de documento diferente.
- Mediación de datos, es decir, cambiar el contenido del cuerpo del mensaje. Por ejemplo, si la aplicación de origen proporciona el valor `each` en un campo, pero la aplicación de destino espera que el valor de dicho campo sea `ea`, la mediación de datos sustituye el valor facilitado por el valor esperado.

Puede utilizar MQSeries Integrator para realizar la mayor parte del direccionamiento de su sitio y también puede utilizar en menor medida la funcionalidad de direccionamiento de MQSeries Adapter Kernel.

Consulte el "Apéndice B. Configuraciones validadas" en la página 109, en el que se ha incluido una lista de las configuraciones validadas de MQSeries Integrator y la Oferta MQSeries Adapter. En el apartado "Software" en la página 32 puede encontrar una lista de las versiones de MQSeries Integrator y otro software para el que se ofrece soporte.

Cómo funciona el kernel

En este apartado se tratan los puntos siguientes:

Componentes de la ejecución del kernel

Cuando se ejecutan conjuntamente los adaptadores que ha generado, el código personalizado que ha desarrollado y MQSeries Adapter Kernel, proporcionan la funcionalidad de la Oferta MQSeries Adapter.

Los componentes principales de la ejecución del kernel son los siguientes:

adaptador origen
: Software que se genera para una aplicación específica (generalmente, utilizando MQSeries Adapter Builder) con el fin de convertir los datos de dicha aplicación en un formato de mensajería de integración (datos del cuerpo). Por lo general, los adaptadores origen se ejecutan en la misma máquina que la aplicación de origen, dentro del proceso de la aplicación o como un proceso separado. Algunos ejemplos de datos de origen incluyen archivos, estructuras C y objetos Java. XML es un ejemplo de formato de mensajería de integración que, por lo general, cumple un estándar del sector, como puede ser OAG o RosettaNet.

soporte de mensajes
Contenedor para los metadatos que utiliza el kernel para encapsular el mensaje de integración y otros datos de control que usa el kernel. Entre los ejemplos de metadatos se incluyen los identificadores de aplicación (identificadores lógicos) de las aplicaciones de origen y de destino, la categoría del mensaje (por ejemplo, OAG), el tipo del mensaje (por ejemplo, ″Pedido de compra″) y el mensaje de comunicaciones (datos del cuerpo) que se envía o recibe.

adaptador nativo
Software que se utiliza para enviar y recibir objetos de soporte de mensajes. Al enviar mensajes, el adaptador nativo proporciona un direccionamiento simple de datos y la posibilidad de ofrecer soporte para uno o más mecanismos de transporte de comunicaciones. El direccionamiento simple de datos se basa en los metadatos del objeto de soporte de mensajes, por ejemplo, la categoría y el tipo de mensaje. Los mensajes se pueden enviar de modo síncrono o asíncrono. Si el mecanismo de transporte de comunicaciones subyacente da soporte a los mensajes transaccionales, los mensajes se pueden enviar bajo un control transaccional de una sola fase. El soporte transaccional se limita a las posibilidades del mecanismo de transporte utilizado. El objeto de soporte de mensajes se clasifica en el formato de mensaje de comunicaciones que utiliza el mecanismo de transporte. Cuando se recibe un mensaje de comunicaciones, el adaptador nativo desclasifica el mensaje en el objeto de soporte de mensajes.

daemon del adaptador
Proceso que convierte en instancia los trabajos del adaptador. Después de iniciarlo, el daemon del adaptador permanece activo. Cada aplicación de destino puede tener un daemon del adaptador para cada cola de recepción de la aplicación.

trabajo del adaptador
Proceso que entrega cada mensaje al adaptador destino adecuado. Cada trabajo gestiona un adaptador nativo. El daemon del adaptador crea e inicia los trabajos.

Disponer de varios trabajos permite la *entrega de mensajes de múltiples hebras* a los adaptadores destino. Cada trabajo, junto con su adaptador destino, puede manejar una hebra. Si sólo existe un trabajo, la entrega de mensajes al adaptador destino y, por consiguiente, a la aplicación de destino, es de una sola hebra.

Además de gestionar un adaptador nativo, el trabajo también se encarga de llevar a cabo las tareas siguientes:

- Convierte en instancia el cliente de rastreo, si se ha habilitado el rastreo.

- Convierte en instancia la clase de inicio de sesión adecuada para cada aplicación de destino.
- Selecciona el adaptador destino según el tipo y la categoría del cuerpo del mensaje.
- Envía el mensaje al adaptador destino seleccionado.
- Si no puede efectuar una confirmación, lleva a cabo una restitución, establece un indicador para todos los demás trabajos del daemon del adaptador y concluye, así como su adaptador nativo, lo que indica que el mensaje tiene algún problema. La conclusión de todos los trabajos evita que otros trabajos vuelvan a procesar el mismo mensaje que presenta el problema con el mismo resultado.
- Cuando reconoce el indicador que ha establecido otro trabajo para la conclusión, lleva a cabo su conclusión, así como la de su adaptador nativo.

adaptador destino
Software que se genera para una aplicación específica (generalmente, utilizando MQSeries Adapter Builder) para convertir datos de un formato de mensajería de integración (datos del cuerpo) en los tipos de datos que necesita una aplicación de destino. El adaptador destino invoca las API necesarias en la aplicación de destino para entregar el mensaje. Los adaptadores destino se ejecutan en la misma máquina que la aplicación o el cliente de la aplicación.

adaptador bean de sesión de servicio Java
Un tipo de adaptador EJB de lenguaje Java que se aloja en un servidor EJB como WebSphere Application Server.

componente de configuración
Datos que se utilizan para convertir identificadores lógicos en objetos como, por ejemplo, nombres de cola. Los datos de configuración se pueden especificar en un archivo o en la estructura LDAP del producto WebSphere Business Integrator. Los datos controlan los aspectos siguientes de la configuración del kernel:

- Clasificación y direccionamiento de mensajes
- Verificación de la instalación
- Modalidad de comunicación
- Rastreo

En el apartado “El archivo de configuración” en la página 65 se proporciona una descripción completa del archivo de configuración. Consulte la documentación de WebSphere Business Integrator para obtener más información sobre la configuración de dicho producto de modo que funcione con el kernel.

componente de rastreo

Software que escribe mensajes de rastreo. La mayor parte de los componentes del kernel utilizan el componente de rastreo. Consulte el apartado "Rastreo" en la página 29 para obtener una visión general del rastreo y la publicación *Problem Determination Guide* para obtener información detallada acerca del rastreo.

Mensaje y formato del mensaje

En MQSeries y la Oferta MQSeries Adapter, un *mensaje* es un colección de datos que envía un programa a otro programa. El formato del mensaje depende siempre de su ubicación en el flujo de mensajes en un momento determinado. MQSeries Adapter Kernel especifica tres tipos de mensajes, tal como se indica a continuación:

- *Mensaje de integración*—Es un mensaje que consta de datos de una aplicación de origen convertidos en otro formato como, por ejemplo, XML para enviar a una aplicación de destino. El mensaje de integración se inserta en el objeto de soporte de mensajes como datos del cuerpo del mensaje. XML es estándar para la representación de datos. Cuando el formato es XML, una *definición de tipo de documento* (DTD) define el formato. Una DTD consiste en uno o más archivos que contienen una definición formal de un documento, en este caso, del cuerpo del mensaje. Aunque es muy recomendable, no es necesario que el cuerpo del mensaje esté en un formato neutro de aplicación. El formato del cuerpo del mensaje puede estar patentado o especializado de otro modo; sin embargo, no se aconseja este tipo de formato.

  La Oferta MQSeries Adapter puede utilizar *Documentos de objetos de negocio* (BOD) para definir cuerpos de mensaje en los mensajes de integración. Una BOD es una representación de un proceso de negocio estándar que fluye en una organización o entre organizaciones. Algunos ejemplos incluyen: "añadir pedido de compra," "mostrar la disponibilidad del producto" y "añadir pedido de venta." OAG (Open Applications Group) define una BOD en XML. Se aconseja utilizar una BOD, pero no es obligatorio.
- *Objeto de soporte de mensajes*—Es un objeto que contiene el mensaje de integración y otros metadatos de cabecera que representan valores de control específicos de MQSeries Adapter Kernel. El adaptador origen crea el objeto de soporte de mensajes, establece la información de control adecuada y, si hay algún mensaje de integración que enviar, establece los datos del cuerpo. Los adaptadores destino reciben los objetos de soporte de mensajes, obtienen los datos del cuerpo y los convierten en datos específicos de la aplicación de destino. Los adaptadores origen y los adaptadores destino se crean utilizando MQSeries Adapter Builder.
- *Mensaje de comunicaciones*—Cualquier información específica del transporte de comunicaciones, además del objeto de soporte de mensajes, convertida en un formato de mensajería específico del transporte de comunicaciones

que se está utilizando. Algunos transportes de comunicaciones ofrecen soporte para más de un formato de mensajería. Generalmente, el transporte de comunicaciones considera que los valores de metadatos de cabecera del kernel combinados con el mensaje de comunicaciones son datos de la aplicación. Si desea ver más información, consulte el "Apéndice A. Modalidades de comunicación" en la página 103. Algunos ejemplos del transporte de MQSeries son la cabecera del mensaje específica de MQSeries y el objeto de soporte de mensajes clasificados. Los formatos específicos de MQSeries incluyen los siguientes:

– La cabecera del mensaje de MQSeries que añade MQSeries
– Si se utiliza MQSeries Integrator, la cabecera del mensaje específica de la versión:
  - La cabecera del mensaje de la versión 1 de MQSeries Integrator, si se utiliza la versión 1.1 de MQSeries Integrator
  - La cabecera del mensaje de la versión 2 de MQSeries Integrator, si se utiliza la versión 2 de MQSeries Integrator
– Los metadatos de la cabecera específica del kernel que representan valores de control
– El mensaje de integración (datos del cuerpo)

En el "Apéndice C. Cabeceras de mensaje" en la página 111 se incluye una lista de los campos más importantes que se utilizan en las cabeceras de mensaje de la Oferta MQSeries Adapter y sus descripciones.

Direccionamiento y entrega

El kernel direcciona cada mensaje desde el adaptador origen y lo entrega al adaptador destino adecuado. El direccionamiento se lleva a cabo en dos fases:

1. La parte origen del kernel coloca el mensaje en la cola de mensajes adecuada.
2. La parte destino del kernel obtiene el mensaje de la cola de mensajes e invoca al adaptador destino adecuado.

Numerosos factores determinan el direccionamiento:

• Colas de mensajes. En el nivel más básico, se deben configurar colas de mensajes para ofrecer soporte para el direccionamiento de la Oferta MQSeries Adapter.
• Valores de control de mensaje. Incluyen el identificador lógico de origen, el identificador lógico de destinación, el identificador lógico de respuestas, la categoría del cuerpo, el tipo de cuerpo, el identificador de la transacción, el identificador del mensaje, el acuse de recibo solicitado e indicaciones de la hora. Si desea obtener más información, consulte el apartado "Valores de control de mensaje" en la página 16. El identificador lógico de destinación del mensaje puede alterar temporalmente el archivo de configuración del

kernel. El direccionamiento puede cambiar de modo dinámico a medida que cambian los valores de control de mensaje en cada cabecera de mensaje. Sin embargo, el contenido de los datos del cuerpo del mensaje (mensaje de integración) no puede determinar el direccionamiento.

- Los valores de control de mensaje del archivo de configuración del kernel. El archivo puede especificar los identificadores lógicos de destino, nombres de cola y adaptadores destino asociados. Editando este archivo, puede determinar y modificar la configuración. Para obtener más información, consulte el apartado "El archivo de configuración" en la página 65.
- Además, se puede utilizar MQSeries Integrator para la intermediación de mensajes, incluido el direccionamiento complejo. El direccionamiento puede cambiar dinámicamente a medida que se modifica el contenido del cuerpo del mensaje. Consulte el apartado "Función de MQSeries Integrator" en la página 8. Por el contrario, la Oferta MQSeries Adapter sólo puede realizar el direccionamiento simple. El direccionamiento simple se basa en una combinación de valores de control de mensaje en el mensaje y valores de control de mensaje asociados en el archivo de configuración. No se basa en el contenido del cuerpo del mensaje.

Se puede solicitar que el kernel acuse recibo de la entrega del mensaje. Se trata de un acuse de recibo a nivel de la aplicación.

Flujo de ejecución

En este apartado se detalla el flujo de ejecución, es decir, cómo envía, direcciona, rastrea y entrega el kernel un mensaje en un entorno de trabajo real habitual. Consulte la Figura 2 en la página 15, en la que se incluye un diagrama del flujo de ejecución.

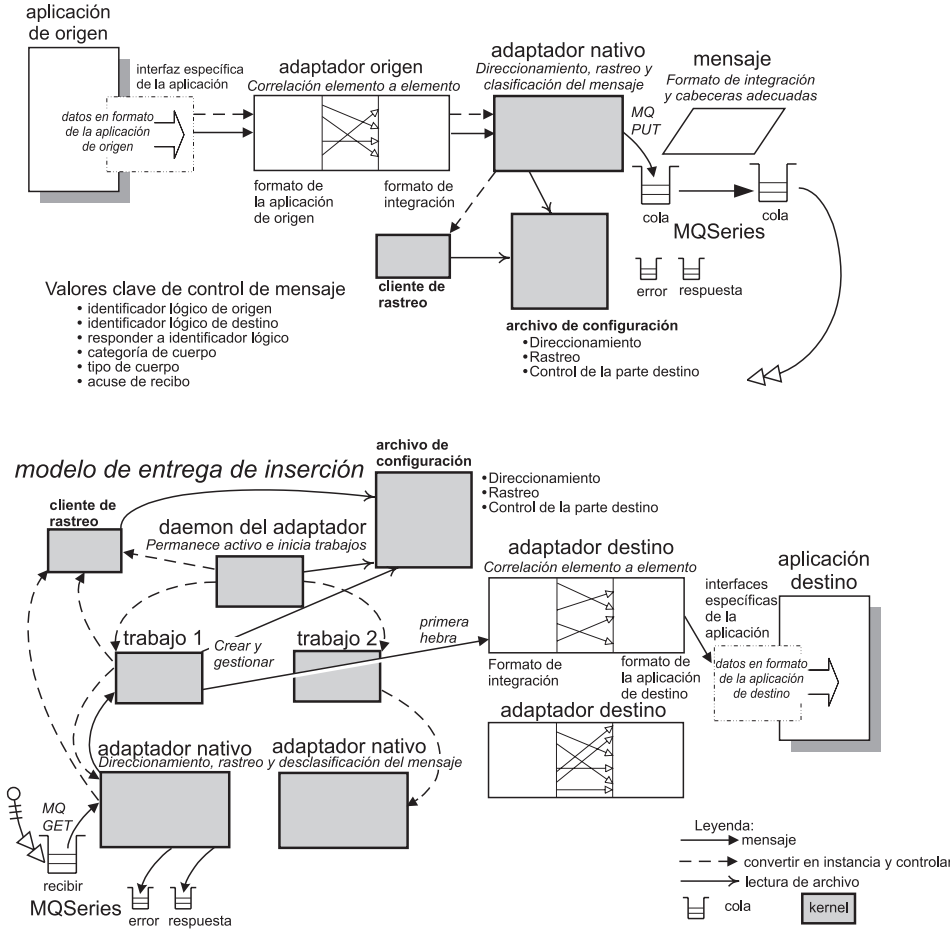


Figura 2. Visión general de la clasificación, el envío, el direccionamiento y el rastreo de un mensaje.

Parte origen del kernel

En este apartado se trata el flujo de ejecución de la parte origen del kernel, es decir, cómo se mueven los datos de la aplicación de origen a través de un adaptador origen y en un transporte de comunicaciones. El apartado "Parte destino del kernel" en la página 20 trata cómo los datos se mueven desde el transporte de comunicaciones al destino.

1. Utilizando una interfaz específica de la aplicación, el adaptador origen obtiene un mensaje de la aplicación de origen. Por lo general, la interfaz específica de la aplicación invoca al adaptador origen.
2. El adaptador origen lleva a cabo las funciones que se le han incorporado en MQSeries Adapter Builder. Generalmente, transforma los datos del

formato de la aplicación de origen a un formato de integración neutro de aplicación (para el cuerpo del mensaje).

Como parte de sus funciones, el adaptador origen pone varios valores de control de mensaje en la cabecera de MQSeries Adapter Kernel y los utiliza para envolver el mensaje. Los cinco primeros valores de control de mensaje determinan la clasificación y el direccionamiento, y el último valor determina el acuse de recibo.

valores de control de mensaje

identificador lógico de origen
Identificador lógico de la aplicación de origen. Siempre es obligatorio en el mensaje.

identificador lógico de destinación
Identificador lógico de la aplicación de destino. Si no se incluye en el mensaje, se utilizan los valores por omisión del archivo de configuración. En el archivo de configuración se pueden utilizar varios identificadores lógicos de destino, en lugar de valores que no están contenidos en el mensaje.

identificador lógico de respuestas
Identificador lógico de la aplicación al que se envían las respuestas en caso de que se solicite una respuesta. Toma el valor por omisión del identificador lógico de origen del mensaje.

categoría del cuerpo
Representa el tipo de aplicación del mensaje, por ejemplo, OAG o RosettaNet. Siempre es obligatorio en el mensaje.

tipo de cuerpo
Representa la finalidad específica del mensaje, por ejemplo "añadir pedido de venta" o "sincronizar inventario". Siempre es obligatorio en el mensaje.

acuse de recibo solicitado
Determina si la aplicación de origen solicita una respuesta. La respuesta puede tener los formatos siguientes:

- Datos de respuesta desde la aplicación de destino
- Un mensaje Confirmar BOD de OAG

Nota: OAG predefine el mensaje Confirmar BOD. Su categoría del cuerpo es `OAG` y su tipo de cuerpo es `CONFIRM_BOD_003`. También puede contener datos.

Esta respuesta es un acuse de recibo a nivel de la aplicación.

Cuando el kernel utiliza el método `sendRequestResponse` para enviar el mensaje, sólo se utiliza la primera respuesta que recibe

el método `sendRequestResponse`. Si se envía el mensaje original a varios destinos y se solicita una respuesta (lo que no se aconseja), sólo se devuelve la primera respuesta a la aplicación de origen.

El valor por omisión es sin acuse de recibo, por lo que no se solicita o envía ninguna respuesta.

3. El adaptador origen inicializa el adaptador nativo y sucede lo siguiente:
   - El identificador lógico de la aplicación en la que se ejecuta el adaptador origen.
   - El objeto de soporte de mensajes, que contiene los valores de control de mensaje y los datos del cuerpo del mensaje.
4. El adaptador nativo busca en el archivo de configuración si se ha habilitado el rastreo para este identificador lógico de origen. Si se ha habilitado el rastreo, el adaptador nativo convierte en instancia un cliente de rastreo.
5. El cliente de rastreo consulta el archivo de configuración para determinar el nivel de rastreo que debe utilizar, así como para obtener otros valores. El cliente de rastreo utiliza el nivel de rastreo para filtrar mensajes de rastreo. Consulte el apartado "Rastreo" en la página 29 para obtener una visión general del rastreo y la publicación *Problem Determination Guide* para obtener información detallada acerca del rastreo.
6. El adaptador nativo busca en el objeto de soporte de mensajes un identificador lógico de destinación y, si hay alguno, lo utiliza.
   - Si no hay ningún identificador lógico de destinación, el adaptador nativo busca el identificador lógico de destinación por omisión en el archivo de configuración, basado en el identificador lógico de origen, la categoría y el tipo de cuerpo.
   - Dependiendo del identificador lógico de origen, el adaptador nativo efectúa una búsqueda de múltiples fases de los valores de la categoría del cuerpo y el tipo de cuerpo en el archivo de configuración, siguiendo el orden que se indica a continuación:
     a. Valores de categoría y de tipo de cuerpo específicos.
     b. Valor de categoría de cuerpo específico y valor de tipo de cuerpo por omisión.
     c. Valor de categoría de cuerpo por omisión y valor de tipo de cuerpo específico.
     d. Valores de categoría y de tipo de cuerpo por omisión.

   Nota: el kernel utiliza esta búsqueda de múltiples fases cada vez que busca valores en el archivo de configuración.
7. El adaptador nativo busca la *modalidad de comunicación*, basada en el identificador lógico de destinación, la categoría y el tipo de cuerpo para

cada identificador lógico de destinación que se ha determinado en el paso anterior. Se ofrece soporte para las modalidades de comunicación siguientes:

MQPP El kernel transporta mensajes utilizando servicios base de MQSeries.

MQRFH1 El kernel transporta mensajes utilizando MQSeries y actúa como intermediario de mensajes utilizando MQSeries Integrator, versión 1.1.

MQRFH2 El kernel transporta mensajes utilizando MQSeries y actúa como intermediario de mensajes utilizando MQSeries Integrator, versión 2.

MQBD El kernel transporta mensajes utilizando servicios base de MQSeries, pero sólo envía y recibe datos del cuerpo.

MQ El kernel transporta mensajes utilizando MQSeries.

JMS El kernel transporta mensajes utilizando JMS (Java Message Service).

FILE El kernel coloca y obtiene los mensajes de un archivo. Esta modalidad sólo se proporciona para diagnósticos.

La estructura del mensaje es diferente en cada modalidad de comunicación. Consulte el apartado "Mensaje y formato del mensaje" en la página 12. Si desea ver más información acerca de las modalidades de comunicación, consulte el "Apéndice A. Modalidades de comunicación" en la página 103.

Nota: si se utiliza MQSeries Integrator, el destino final al que MQSeries Integrator envía el mensaje debe utilizar la misma modalidad de comunicación que MQSeries Integrator para recibir mensajes.

8. Dependiendo de la modalidad de comunicación, el adaptador nativo convierte en instancia una subclase en sí mismo para manejar el mensaje. La subclase se denomina *servicio lógico de mensajes*. Cada modalidad de comunicación tiene una subclase de servicio lógico de mensajes diferente.

   El adaptador nativo pasa los identificadores lógicos de destino y la categoría y el tipo de cuerpo al servicio lógico de mensajes.

9. La subclase del servicio lógico de mensajes busca los parámetros que necesita para enviar el mensaje. Por ejemplo, si la modalidad de comunicación es MQPP, los parámetros incluyen el formato y los nombres de las colas de error, respuesta y recepción. Dependiendo de los identificadores lógicos de destino y la categoría y el tipo de cuerpo que se le pasan, el servicio lógico de mensajes realiza una búsqueda de múltiples fases en el archivo de configuración:

a. Valores de categoría y de tipo de cuerpo específicos.
b. Valor de categoría de cuerpo específico y valor de tipo de cuerpo por omisión.
c. Valor de categoría de cuerpo por omisión y valor de tipo de cuerpo específico.
d. Valores de categoría y de tipo de cuerpo por omisión.

En este punto, el servicio lógico de mensajes tiene toda la información que necesita para direccionar y clasificar el mensaje.

10. El servicio lógico de mensajes lleva a cabo las tareas siguientes:
    - Clasifica el mensaje como adecuado para el formato y la modalidad de comunicación. En caso de que no se especifique de otro modo, cada modalidad de comunicación utiliza un formato por omisión. Por ejemplo, si la modalidad de comunicación es MQRFH2, el servicio lógico de mensajes crea cabeceras adecuadas y estructura el mensaje para el transporte utilizando MQSeries y para la intermediación utilizando MQSeries Integrator versión 2.
    - Envía el mensaje. Por ejemplo, si la modalidad de comunicación es MQRFH2, lo coloca en la cola de mensajes de MQSeries correspondiente.
11. Para enviar el mensaje se pueden utilizar dos métodos:
    - Si el adaptador nativo utiliza el método `sendMsg` para enviar el mensaje, no espera una respuesta.
    - Si el adaptador nativo utiliza el método `sendRequestResponse` para enviar el mensaje, el servicio lógico de mensajes espera la respuesta. Utilizando el servicio lógico de mensajes, el adaptador nativo supervisa el *período de tiempo de espera de recepción* que se ha establecido en el archivo de configuración para la cola de respuestas.

      El período de tiempo de espera de recepción se basa en el identificador de la aplicación de origen y en la categoría y el tipo de cuerpo.
      – Si se recibe un acuse de recibo, el adaptador nativo devuelve un mensaje.
      – Si no se recibe un acuse de recibo durante el período de tiempo de espera de recepción, el adaptador nativo no devuelve un mensaje.
12. MQSeries u otro software de mensajería transporta el mensaje según el modo en que se ha configurado. Si se desea, MQSeries Integrator puede realizar servicios de intermediación. Consulte el apartado "Función de MQSeries Integrator" en la página 8.
13. Cuando el adaptador origen ya no necesita el adaptador nativo, lo cierra para liberar recursos.

Parte destino del kernel

En este apartado se trata la utilización de MQSeries Adapter Kernel autónomo para recibir y procesar mensajes en la parte destino y se ofrece una descripción de alto nivel de la utilización del kernel con Websphere Application Server. Consulte el apartado "Utilización del MQSeries Adapter Kernel con WebSphere Business Integrator y WebSphere Application Server" en la página 29 para conocer la utilización del kernel con JMS, el componente JMS Listener de WebSphere Business Integrator y WebSphere Application Server en la parte destino del kernel. Este apartado describe el modelo de entrega de inserción, en el que el kernel se encarga de iniciar y gestionar la entrega del mensaje a la aplicación de destino. En el apartado "modelos de entrega" en la página 134 se proporciona una breve descripción de los modelos.

Visión general del trabajo adaptador

En este apartado se describen la estructura y comportamiento de los trabajos adaptadores de MQSeries Adapter Kernel. Una de las propiedades de la arquitectura de MQSeries Adapter Kernel es que las aplicaciones de destino no participan de manera activa en los flujos de datos de integración con otras aplicaciones; es decir, que normalmente las aplicaciones no sondean activamente los mensajes que se deben procesar. En este caso, los datos del mensaje se deben insertar activamente en la aplicación de destino. Los trabajos adaptadores insertan los datos del mensaje en una aplicación u otro servicio al seleccionar y llamar una gama de tipos de interfaz de servicio.

MQSeries Adapter Kernel puede alojar trabajos adaptadores que se ejecuten en un daemon autónomo (daemon del adaptador) o un servidor de aplicaciones Enterprise JavaBeans (actualmente, IBM WebSphere Application Server Advanced Edition). Los mensajes llegan al trabajo adaptador de diferentes maneras, según el tipo de entorno de destino que se utiliza. Si se utiliza un daemon del adaptador autónomo, éste aloja uno o más trabajos adaptadores autónomos que utilizan el adaptador nativo para recibir mensajes. Si se utiliza un servidor EJB, el componente JMS Listener recibe los mensajes y los pasa a un bean de mensajes de trabajo (a veces denominado trabajo adaptador de bean de mensaje).

Independientemente del entorno de destino que se utilice, una vez que el trabajo adaptador recibe el mensaje, lo envía al adaptador destino apropiado. A continuación, el adaptador destino realiza el trabajo necesario para entregar el mensaje a la aplicación de destino. Los adaptadores destino se han creado para que funcionen con aplicaciones de destino específicas. El daemon del adaptador, el servidor de aplicaciones, el trabajo adaptador autónomo y el bean de mensajes de trabajo no son específicos de ninguna aplicación de origen o destino.

El trabajo adaptador maneja dos tipos de interfaces de adaptador destino: adaptadores de mandato Enterprise Access Builder (EAB) y beans de sesión de servicio EJB. Cada tipo de adaptador incluye un manejador que establece el entorno adecuado, accede a cualquier tipo de información adicional sobre la configuración que necesita el adaptador y realiza otras tareas de bajo nivel necesarias para que el adaptador funcione. El manejador que se utiliza depende del tipo de adaptador que aparece en la lista del archivo de configuración. Los dos tipos de manejadores realizan las siguientes tareas adicionales:

- El manejador EAB obtiene una clase de conexión, que se utiliza para proporcionar información de conexión al adaptador destino e inicializa la ejecución de IBM Common Connector Framework (CCF). La clase de conexión pasa al identificador lógico de la aplicación de destino, que la utiliza para obtener información de conexión específica de la aplicación.
- El manejador EJB obtiene una conexión JNDI (Java Naming and Directory Interface™) y, a continuación, obtiene información sobre la interfaz remota del bean de sesión de servicio y cualquier otra información necesaria para acceder al bean de sesión de servicio.

El flujo del proceso básico de un trabajo adaptador dentro de un daemon del adaptador autónomo es como se indica a continuación:

1. Al arrancar, el daemon del adaptador convierte en instancia a uno o más trabajos adaptadores autónomos, según la información proporcionada en el archivo de configuración del kernel. Los valores del identificador lógico de la aplicación, la categoría de cuerpo opcional y el tipo de cuerpo pasan al daemon del adaptador. Los valores de la categoría y tipo de cuerpo se utilizan para obtener valores de configuración adicionales.
2. Todos los trabajos adaptadores autónomos realizan las tareas siguientes:
   a. El trabajo adaptador convierte en instancia a un adaptador nativo y empieza a recibir mensajes. Cada mensaje se recibe bajo el control transaccional y se devuelve al trabajo adaptador como objeto de soporte de mensajes.
   b. Por cada mensaje recibido, el trabajo adaptador recupera el tipo de mandato del adaptador destino para procesar el mensaje desde el archivo de configuración y obtiene el manejador adecuado para ese tipo de mandato.
   c. El manejador obtiene del archivo de configuración cualquier información adicional que necesite para replicar la instancia del adaptador destino. Convierte en instancia el adaptador destino y le pasa el mensaje.
   d. Si el mensaje se ha procesado correctamente (es decir, sin excepciones, errores o datos de retorno erróneos), la cola de mensajes entrantes lo confirma. Si el mensaje no se ha procesado correctamente, se coloca en

una cola de error. Si el mensaje no se ha procesado correctamente y no se puede colocar en una cola de error, entonces se restituye y todos los trabajos concluyen.

El flujo de proceso básico de un trabajo adaptador dentro de WebSphere Application Server es como se indica a continuación:

1. Un proceso JMS Listener que funcione con el servidor EJB de WebSphere Application Server Advanced Edition recibe un mensaje JMS. A continuación obtiene un bean de mensajes de trabajo para procesar el mensaje. Los valores del identificador lógico de la aplicación, la categoría de cuerpo opcional y el tipo de cuerpo son parte del entorno del bean de mensajes de trabajo. Los valores de la categoría y tipo de cuerpo se utilizan para obtener valores de configuración adicionales.
2. Todos los beans de mensajes de trabajo realizan las tareas siguientes:
   a. El bean de mensajes de trabajo convierte en instancia un adaptador nativo y utiliza el método `receiveMsg` en el adaptador nativo, pasándole el mensaje JMS. El adaptador nativo convierte el mensaje JMS en un objeto de mensaje y lo devuelve al objeto de soporte de mensajes.
   b. Por cada objeto de soporte de mensajes recibido, el trabajo adaptador recupera el tipo de mandato del adaptador destino para procesar el objeto de soporte de mensajes desde el archivo de configuración y obtiene el manejador adecuado para ese tipo de mandato.
   c. El manejador obtiene del archivo de configuración cualquier información adicional que necesite para replicar la instancia del adaptador destino. Convierte en instancia el adaptador destino y le pasa el mensaje.
   d. Si el objeto de soporte de mensajes se ha procesado correctamente (es decir, sin excepciones, errores o datos de retorno erróneos), la cola de mensajes entrantes lo confirma. Si el objeto de soporte de mensajes no se ha procesado correctamente, se coloca en una cola de error. Si el mensaje no se ha procesado correctamente y no se puede colocar en una cola de error, entonces se restituye y todos los trabajos concluyen.

El flujo de ejecución de la parte destino con un daemon del adaptador y un trabajo adaptador autónomo es como se indica a continuación:

1. Hay un daemon del adaptador para cada cola de recepción de la aplicación de destino. Se inicia el daemon del adaptador.

   Durante el arranque se le proporciona un nombre que sirve como identificador de la aplicación. Por lo general, cada uno de los nombres del daemon del adaptador se basa en el identificador lógico de destinación, es decir, el identificador lógico de la aplicación de destino. Por ejemplo, si el

daemon del adaptador presta servicio a una aplicación de destino cuyo identificador lógico de destinación es `ABC`, el nombre del daemon del adaptador es `ABCdaemon`.

Otros parámetros que se pueden pasar al daemon del adaptador durante el arranque incluyen la categoría y el tipo de cuerpo. Después, el adaptador nativo los utiliza para determinar la modalidad de comunicación y la cola de recepción para los mensajes de entrada.

Consulte el apartado "Inicio del kernel" en la página 92 para obtener instrucciones acerca del inicio del daemon del adaptador.

2. Cuando arranca, el daemon del adaptador busca en el archivo de configuración si se ha habilitado el rastreo para este nombre de daemon del adaptador. Si se ha habilitado, el daemon del adaptador convierte en instancia un cliente de rastreo.

   En la publicación *Problem Determination Guide* encontrará información detallada acerca del rastreo.

3. Cuando arranca, el daemon del adaptador convierte en instancia el primer trabajo y le pasa el nombre del daemon del adaptador, así como la categoría y el tipo de cuerpo del mensaje.
4. El primer trabajo busca en el archivo de configuración si se ha habilitado el rastreo para este nombre de daemon del adaptador. Si se ha habilitado el rastreo, el primer trabajo convierte en instancia un cliente de rastreo, que consulta el archivo de configuración para determinar el nivel de rastreo. Consulte en la publicación *Problem Determination Guide* la lista de los niveles de rastreo válidos.
5. Dependiendo del identificador de aplicación del daemon del adaptador, el primer trabajo busca en el archivo de configuración los valores que indican el número mínimo de trabajos que hay que convertir en instancia e iniciar.

   El primer trabajo también busca el *identificador de aplicación de dependencia*. El identificador de aplicación de dependencia es el nombre de la aplicación a la que atiende el trabajo. Posteriormente, se pasa al adaptador nativo.

6. El daemon del adaptador consulta el número mínimo de trabajos al primer trabajo.
7. El daemon del adaptador inicia el primer trabajo y, a continuación, convierte en instancia e inicia el número mínimo de trabajos.

   La finalidad de disponer de varios trabajos es permitir la entrega de mensajes de múltiples hebras a los adaptadores destino. Cada trabajo, junto con su adaptador destino, puede manejar una hebra. Si sólo existe un trabajo, la entrega de mensajes al adaptador destino y, por consiguiente, a la aplicación de destino, es de una sola hebra.

   En los sistemas AIX, hay dos políticas de planificación disponibles para hebras: planificación basada en el proceso y planificación basada en el sistema. En la planificación basada en el proceso (el valor por omisión),

todas las hebras de usuario se correlacionan con una agrupación de hebras de kernel del sistema operativo (OS) y se ejecutan en una agrupación de procesadores virtuales. En la planificación basada en el sistema, cada hebra de usuario se correlaciona con una única hebra de kernel de OS y se ejecuta en un único procesador virtual. Si utiliza adaptadores origen C que se invocan desde archivos ejecutables C en AIX, debe utilizar la planificación basada en el sistema. Para obtener más información acerca del establecimiento de la política de planificación de hebras en AIX, consulte el Paso 6 en la página 45.

Se debe tener en cuenta que sólo se ofrece soporte para la planificación basada en el proceso en los sistemas Windows, HP-UX, Solaris y OS/400.

Los demás trabajos también llevan a cabo los pasos siguientes que realiza el primer trabajo:

8. Cada trabajo convierte en instancia su adaptador nativo asociado, y cada trabajo tiene un adaptador nativo asociado. El identificador de aplicación de dependencia, así como la categoría y el tipo de cuerpo, se pasan al adaptador nativo, que utiliza estos tres valores para determinar la modalidad de comunicación y, utilizando el servicio lógico de mensajes, el formato y la cola de recepción para los mensajes de entrada. Este proceso es similar al que se utiliza para enviar mensajes.
9. El adaptador nativo obtiene el mensaje de comunicaciones de la cola de recepción bajo el control de confirmación y lo convierte en un objeto de soporte de mensajes. Elimina todas las cabeceras específicas del transporte de comunicaciones a excepción de la cabecera del kernel nativo.
10. El adaptador nativo pasa el objeto de soporte de mensajes al trabajo, que lee la categoría y el tipo de cuerpo, así como el valor de acuse de recibo solicitado de la cabecera de kernel nativo del mensaje.

    Según el identificador de aplicación de dependencia, la categoría y el tipo de cuerpo, el trabajo realiza una búsqueda de múltiples fases en el archivo de configuración del mandato de destino para invocar, en el orden siguiente:

    a. Valores de categoría y de tipo de cuerpo específicos.
    b. Valor de categoría de cuerpo específico y valor de tipo de cuerpo por omisión.
    c. Valor de categoría de cuerpo por omisión y valor de tipo de cuerpo específico.
    d. Valores de categoría y de tipo de cuerpo por omisión.

    Dependiendo del tipo de mandato de destino, el trabajo determina el manejador de tipo de adaptador destino adecuado, una clase Java que procesa el tipo de adaptador específico. Convierte en instancia el adaptador destino determinado.

11. Existen dos tipos de manejadores de tipo de adaptador: manejadores de adaptadores destino de mandato EAB y manejadores de adaptadores destino de bean de sesión de servicio EJB. Los diferentes tipos de manejadores de adaptador funcionan como se indica a continuación:

    Nota: El manejador de adaptadores destino de bean de sesión de servicio EJB sólo recibe soporte de WebSphere Business Integrator que se ejecuta con WebSphere Application Server en la plataforma Windows NT.

    - Si se invoca un manejador de adaptadores destino de mandato EAB, éste inicia el entorno CCF (Common Connector Framework), establece una clase de conexión con un nombre obtenido del archivo de configuración e invoca el adaptador destino EAB con el nombre obtenido del archivo de configuración.
    - Un adaptador destino de bean de sesión de servicio EJB debe interactuar con WebSphere Business Integrator y WebSphere Application Server para obtener la información de configuración adecuada e invocar el bean de sesión de servicio EJB. Consulte el apartado "Utilización del MQSeries Adapter Kernel con WebSphere Business Integrator y WebSphere Application Server" en la página 29 para conocer la utilización del kernel con JMS, el componente JMS Listener de WebSphere Business Integrator y WebSphere Application Server en la parte destino del kernel.

12. Cada tipo de adaptador tiene una interfaz diferente y clases de soporte necesarias, como se indica a continuación:
    - Un mandato de adaptador destino EAB tiene tres métodos para invocar, que se ejecutan en el siguiente orden:
        a. El método *set message input*, que establece el mensaje para procesarlo en el adaptador destino.
        b. El método *execute*, que procesa el mensaje que se ha colocado en el adaptador destino utilizando el método "set message input" y, a continuación, espera.
            1) El adaptador destino lleva a cabo las funciones que se le han incorporado utilizando MQSeries Adapter Builder. Por lo general, transforma los datos del mensaje de integración al formato de la aplicación de destino. Correlaciona elemento con elemento.
            2) Utilizando una interfaz específica de la aplicación, el adaptador destino envía el mensaje a la aplicación de destino.
            3) Dependiendo de la naturaleza de la aplicación de destino, la aplicación de destino envía o no envía una respuesta al adaptador destino.

c. El método *get message output*, que obtiene la respuesta del adaptador destino. La respuesta puede indicar, simplemente, que la aplicación de destino ha recibido el mensaje, pero también puede contener datos.

- Un bean de sesión de servicio EJB invoca un método, que requiere un objeto `TerminalDataContainer`. Los datos devueltos por el método se consideran los datos de respuesta y deben ser un objeto de tipo `TerminalDataContainer`.

13. Si el mandato del adaptador destino no emite una excepción o no tiene una respuesta Confirmar BOD (que puede indicar un error), el trabajo confirma que ha recibido el mensaje desde la cola de recepción utilizando el adaptador nativo.
14. Si se solicita un acuse de recibo, el trabajo llama al método `sendResponse` del adaptador nativo.
    - Si el adaptador destino crea una respuesta, coloca el identificador lógico de respuestas del mensaje original en el campo del identificador lógico de destinación del mensaje de respuesta.
    - Si el adaptador destino no crea una respuesta, el trabajo crea un mensaje de respuesta Confirmar BOD que contiene el estado de finalización.
      – Si no hay errores, el estado de finalización es satisfactorio.
      – Si hay errores, el estado de finalización se establece en una condición de error.
15. Se envía la respuesta.
    a. El trabajo envía el mensaje de respuesta, si se ha creado uno, al adaptador nativo.
    b. El adaptador nativo coloca el mensaje de respuesta en la cola de respuestas.
    c. El adaptador nativo envía el mensaje de respuesta, según el mensaje original que haya recibido:
       - Si se trata de un mensaje de solicitud de MQSeries, el adaptador nativo obtiene la información de la cola para la respuesta del mensaje de solicitud de MQSeries. La información de la cola prevalece sobre el identificador lógico de destinación del mensaje.
       - Si no se trata de un mensaje de solicitud de MQSeries, el adaptador nativo utiliza el método `sendMsg` para enviar la respuesta.
16. En caso de excepción o de un mensaje de respuesta Confirmar BOD con un estado de error, el trabajo anota un mensaje de excepción en el archivo de excepciones denominado `EpicSystemExceptionFile`*`nnnnnnnn`*`.log` que reside en el mismo directorio que el daemon del adaptador, donde *nnnnnnnn* es el número del archivo de anotaciones. Además, si se han instalado las clases WebSphere

Business Integrator, éstas envían una excepción al componente WebSphere Business Integrator Solution Management. Consulte el apartado "Mensajes de excepción" en la página 96.

17. En caso de excepción o de un mensaje de respuesta Confirmar BOD con un estado de error, el trabajo indica al adaptador nativo que coloque el mensaje original en la cola de errores. El nombre de la cola de errores se obtiene a partir del archivo de configuración según el identificador lógico de dependencia, y la categoría y el tipo de cuerpo del mensaje original.

    Según el identificador de aplicación de dependencia, la categoría y el tipo de cuerpo, el trabajo realiza una búsqueda de múltiples fases en el archivo de configuración en el orden siguiente:

    a. Valores de categoría y de tipo de cuerpo específicos.
    b. Valor de categoría de cuerpo específico y valor de tipo de cuerpo por omisión.
    c. Valor de categoría de cuerpo por omisión y valor de tipo de cuerpo específico.
    d. Valores de categoría y de tipo de cuerpo por omisión.

    - Si el adaptador nativo puede colocar el mensaje de error en la cola de errores, se indica al adaptador nativo que confirme el mensaje de la cola de recepción.
    - Si el adaptador nativo no puede colocar el mensaje de error en la cola de errores, se produce lo siguiente:
        a. El trabajo indica al adaptador nativo que restituya, es decir, que no confirme.
        b. El trabajo establece un indicador que indica a todos los trabajos del daemon del adaptador que concluyan, lo que significa que el mensaje tiene algún problema. La conclusión de todos los trabajos evita que otros trabajos vuelvan a procesar el mismo mensaje que presenta el problema con el mismo resultado.
        c. Si se produce un error de memoria insuficiente, la excepción se trata del mismo modo que el resto de excepciones, salvo que, en este caso, el trabajo establece un indicador para detenerse a sí mismo después de finalizar el proceso del mensaje actual, lo que permite que haya más memoria disponible para los demás trabajos.

18. Cuando el adaptador nativo notifica al trabajo que la tarea se ha llevado a cabo, el trabajo marca dos indicadores:
    - Si se debe detener el trabajo. Puede deberse a una condición de memoria insuficiente de Java.
    - Si se deben detener todos los trabajos, cuya causa puede ser la descrita en el paso anterior.

19. Si se establece uno de estos dos indicadores, el trabajo se detiene. Si no se establece ningún indicador, el trabajo procesa el mensaje siguiente. El trabajo solicita que el adaptador nativo reciba un mensaje.
20. Si se coloca un mensaje de respuesta en la cola de respuestas o un mensaje de error en la cola de errores, se produce lo siguiente:
    a. MQSeries u otro software de mensajería lo devuelve a la parte origen del kernel.
    b. Si el adaptador origen ha invocado al método `sendRequestResponse` del adaptador nativo, el kernel recupera el mensaje de la cola de respuestas y lo devuelve al adaptador origen. Si el adaptador origen ha invocado al método `sendMsg`, el kernel coloca el mensaje en la cola de recepción de la aplicación de origen.

Posibilidades transaccionales

Una *transacción* es un conjunto de operaciones que se deben ejecutar como una unidad de trabajo indivisible. Si todas las operaciones que constituyen la transacción obtienen un resultado satisfactorio, la transacción se *confirma*, lo que significa que se han llevado a cabo todas las operaciones. Si una o más de las operaciones que constituyen la transacción no se ejecutan correctamente, la transacción se *restituye*, lo que significa que no se ha realizado ninguna operación. Utilizando las posibilidades transaccionales de MQSeries Adapter Kernel, un adaptador origen puede llevar a cabo una serie de operaciones como una única unidad, con la seguridad de que todas las operaciones han sido satisfactorias si se confirma la transacción o que no se ha realizado ninguna operación si se restituye la transacción.

Las posibilidades transaccionales se pueden incorporar en adaptadores utilizando MQSeries Adapter Builder o los métodos `begin`, `rollback` y `commit` en la clase `EpicNativeAdapter` de la API Java del kernel. Si se invoca un método transaccional en un contexto no permitido (por ejemplo, se invoca al método `commit` sin haber invocado antes al método `begin` o si se invoca al método `begin` dentro del ámbito de otra transacción), el kernel pasa por alto la invocación y emite un aviso para el rastreo. Para obtener más información acerca de la utilización de API, consulte el "Capítulo 5. Utilización de API de MQSeries Adapter Kernel" en la página 99.

Limitaciones

Las posibilidades transaccionales del kernel presentan las limitaciones siguientes:

- No se ofrece soporte para transacciones con el método `sendRequestResponse`.
- No se ofrece soporte para transacciones anidadas (es decir, transacciones que se invocan desde otras transacciones).

- No todas las modalidades de comunicación ofrecen soporte para las transacciones. Para obtener información más detallada, consulte el "Apéndice A. Modalidades de comunicación" en la página 103.

Rastreo

Un mensaje de rastreo contiene el estado del proceso de un mensaje en un punto determinado del kernel. Puede utilizar mensajes de rastreo como ayuda para el diagnóstico de problemas con el kernel o con los adaptadores. En la publicación *Problem Determination Guide* de MQSeries Adapter Kernel se explica cómo utilizar el rastreo con el kernel.

Utilización del MQSeries Adapter Kernel con WebSphere Business Integrator y WebSphere Application Server

En este apartado se trata la utilización del MQSeries Adapter Kernel con los productos WebSphere Business Integrator y WebSphere Application Server. Si desea información más detallada, consulte la documentación de WebSphere Business Integrator.

JMS Listener

WebSphere Business Integrator ofrece un componente denominado JMS Listener que funciona con MQSeries Adapter Kernel y WebSphere Application Server Advanced Edition para proporcionar un método alternativo de entrega de mensajes a las aplicaciones de destino. El JMS Listener se ejecuta dentro del servidor EJB (Enterprise Javabeans) de WebSphere Application Server. En este apartado se proporciona una visión general de la funcionalidad del JMS Listener. Para obtener información adicional, incluidos los detalles de la configuración de WebSphere Business Integrator y el JMS Listener, consulte la documentación de WebSphere Business Integrator. Consulte el apartado "Configuración del kernel" en la página 58 para obtener información sobre la configuración de MQSeries Adapter Kernel de modo que reconozca el JMS Listener como destino. La utilización del JMS Listener como destino es equivalente a enviar un mensaje a un daemon del adaptador.

Antes de poder utilizar el JMS Listener, debe difundir un trabajo adaptador de bean de mensaje MQSeries Adapter Kernel y los adaptadores bean de sesión de servicio Java o los adaptadores EAB para la parte destino del kernel. Realice estas tareas utilizando MQSeries Adapter Builder. En un entorno WebSphere Business Integrator, el funcionamiento del kernel dentro de WebSphere Application Server es similar a su funcionamiento con un daemon del adaptador autónomo, con la excepción que el JMS Listener recibe el mensaje en nombre del trabajo adaptador e invoca al trabajo adaptador adecuado. En un entorno MQSeries Adapter Kernel autónomo, el daemon del adaptador arranca los trabajos adaptadores, que a su vez reciben los mensajes directamente.

La secuencia de eventos cuando MQSeries Adapter Kernel funciona con el JMS Listener es como se indica a continuación:

1. El JMS Listener, supervisando una cola JMS, recibe un objeto de mensaje JMS, bien de un cliente EJB o de una aplicación distinta a EJB.
2. El JMS Listener convierte en instancia un *bean de mensajes de trabajo* y le pasa el objeto de mensaje. El bean de mensajes de trabajo es una instancia de un *bean de sesión*, un tipo de Enterprise Bean que encapsula los datos temporales asociados a un cliente específico.
3. El bean de mensajes de trabajo convierte el objeto de mensaje JMS en un objeto de soporte de mensajes MQSeries Adapter Kernel.
4. Según los valores de cabecera del mensaje, el kernel invoca a un adaptador EAB o a un adaptador EJB. Si el tipo de adaptador que se debe invocar es un adaptador EAB, el flujo de datos es el mismo que en el caso de un adaptador autónomo. Si el tipo de adaptador que se debe invocar es un adaptador EJB, se invoca a un manejador EJB que realiza las siguientes tareas:
   - Determina el bean de sesión de servicio correcto (interfaz inicial) que se debe invocar, el método adecuado que se debe invocar y el tipo de parámetro de entrada del método para el objeto `TerminalDataContainer`.
   - Convierte los datos de aplicación que contiene el objeto de soporte de mensajes en la estructura de datos apropiada `TerminalDataContainer` para el bean de sesión de servicio utilizando una clase `Mapper`. El objeto `TerminalDataContainer` contiene los metadatos del objeto de soporte de mensajes además de los objetos de aplicación. En muchos casos, el objeto de aplicación es la serie de caracteres del documento XML de datos de cuerpo del objeto de soporte de mensajes.
   - Invoca al bean de sesión de servicio, pasando el objeto `TerminalDataContainer` al método adecuado del bean de sesión de servicio. El bean de sesión de servicio, que forma parte del adaptador de servicio Java, es el destino del mensaje.
5. Si se solicita una respuesta, el bean de mensajes de trabajo convierte el objeto de respuesta `TerminalDataContainer` en un objeto de soporte de mensajes y envía la respuesta utilizando el adaptador nativo.
6. Si se produce un error, el bean de mensajes de trabajo coloca el objeto de soporte de mensajes en una cola de error utilizando el adaptador nativo.

Soporte del idioma nacional

MQSeries Adapter Kernel ofrece soporte del idioma nacional cuando se utilizan los adaptadores Java. No se ofrece soporte del idioma nacional para los adaptadores C.

Capítulo 2. Planificación de la instalación del kernel

En este capítulo se enumeran los requisitos previos y los componentes de MQSeries Adapter Kernel.

Para obtener la información detallada más actualizada, consulte el sitio web de la familia de productos MQSeries en la dirección siguiente:

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

IBM se reserva el derecho de actualizar la información que se muestra aquí. Para obtener la información más actualizada en relación con los niveles de software para el que se ofrece soporte, consulte la dirección siguiente:

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

Hardware

MQSeries Adapter Kernel se ejecuta en el hardware siguiente:

- Una máquina IBM PC (o compatible) que ejecute Windows NT 4.0, Service Pack 5 o posterior, o Windows 2000, Service Pack 1.
- Una máquina IBM RS/6000 que ejecute AIX, versión 4.3.2 ó 4.3.3.
- Una máquina HP Series 9000 que ejecute HP-UX, versión 11.0.
- Una máquina Sun SPARC o UltraSPARC que ejecute Solaris versión 8.
- Una máquina IBM AS/400 o iSeries que ejecute OS/400, versión 4.4 ó 4.5.

Nota: la instalación de MQSeries Adapter Kernel en OS/400 requiere un sistema Windows para la interfaz con la máquina AS/400. Si desea obtener más información, consulte el apartado "Requisitos previos para la instalación en OS/400" en la página 34.

MQSeries Adapter Kernel requiere, aproximadamente, 25 Mb de espacio de disco, como mínimo, para los datos y el código del producto.

Asegúrese de que hay suficiente espacio de disco para contener los adaptadores. Su tamaño depende de las estructuras de datos, la complejidad de las correlaciones y el código personalizado que se utilice. A continuación, se incluyen algunos ejemplos de diferentes tamaños de adaptador en sistemas Windows. Los adaptadores de su sitio pueden requerir más o menos espacio de disco. Cada ejemplo representa el origen del adaptador, el código de adaptador compilado, el origen de la API y el código compilado de la API en Mb o Kb.

- Adaptador origen para añadir un pedido de venta: 1,89 Mb

© Copyright IBM Corp. 2000, 2001

- Adaptador destino para sincronizar un registro de cliente: 389 Kb
- Adaptador destino para sincronizar un registro de inventario: 161 Kb
- Adaptador destino para sincronizar un elemento: 249 Kb
- Adaptador destino para sincronizar un pedido de venta: 579 Kb

Además, se deben destinar 20 Mb, como mínimo, para el espacio de trabajo del kernel y los adaptadores. Los requisitos de espacio de trabajo pueden variar según unos factores determinados como, por ejemplo, el número y el tamaño de las colas y el tamaño de los archivos de rastreo.

Software

En este apartado se enumera el software que se puede utilizar con MQSeries Adapter Kernel. Se indican los niveles para los que se ofrece soporte. Consulte el "Apéndice B. Configuraciones validadas" en la página 109. En los sistemas de desarrollo se necesitan compiladores C, pero no en los sistemas de trabajo real. Los compiladores C que se enumeran aquí se han probado con resultado satisfactorio con MQSeries Adapter Kernel. Es posible que otros compiladores C funcionen correctamente con el kernel, pero no se ofrece soporte para éstos oficialmente.

Para sistemas Windows:

- Microsoft Windows NT, versión 4.0, Service Pack 5 o posterior; o Microsoft Windows 2000, Service Pack 1. Para determinar la versión y el paquete de servicio de Microsoft Windows, abra Windows Explorer y, a continuación, pulse en **Ayuda > Acerca de Windows**.
- Compilador Microsoft Visual C++ 6.0.
- MQSeries versión 5.2 con SupportPac MA88.
- IBM Java Development Kit (JDK) versión 1.2.2 o 1.3.

  Nota: Windows NT y Windows 2000 son actualmente las únicas plataformas en las que MQSeries Adapter Kernel da soporte a JDK versión 1.3.

Para AIX:

- Sistema operativo AIX, versión 4.3.2 ó 4.3.3.
- IBM C Set++ para AIX, versión 3.1.3.
- MQSeries versión 5.2 con SupportPac MA88.
- Java Development Kit versión 1.2.2. No se da soporte a JDK 1.3.
- X Window System (X11R5 o posterior). Se necesita para la instalación, pero no para la ejecución.

Para HP-UX:

- Sistema operativo HP-UX, versión 11.0.
- Compilador C C/ANSI HP-UX. En el archivo `readme.txt` encontrará información detallada.
- MQSeries versión 5.2 con SupportPac MA88.
- Java Development Kit versión 1.2.2. No se da soporte a JDK 1.3.
- X Window System (X11R5 o posterior). Se necesita para la instalación, pero no para la ejecución.

Para Solaris:

- Entorno operativo Solaris, versión 8.
- Compiladores C/C++ Sun Workshop. En el archivo `readme.txt` encontrará información detallada.
- MQSeries versión 5.2 con SupportPac MA88.
- Java Development Kit versión 1.2.2. No se da soporte a JDK 1.3.
- X Window System (X11R5 o posterior). Se necesita para la instalación, pero no para la ejecución.

Para OS/400:

- Sistema operativo OS/400, versión 4.4 ó 4.5, con los programas siguientes:
  - Kit de herramientas de Java y Java Developer Kit, versión 1.2.2. No se da soporte a JDK 1.3. El Kit de herramientas de Java y Java Developer Kit se entregan con el número de programa bajo licencia 5769–JV1. Para obtener más información acerca de las versiones de Java Developer Kit que se necesitan para instalar MQSeries Adapter Kernel en un sistema AS/400, consulte el apartado “Requisitos previos para la instalación en OS/400” en la página 34.
  - La opción de servidores de sistema principal, que se entrega con el número de programa bajo licencia 5769–SS1, opción 12.
  - Qshell Interpreter, que se entrega con el número de programa bajo licencia 5769–SS1, opción 30.
  - TCP/IP, que se entrega con el número de programa bajo licencia 5769–TC1.
  - Integrated Language Environment C para AS/400, que se entrega con el número de programa bajo licencia 5769–CX2.
- MQSeries versión 5.2 con SupportPac MA88.

Para informarse acerca de los requisitos adicionales para la instalación de MQSeries Adapter Kernel en OS/400, consulte el apartado “Requisitos previos para la instalación en OS/400” en la página 34.

Con MQSeries Adapter Kernel se ofrece soporte para los productos siguientes:

- MQSeries versión 5.2 con SupportPac MA88

Nota: si no se utiliza MQSeries, se debe utilizar otro producto de software de envío de mensajes como, por ejemplo, una implementación de JMS (Java Message Service).

- MQSeries Integrator, versión 1.1
- MQSeries Integrator, versión 2

Consulte el "Apéndice B. Configuraciones validadas" en la página 109, en el que se ha incluido una lista de las configuraciones validadas de MQSeries Adapter Kernel, MQSeries y MQSeries Integrator.

Requisitos previos para la instalación en OS/400

En este apartado se describen los requisitos previos para la instalación de MQSeries Adapter Kernel en un sistema AS/400 o iSeries. En el Paso 3 en la página 42 encontrará instrucciones detalladas para la instalación de MQSeries Adapter Kernel en un sistema AS/400. Puesto que el terminal AS/400 no ofrece soporte nativo para gráficos Java, se necesita una estación de trabajo habilitada para gráficos como, por ejemplo, un sistema Windows, para ejecutar el programa de instalación de la GUI basada en Java del kernel. La estación de trabajo puede tener interfaz con el sistema AS/400 de los modos siguientes:

- A través de Remote AWT, en el que todos los gráficos se procesan en el sistema AS/400 y se muestran en la estación de trabajo. En el apartado "Utilización de Remote AWT" se proporciona información detallada.
- Como un cliente conectado, en el que la estación de trabajo procesa y muestra los gráficos. En el apartado "Utilización de un cliente conectado" en la página 35 se proporciona información detallada.

En este apartado se da por supuesto que se está utilizando un sistema Windows como estación de trabajo habilitada para gráficos.

Utilización de Remote AWT

Cuando se utiliza Remote AWT, el proceso de gráficos de Java se realiza en el sistema AS/400 y los gráficos se muestran en una estación de trabajo cliente conectada al sistema AS/400. En este apartado se describen los requisitos que se deben cumplir para instalar MQSeries Adapter Kernel en un sistema AS/400 utilizando Remote AWT.

Se deben instalar los programas siguientes con OS/400:

- Kit de herramientas de Java y Java Developer Kit, versión 1.2.2. El Kit de herramientas de Java y Java Developer Kit se entregan con el número de programa bajo licencia 5769–JV1. Java Developer Kit proporciona las posibilidades de Remote AWT en OS/400.
- TCP/IP, que se entrega con el número de programa bajo licencia 5769–TC1. Si desea obtener más información acerca de TCP/IP, consulte las publicaciones *AS/400 TCP/IP Fastpath Setup Information* y *AS/400 TCP/IP*

Configuration, que están disponibles en la biblioteca de AS/400 en la dirección siguiente: www.ibm.com/servers/eserver/iseries/library/.

Los requisitos para la estación de trabajo son los siguientes:

- Una máquina IBM PC (o compatible) que ejecute Windows 95, Windows 98, Windows NT o Windows 2000.
- Una conexión TCP/IP al sistema AS/400.
- JDK 1.2.2 o posterior.

Para configurar e iniciar Remote AWT, lleve a cabo los pasos que se indican a continuación:

1. Asegúrese de que ha instalado JDK 1.2.2, o posterior, en la estación de trabajo.
2. Asegúrese de que hay una conexión TCP/IP entre el sistema AS/400 y la estación de trabajo.
3. Copie el archivo `RAWTGui.jar` del directorio `/QIBM/ProdData/Java400/jdk12` del sistema AS/400 en un directorio de la estación de trabajo.
4. En la estación de trabajo, cambie al directorio en el que ha copiado el archivo `RAWTGui.jar` e inicie Remote AWT entrando el mandato siguiente:

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

Nota: Debido a que el proceso de gráficos Java en un sistema AS/400 necesita muchos recursos, es posible que si se utiliza Remote AWT se tarde más que si se utiliza un cliente conectado para instalar MQSeries Adapter Kernel.

Si desea ver más información acerca de Remote AWT, consulte la biblioteca de AS/400 en la dirección siguiente: www.ibm.com/servers/eserver/iseries/library/.

Utilización de un cliente conectado

Cuando se utiliza un cliente conectado para instalar MQSeries Adapter Kernel en un sistema AS/400, el proceso de gráficos de Java se realiza en la estación de trabajo cliente, no en el sistema AS/400. En este apartado se describen los requisitos que se deben cumplir para instalar MQSeries Adapter Kernel en un sistema AS/400 utilizando un cliente conectado.

Se deben instalar los programas siguientes con OS/400:

- Kit de herramientas de Java y Java Developer Kit, versión 1.2.2. El Kit de herramientas de Java y Java Developer Kit se entregan con el número de programa bajo licencia 5769–JV1.
- La opción de servidores de sistema principal, que se entrega con el número de programa bajo licencia 5769–SS1, opción 12.
- TCP/IP, que se entrega con el número de programa bajo licencia 5769–TC1.

Los requisitos para la estación de trabajo son los siguientes:

- Una máquina IBM PC (o compatible) que ejecute Windows NT 4.0, Service Pack 5, o Windows 2000, Service Pack 1.
- Una conexión TCP/IP al sistema AS/400.
- JDK 1.2.2 o posterior.

Componentes del kernel

Después de la instalación, MQSeries Adapter Kernel reside en el directorio raíz. Contiene subdirectorios que, a su vez, contienen otros directorios. Se enumeran la raíz y sus subdirectorios, junto con un resumen de los archivos más importantes para la instalación y la configuración.

root El nombre por omisión es `C:\Archivos de programa\MQAK` en los sistemas Windows, `/usr/lpp/mqak` en AIX, `/MQAK` en HP-UX, `/opt/MQAK` en Solaris y `/QIBM/ProdData/mqak` en OS/400. Contiene lo siguiente:

- Todos los demás directorios de MQSeries Adapter Kernel.
- El archivo `aqmsetenv.bat` (sistemas Windows) o `aqmsetenv.sh` (UNIX), que cambia las variables de entorno del sistema después de la instalación, si se desea.
- El archivo `readme.txt`.
- El archivo `aqmuninstall.bat` (sistemas Windows) o `aqmuninstall.sh` (UNIX).

bin Contiene lo siguiente:

- Bibliotecas de clase y bibliotecas compartidas.
- Adaptadores que se proporcionan como parte del kernel, para utilizarlos únicamente para verificación.
- El archivo `aqmversion.bat` (sistemas Windows) o `aqmversion.sh` (UNIX y OS/400), un script que se ejecuta para visualizar el número de versión del kernel.
- El archivo `aqmcrtmsg.bat` (sistemas Windows) o `aqmcrtmsg.sh` (UNIX y OS/400), un script que se ejecuta para crear un archivo XML para validar el archivo de configuración antes de utilizarlo para el trabajo real.
- El archivo `aqmsndmsg.bat` (sistemas Windows) o `aqmsndmsg.sh` (UNIX y OS/400), un script que se ejecuta para validar el archivo de configuración antes de utilizarlo para el trabajo real.
- El archivo `aqmstrad.bat` (sistemas Windows) o `aqmstrad.sh` (UNIX y OS/400), un script que se ejecuta para iniciar el daemon del adaptador.

- El archivo `aqmstrtd.bat` (sistemas Windows) o `aqmstrtd.sh` (UNIX y OS/400), un script que se ejecuta para iniciar el servidor de rastreo.

documentation
Contiene la documentación del producto, incluido el Centro de información.

runtimefiles
Contiene archivos de ejecución del kernel.

samples

Contiene ejemplos de adaptadores y archivos de programas de utilidad y configuración asociados que le permiten realizar pruebas y aprender.

Nota: La finalidad del kernel es utilizarlo con adaptadores que se generan usando MQSeries Adapter Builder. El kernel no tiene como finalidad que lo utilicen sólo las invocaciones a las API del kernel desde el código personalizado. Los ejemplos del adaptador se proporcionan únicamente como ayuda para comprender el funcionamiento del kernel y para los diagnósticos.

- Ejemplos de adaptador.
- El archivo de instalación del kernel, `aqmsetup`, con valores que ofrecen soporte para los ejemplos de adaptadores. En el apartado "El archivo de instalación" en la página 64 se trata este archivo con más detalle.
- El archivo de configuración del kernel, `aqmconfig.xml`, con valores que ofrecen soporte para los ejemplos de adaptadores, además de incluir valores de rastreo de ejemplo. En el apartado "El archivo de configuración" en la página 65 se trata este archivo con más detalle.

toolkit
Contiene SDK (Kit de herramientas de desarrollo de software), que consta de lo siguiente:

- Archivos de cabecera.
- Archivos de biblioteca, que se utilizan durante la compilación en los sistemas Windows.

uninstall
Contiene los archivos que se utilizan para desinstalar el kernel.

verification

Contiene los archivos siguientes, que ofrecen soporte para verificar la instalación del kernel:

- El archivo `aqmverifyinstall.bat` (sistemas Windows) o `aqmverifyinstall.sh` (UNIX y OS/400), un script que se ejecuta para verificar la instalación del kernel en un sistema.
- El archivo `aqmcreateq.bat` (sistemas Windows) o `aqmcreateq.sh` (UNIX y OS/400), un script que crea colas de MQSeries para la verificación. Consulte el apartado "Creación de colas de MQSeries" en la página 97.
- El archivo `aqmconfig.xml`. En el apartado "El archivo de configuración" en la página 65 se trata este archivo con más detalle.
- El archivo `aqmsetup`. En el apartado "El archivo de instalación" en la página 64 se trata este archivo con más detalle.
- El archivo `aqminstalltest.xml`.

Capítulo 3. Instalación del kernel

En este capítulo se explican los pasos necesarios para instalar y verificar MQSeries Adapter Kernel. La instalación consta de los siguientes pasos generales:

Paso 1. Lleve a cabo la preparación para la instalación. Si desea obtener más información, consulte el apartado "Preparación para la instalación" en la página 40.

Paso 2. Instale el kernel. Si desea obtener más información, consulte el apartado "Instalación del kernel" en la página 41.

Paso 3. Complete los diferentes pasos de la instalación posterior. Si desea obtener más información, consulte el apartado "Conclusión de la instalación posterior" en la página 44.

Paso 4. Verifique la instalación. Si desea obtener más información, consulte el apartado "Verificación de la instalación" en la página 47.

En este capítulo también se tratan los siguientes temas:

- Utilización de la instalación silenciosa para instalar MQSeries Adapter Kernel. Si desea obtener más información, consulte el apartado "Utilización de la instalación silenciosa" en la página 52.
- Actualización de MQSeries Adapter Kernel desde una versión anterior. Si desea obtener más información, consulte el apartado "Actualización del kernel" en la página 53.
- Eliminación de una instalación de MQSeries Adapter Kernel. Si desea obtener más información, consulte el apartado "Eliminación del kernel" en la página 55.

Después de instalar el kernel, realice las siguientes tareas adicionales para prepararlo para su uso:

1. Configure el kernel. Si desea obtener más información, consulte el apartado "Configuración del kernel" en la página 58.
2. Configure el software de envío de mensajes y el software opcional. Si desea obtener más información, consulte el apartado "Configuración de MQSeries y MQSeries Integrator" en la página 92.
3. Cree los adaptadores utilizando MQSeries Adapter Builder y, a continuación, pruébelos y difúndalos.
4. Inicie el kernel. Si desea obtener más información, consulte el apartado "Inicio del kernel" en la página 92.

© Copyright IBM Corp. 2000, 2001

Preparación para la instalación

Para instalar MQSeries Adapter Kernel debe tener autorización de administrador o root. Debe disponer del permiso necesario para crear y acceder a los archivos en la ubicación en la que va a instalar MQSeries Adapter Kernel y la ubicación en la que va a colocar los dos archivos de configuración del kernel, y debe tener el directorio actual en la vía de acceso ejecutable. Asegúrese de que todos los ID de usuario que ejecuten el kernel dispongan de los permisos de lectura, escritura y ejecución.

Debe tener autorización para realizar operaciones de MQSeries como, por ejemplo, crear gestores de colas y crear y acceder a las colas. Estas operaciones se llevan a cabo de modos diferentes en las distintas plataformas. Para obtener más información, consulte en la publicación *MQSeries Guía de administración* el apartado dedicado a su plataforma.

El identificador de usuario que inicie los procesos del kernel debe estar en el grupo mqm. Hay dos tipos de procesos de kernel:

- Daemon del adaptador, uno para cada aplicación de destino a la que atiende el sistema
- Servidor de rastreo (opcional)

Se debe tener en cuenta que el adaptador origen se ejecuta en el proceso de la aplicación de origen, por lo que se deben iniciar todos los daemons o servidores que contengan el adaptador origen para que éste se ejecute.

Debe instalar y configurar el kernel para ejecutar los adaptadores que ha generado. Sin embargo, no es necesario instalar el kernel para instalar MQSeries Adapter Builder o utilizarlo para generar los adaptadores.

Realice los siguientes pasos antes de empezar la instalación:

- Lea el archivo `readme.txt` del CD-ROM o la red de área local. Es posible que contenga información importante, disponible después de finalizar esta publicación. Está situado en el directorio de instalación raíz.
- Visite el sitio web de MQSeries, que se encuentra en la dirección siguiente: www.ibm.com/software/ts/mqseries/. Es posible que contenga información importante, disponible después de publicar este documento, y también podría incluir una nueva edición de esta publicación.
- Si está actualizando una versión anterior de MQSeries Adapter Kernel, en el apartado “Actualización del kernel” en la página 53 puede obtener las instrucciones necesarias.
- Asegúrese de que se cumplen los requisitos previos de software y de hardware. En los apartados “Hardware” en la página 31 y “Software” en la página 32 encontrará información detallada. MQSeries debe estar instalado

y en ejecución para poder verificar la instalación de MQSeries Adapter Kernel. Asegúrese de que ha instalado y configurado el soporte Java de MQSeries.

Instalación del kernel

Para instalar MQSeries Adapter Kernel en un sistema Windows (Windows NT o Windows 2000), en una plataforma UNIX (AIX, HP-UX o Solaris) o en OS/400, lleve a cabo los pasos siguientes específicos del sistema operativo:

En sistemas Windows:

Paso 1. Inicie el programa de instalación tal como se indica a continuación:

- Si va a realizar la instalación desde una red de área local, cambie al directorio que contiene los archivos de instalación de MQSeries Adapter Kernel y ejecute el archivo `install.bat`.
- Si va a realizar la instalación desde el CD-ROM, inserte el CD-ROM de MQSeries Adapter Kernel en la unidad de CD-ROM. Si ha habilitado la ejecución automática, el programa de instalación se inicia automáticamente; en caso contrario, ejecute el archivo `install.bat` del directorio raíz del CD-ROM para iniciar el programa de instalación.

Nota: En los sistemas Windows, no es necesario copiar el archivo `install.bat` a otra ubicación antes de ejecutarlo. Durante el proceso de instalación, se le solicita que elija el lugar en que desea instalar MQSeries Adapter Kernel.

Paso 2. Siga las indicaciones del programa de instalación. Tenga en cuenta que si elige instalar MQSeries Adapter Kernel en una ubicación distinta de la que se indica por omisión (que, en los sistemas Windows es `C:\Archivos de programa\MQAK`), debe especificar el directorio de instalación como un nombre de vía de acceso completamente calificado, no como un nombre de vía de acceso relativo.

En UNIX:

Paso 1. Inicie el programa de instalación tal como se indica a continuación:

- Si va a realizar la instalación desde una red de área local, cambie al directorio que contiene los archivos de instalación de MQSeries Adapter Kernel y ejecute el script `install.sh`.
- Si va a realizar la instalación desde el CD-ROM, inserte el CD-ROM de MQSeries Adapter Kernel en la unidad de CD-ROM y, si fuera necesario, monte la unidad de CD-ROM siguiendo las

instrucciones que se proporcionan en la documentación del sistema operativo. Ejecute el script `install.sh` en el directorio raíz del CD-ROM.

Paso 2. Siga las indicaciones del programa de instalación. Tenga en cuenta que si elige instalar MQSeries Adapter Kernel en una ubicación distinta de la que se indica por omisión, debe especificar el directorio de instalación como un nombre de vía de acceso completamente calificado, no como un nombre de vía de acceso relativo. Los directorios de instalación por omisión en UNIX son los que se indican a continuación:

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

En OS/400:

Paso 1. Asegúrese de que se cumplen todos los requisitos previos que se indican en la lista de los apartados "Hardware" en la página 31, "Requisitos previos de software para OS/400" en la página 33 y "Requisitos previos para la instalación en OS/400" en la página 34. Tenga en cuenta que para instalar MQSeries Adapter Kernel en OS/400 se utiliza un programa basado en InstallShield que requiere que se utilice una estación de trabajo para la interfaz con el sistema AS/400. En el apartado "Requisitos previos para la instalación en OS/400" en la página 34 encontrará información detallada.

Paso 2. Cree un perfil de usuario denominado MQAKSRV en el sistema AS/400 utilizando el mandato **CRTUSRPRF** en un indicador Control Language (CL).

Paso 3. Dependiendo de si utiliza Remote AWT o una estación de trabajo de cliente conectada para realizar la instalación, lleve a cabo los pasos siguientes:

- Si utiliza Remote AWT para realizar la instalación, lleve a cabo los pasos que se indican a continuación:
  a. Asegúrese de que Remote AWT está configurado y en ejecución. Si desea obtener más información, consulte el apartado "Utilización de Remote AWT" en la página 34.
  b. Asegúrese de que el sistema AS/400 puede acceder al archivo `installAS400.jar`. El archivo debe estar situado en IFS (sistema de archivos integrado) o en un dispositivo conectado al sistema AS/400. Si el archivo se encuentra en un dispositivo conectado, utilice el mandato de creación de enlace (**CRTLINK**) para crear un enlace simbólico al archivo.

c. Para mejorar el rendimiento del proceso de instalación, ejecute el mandato de creación de programa Java (**CRTJVAPGM**) para el archivo `installAS400.jar`.

d. Utilice el mandato de ejecución de Java (**RUNJVA**) tal como se indica a continuación, donde *n.n.n.n* representa la dirección TCP/IP de la estación de trabajo que ejecuta Remote AWT:

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

- Si utiliza una estación de trabajo cliente conectada para realizar la instalación, lleve a cabo los pasos que se indican a continuación:

  Paso a. Asegúrese de que se cumplen los requisitos especificados en el apartado "Utilización de un cliente conectado" en la página 35.

  Paso b. Asegúrese de que la opción de servidores de sistema principal está instalada y en ejecución en la máquina AS/400. Puede iniciar la opción de servidores de sistema principal utilizando el mandato de inicio de servidores de sistema principal (**STRHOSTSVR**) en un indicador de CL.

  Paso c. Asegúrese de que TCP/IP está instalado y en ejecución en la máquina AS/400. Puede iniciar TCP/IP utilizando el mandato de inicio de TCP/IP (**STRTCP**) desde un indicador de CL.

  Paso d. En la estación de trabajo, abra un indicador de mandatos y cambie al directorio `AS400` del soporte de instalación de MQSeries Adapter Kernel (red de área local o CD-ROM).

  Paso e. Entre el mandato siguiente:

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

Paso 4. Se inicia el programa de instalación y se muestra el panel **Conexión a AS/400**. Entre la dirección TCP/IP de la máquina AS/400 en el campo **Sistema:** y la contraseña y el ID de usuario en los campos correspondientes. No marque el recuadro de selección **Usuario por omisión**. Pulse en **Siguiente**.

Paso 5. Siga las indicaciones del programa de instalación. Dependiendo de la velocidad de la red y de las máquinas, el proceso de instalación puede tardar una hora, aproximadamente, en finalizar. En la estación de trabajo se muestra una barra de progreso que indica el estado de la instalación.

Se debe tener en cuenta que en OS/400, MQSeries Adapter Kernel siempre se instala en el directorio `/QIBM/ProdData/mqak` en la raíz de IFS (sistema de archivos integrado).

Paso 6. Establezca las variables de entorno CLASSPATH, PATH y QIBM_MULTI_THREADED tal como se indica a continuación:
- Añada el directorio `/QIBM/ProdData/mqak/bin` a la variable de entorno CLASSPATH.
- Añada el directorio `/QIBM/ProdData/mqak/bin` a la variable de entorno PATH.
- Establezca la variable de entorno QIBM_MULTI_THREADED en `Y`.

Paso 7. Añada la biblioteca MQAK a la lista de bibliotecas de QSYS.LIB.

La instalación del kernel ha finalizado. Una vez instalado, el kernel está configurado para ofrecer soporte para la verificación, no para el trabajo real en un sitio determinado. Verifique la instalación realizando los pasos que se indican en el apartado "Verificación de la instalación" en la página 47. Después de verificar la instalación, siga los pasos que se indican en el apartado "Conclusión de la instalación posterior" para establecer las variables de entorno y trasladar varios archivos de configuración para ofrecer soporte para el trabajo real en su sitio.

Instale el kernel en otros sistemas, según sea preciso.

Conclusión de la instalación posterior

Después de instalar el kernel, realice los pasos siguientes:

Paso 1. Decida dónde va a colocar los archivos `aqmsetup` y `aqmconfig.xml`, que se utilizan para configurar el kernel. Si desea ver más información acerca de estos archivos, consulte el apartado "Configuración del kernel" en la página 58.

PRECAUCIÓN:
Si no crea sus archivos de configuración personalizados y utiliza los archivos de configuración que se proporcionan en el directorio `samples` para el trabajo real, al instalar una nueva versión del kernel se sobrescriben dichos archivos y se destruye la configuración del trabajo real.

Paso 2. Cree un directorio para los dos archivos de configuración. No es necesario que estén situados en el mismo directorio, pero se aconseja por motivos de simplicidad. Si los coloca fuera del directorio donde ha instalado MQSeries Adapter Kernel, habrá menos directorios en caso de que desee desinstalar el kernel más adelante. El proceso de desinstalación deja directorios que contienen todo lo que no sean archivos de MQSeries Adapter Kernel originales.

Paso 3. Copie los archivos `aqmsetup` y `aqmconfig.xml` del directorio `samples` en la ubicación que desee. Puede colocarlos en una unidad de red o

en otra ubicación central a la que puedan acceder numerosos sistemas; de este modo se simplifica la actualización y la copia de seguridad de los mismos.

Si cambia el nombre del archivo `aqmconfig.xml`, el kernel no funciona correctamente. Puede redenominar el archivo `aqmsetup`, siempre que establezca una variable de entorno que lo señale de modo adecuado en el Paso 5.

Paso 4. Utilizando un editor de texto, edite el archivo `aqmsetup` para que señale al directorio adecuado del archivo `aqmconfig.xml`. Utilice un nombre de vía de acceso completamente calificado (no un nombre de vía de acceso relativo) para la ubicación del directorio. En la vía de acceso no incluya el nombre de archivo propiamente dicho. A continuación se incluye un ejemplo:

```
# Ubicación del archivo de configuración aqmconfig.xml.
AQMCONFIG=C:\Archivos de programa\MQAK\Data\
```

Incluso si desea que el archivo `aqmconfig.xml` esté en el mismo directorio que el archivo `aqmsetup`, aquí debe entrar el nombre de la vía de acceso completamente calificado. Guarde y cierre el archivo `aqmsetup`.

Paso 5. Establezca la variable de entorno AQMSETUPFILE para que señale a la ubicación del archivo `aqmsetup` (por ejemplo, `C:\Archivos de programa\MQAK\Data\aqmsetup` en los sistemas Windows, `/MQAK/data/aqmsetup` en UNIX, o `/home/`*`nombre_usuario`*`/aqmsetup` en OS/400). Tenga en cuenta que en OS/400, el archivo `aqmsetup` debe situarse siempre en el directorio home de IFS del usuario actual (es decir, `/home/`*`nombre_usuario`*`).`

Si ha instalado el kernel en una unidad de red, realice el paso que se indica a continuación para cada sistema que accede al mismo.

Paso 6. Si utiliza AIX y prevé utilizar adaptadores origen de lenguaje C nativo que se invocan desde un programa C, establezca la variable de entorno AIXTHREAD_SCOPE en el valor `S`. Para establecer esta variable de entorno en el shell Bourne o Korn, entre el siguiente mandato:

```
export AIXTHREAD_SCOPE=S
```

Para establecer esta variable de entorno en el shell C, entre el siguiente mandato:

```
setenv AIXTHREAD_SCOPE S
```

Para establecer la variable AIXTHREAD_SCOPE automáticamente al iniciar la sesión en AIX, añada este mandato a su archivo `.profile` (si utiliza el shell Bourne o Korn) o al archivo `.cshrc` (si utiliza el shell C).

Para obtener más información acerca de las políticas de planificación, consulte el paso 7 en la página 23.

Paso 7. Si fuera necesario, establezca la variable de entorno THREADS_FLAG. Sólo debe establecer esta variable si *todas* las condiciones siguientes son verdaderas:

- Utiliza el sistema operativo Solaris.
- Utiliza la versión de JDK (Java Development Kit) 1.2.2.
- Utiliza MQSeries para transportar mensajes.
- Los adaptadores origen y destino se han creado en C.

Si todas estas condiciones son verdaderas, establezca la variable de entorno THREADS_FLAG en `native`. Para establecer esta variable de entorno en el shell Bourne o Korn, entre el siguiente mandato:

```
export THREADS_FLAG=native
```

Para establecer esta variable de entorno en el shell C, entre el siguiente mandato:

```
setenv THREADS_FLAG native
```

Para establecer la variable THREADS_FLAG automáticamente al iniciar la sesión en Solaris, añada este mandato a su archivo `.profile` (si utiliza el shell Bourne o Korn) o al archivo `.cshrc` (si utiliza el shell C).

Después de completar los pasos de la instalación posterior, realice las siguientes tareas para preparar el kernel para su uso:

1. Lleve a cabo la preparación para el trabajo real. Consulte el apartado "Preparación para el trabajo real" en la página 57.
2. Edite el archivo de configuración. Si desea obtener más información, consulte el apartado "Configuración del kernel" en la página 58.
3. Configure MQSeries y el software opcional. Consulte el apartado "Configuración de MQSeries y MQSeries Integrator" en la página 92.
4. Para los sistemas reales, tenga en cuenta las "Recomendaciones de rendimiento" en la página 92.
5. Inicie el kernel. Consulte el apartado "Inicio del kernel" en la página 92.
6. Establezca un plan de mantenimiento del kernel. Consulte el apartado "Mantenimiento del kernel" en la página 95.

Verificación de la instalación

Después de instalar el kernel, verifique si está correctamente instalado ejecutando un script de verificación. El script envía un mensaje de prueba desde la aplicación de origen utilizando un adaptador origen y, a continuación, a MQSeries utilizando el kernel. Después se utiliza el kernel para recibir el mensaje desde MQSeries y se invoca un adaptador destino. Todos estos procesos se ejecutan en un solo sistema.

En esta verificación, la aplicación de origen es una cola de MQSeries denominada TEST1 y la aplicación de destino es otra cola de MQSeries denominada TEST2.

La verificación realiza las tareas siguientes:

- Verifica si el kernel, con el adaptador origen y el adaptador destino suministrados, clasifica y direcciona correctamente el mensaje de prueba, utilizando MQSeries como software de mensajería global en el sistema.
- Verifica los archivos `aqmconfig.xml` y `aqmsetup` que se han proporcionado durante la instalación, puesto que determinan la configuración del kernel. Para obtener más información acerca de estos archivos, consulte el apartado "Configuración del kernel" en la página 58.

Puede validar el archivo de configuración antes de utilizarlo para el trabajo real. Consulte el apartado "Validación del archivo de configuración" en la página 88.

En los scripts de verificación de la instalación que se proporcionan con MQSeries Adapter Kernel se da por supuesto que MQSeries se ha instalado y configurado en la máquina en la que se van a ejecutar los scripts. Si va a utilizar un software de mensajería que no sea MQSeries, puede editar los scripts de verificación de la instalación para que ofrezcan soporte para el software de envío de mensajes, tal como se indica a continuación:

1. Cambie al directorio `verification` de la instalación del kernel.
2. Abra el archivo `aqmconfig.xml` en un editor de texto y cambie la línea `<epicmqppqueuemgr>DEFAULT</epicmqppqueuemgr>` por `<epicmqppqueuemgr>`*`nombre_gestor_colas`*`</epicmqppqueuemgr>`, donde *nombre_gestor_colas* es el nombre del gestor de colas.
3. Edite el archivo `aqmverifyinstall` tal como se indica a continuación:
   - Si realiza la verificación de la instalación en un sistema Windows, abra el archivo `aqmverifyinstall.bat` en un editor de texto y cambie la línea `aqmcreateq TEST2` por `aqmcreateq TEST2` *`nombre_gestor_colas`*, donde *nombre_gestor_colas* es el nombre del gestor de colas.
   - Si realiza la verificación de la instalación en UNIX u OS/400, abra el archivo `aqmverifyinstall.sh` en un editor de texto y cambie la línea

`aqmcreateq.sh TEST2` por `aqmcreateq.sh TEST2` *`nombre_gestor_colas`*, donde *nombre_gestor_colas* es el nombre del gestor de colas.

En esta verificación se utilizan algunos componentes como, por ejemplo, un nombre de adaptador destino, `com.ibm.epic.adapters.eak.test.InstallVerificationTest`, que no forman parte del kernel. Se proporcionan con el kernel sólo con la finalidad de verificar la instalación.

Cuando finaliza la verificación, el daemon del adaptador de verificación se detiene.

Durante la verificación, el rastreo no está habilitado.

Procedimiento de verificación

Paso 1. La verificación crea y utiliza tres colas de MQSeries. Si las colas contienen mensajes antes de realizar la verificación, la verificación no se ejecuta correctamente. Borre los mensajes de las colas siguientes:
- TEST2AIQ
- TEST2AEQ
- TEST2RPL

Paso 2. Asegúrese de que está autorizado para instalar y verificar el kernel. Consulte el apartado "Preparación para la instalación" en la página 40.

Paso 3. Inicie la verificación tal como se indica a continuación:
- En los sistemas Windows, efectúe una doble pulsación en el archivo `aqmverifyinstall.bat` del directorio `verification` o, si lo prefiere, abra un indicador de mandatos, cambie al directorio `verification` y ejecute `aqmverifyinstall.bat`.
- En UNIX, abra un terminal, cambie al directorio `verification` y ejecute el archivo `aqmverifyinstall.sh`.
- En OS/400, realice los pasos siguientes:
  a. Inicie una sesión **qsh** entrando el mandato **STRQSH**.
  b. Copie el archivo `/QIBM/ProdData/mqak/verification/aqmsetup` en el directorio home (`/home/`*`nombre_usuario`*).
  c. Cambie al directorio `/QIBM/ProdData/mqak/verification`.
  d. Ejecute el archivo `aqmverifyinstall.sh`.

  El archivo `aqmverifyinstall` contiene comentarios sobre su funcionamiento.

Paso 4. El mensaje `La Prueba de verificación de la instalación ha finalizado correctamente` indica que el resultado ha sido satisfactorio. Si fuera necesario, cierre la ventana de verificación.

Paso 5. En caso de que se haya producido alguna anomalía, examine la ventana de verificación y el archivo de anotaciones, `EpicSystemExceptionFile`*nnnnnnnn*`.log`, para determinar el error.

Paso 6. Consulte el apartado "Problemas habituales de la verificación", donde se explican los problemas habituales que se pueden producir durante la verificación y se proporcionan las posibles respuestas.

Paso 7. Si lo desea, puede realizar una verificación opcional. Para obtener más información, consulte el apartado "Verificación opcional" en la página 51.

Paso 8. Vuelva al procedimiento de instalación y configure el kernel para que ofrezca soporte para el funcionamiento en su sitio específico. Vaya al Paso 1 en la página 44.

Problemas habituales de la verificación

En este apartado se enumeran los problemas habituales que pueden producirse durante la verificación, junto con sus posibles soluciones. La información importante de los mensajes de excepción aparece resaltada en **negrita**.

Problema: No se ha encontrado el archivo `aqmsetup`.

Respuesta: Asegúrese de que la variable de entorno AQMSETUPFILE se ha establecido en la ubicación del archivo `aqmsetup` en el directorio `verification`.

Mensaje de excepción:

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

Problema: No se ha encontrado el archivo `aqmconfig.xml`.

Respuesta: Edite el archivo `aqmsetup` del directorio `verification` y asegúrese de que la entrada `AQMCONFIG=` señala al directorio `verification`. Utilice un nombre de vía de acceso completamente calificado. Asegúrese también de que el archivo `aqmconfig.xml` está en el directorio `verification`.

Mensaje de excepción:

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

Problema: La cola en la que desea situar el mensaje no existe.

Respuesta: Utilice MQSeries para asegurarse de que la cola denominada en el mensaje de excepción (TEST2AIQ cuando se verifica la instalación) existe y puede aceptar mensajes. Consulte el apartado "Creación de colas de MQSeries" en la página 97.

Mensaje de excepción:

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

Problema: No se ha encontrado el adaptador destino.

Respuesta: Asegúrese de que existe el adaptador destino que se ha especificado en el mensaje: `com.ibm.epic.adapters.eak.test.InstallVerificationTest`. Asegúrese de que la variable de entorno CLASSPATH incluye el directorio `bin` del kernel.

Mensaje de excepción:

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

Problema: No se ha encontrado ningún adaptador en el que cargar el mensaje para la entrega. El identificador lógico de destinación no tiene una entrada en el archivo `aqmconfig.xml` para el tipo y la categoría de cuerpo que se ha especificado en el mensaje de la cola.

Respuesta: Durante la verificación, la causa más probable de este mensaje de excepción es la existencia de mensajes en una cola denominada TEST2AIQ antes de la verificación. Borre todos los mensajes de la cola TEST2AIQ y vuelva a intentar la verificación. La

única entrada para un nombre de clase de mandato para la aplicación TEST2 en el archivo `aqmconfig.xml` del directorio `verification` es para un tipo de cuerpo TESTBOD y una categoría de cuerpo OAG.

Mensaje de excepción:

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

Problema: El gestor de colas de verificación no se ha iniciado.

Respuesta: Asegúrese de que el gestor de colas de MQSeries por omisión se ha iniciado correctamente.

Mensaje de excepción:

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

Problema: Se ha producido un error general de MQSeries.

Respuesta: Asegúrese de que MQSeries está correctamente instalado y configurado y que está en ejecución en la máquina. Examine el código de razón de MQException y utilice el documento *MQSeries Messages* para determinar la causa del código de razón.

Mensaje de excepción:

```
Received MQException "ACTION ATTEMPTED." Message information:
completion code <código_terminación> reason code
<código_razón>
```

Verificación opcional

Después de verificar que el kernel está correctamente instalado en el primer sistema, si lo desea, puede realizar los pasos siguientes:

1. Verifique si el kernel está correctamente instalado en un segundo sistema, utilizando la misma verificación.
2. Verifique si puede enviar un mensaje de prueba de un adaptador origen de un sistema a un adaptador destino de otro sistema. Configure y lleve a cabo manualmente esta verificación. Si elige desarrollar esta verificación modificando los archivos de verificación originales que se proporcionan con el kernel, guarde una copia de los archivos de verificación originales como copia de seguridad.

Utilización de la instalación silenciosa

MQSeries Adapter Kernel se debe instalar en todas las plataformas utilizando la *instalación silenciosa*. La instalación silenciosa le permite omitir el programa de instalación de MQSeries Adapter Kernel, donde debe seleccionar manualmente las opciones de instalación que desea. La instalación silenciosa resulta útil cuando desea instalar la configuración por omisión en múltiples máquinas.

Para instalar el kernel de forma silenciosa, lleve a cabo los siguientes pasos específicos del sistema operativo:

En sistemas Windows:

Paso 1. Abra un indicador de mandatos y cambie al directorio que contiene los archivos de instalación de MQSeries Adapter Kernel.

Paso 2. Entre el mandato siguiente:

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

donde *ubicación_instalación* es la ubicación deseada para la instalación (por ejemplo, `D:\mqak`).

En UNIX:

Paso 1. En un terminal, cambie al directorio que contiene los archivos de instalación de MQSeries Adapter Kernel. Si va a realizar la instalación desde el CD-ROM, inserte el CD-ROM de MQSeries Adapter Kernel en la unidad de CD-ROM y, si fuera necesario, monte la unidad de CD-ROM siguiendo las instrucciones que se proporcionan en la documentación del sistema operativo.

Paso 2. Entre el siguiente mandato:

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

donde *ubicación_instalación* es la ubicación deseada para la instalación (por ejemplo, `/opt/mqak`).

En OS/400:

Si no utiliza un cliente conectado para acceder a la máquina AS/400, lleve a cabo los siguientes pasos:

Paso 1. Asegúrese de que el sistema AS/400 puede acceder al archivo `installAS400.jar`. El archivo debe estar situado en IFS (sistema de archivos integrado) o en un dispositivo conectado al sistema AS/400.

Si el archivo se encuentra en un dispositivo conectado, utilice el mandato de creación de enlace (**CRTLINK**) para crear un enlace simbólico al archivo.

Paso 2. Para mejorar el rendimiento del proceso de instalación, ejecute el mandato de creación de programa Java (**CRTJVAPGM**) para el archivo `installAS400.jar`.

Paso 3. Según si utiliza un indicador CL o una sesión **qsh**, entre uno de los siguientes mandatos:

- Si utiliza un indicador CL, entre el siguiente mandato:

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

- Si utiliza una sesión **qsh**, entre el siguiente mandato:

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

Si utiliza un cliente conectado para tener interfaz con el sistema AS/400, lleve a cabo los siguientes pasos:

Paso 1. Asegúrese de que se cumplen los requisitos especificados en el apartado "Utilización de un cliente conectado" en la página 35.

Paso 2. Asegúrese de que la opción de servidores de sistema principal está instalada y en ejecución en la máquina AS/400. Puede iniciar la opción de servidores de sistema principal utilizando el mandato de inicio de servidores de sistema principal (**STRHOSTSVR**) en un indicador de CL (Control Language).

Paso 3. Asegúrese de que TCP/IP está instalado y en ejecución en la máquina AS/400. Puede iniciar TCP/IP utilizando el mandato de inicio de TCP/IP (**STRTCP**) desde un indicador de CL.

Paso 4. En la estación de trabajo, abra un indicador de mandatos y cambie al directorio `AS400` del soporte de instalación de MQSeries Adapter Kernel (red de área local o CD-ROM).

Paso 5. Entre el siguiente mandato:

```
java -cp installAS400.jar run -silent -os400 nombre_máquina ID_usuario
contraseña
```

donde *nombre_máquina* es la dirección TCP/IP del sistema AS/400, *ID_usuario* es su ID de usuario y *contraseña* es su contraseña.

Actualización del kernel

Si ha instalado MQSeries Adapter Kernel versión 1.0, con o sin CSD (Corrective Service Diskette), o MQSeries Adapter Kernel versión 1.1 con un nivel de modificación anterior, realice los pasos siguientes antes de instalar MQSeries Adapter Kernel versión 1.1 con el nivel de modificación actual:

Paso 1. Haga una copia de seguridad de los archivos `aqmsetup` y `aqmconfig` (`aqmconfig.properties` o `aqmconfig.xml`) en una ubicación externa al directorio de instalación de MQSeries Adapter Kernel.

Paso 2. Si ha instalado un CSD de MQSeries Adapter Kernel, desinstálelo tal como se indica a continuación:

- En Windows NT, utilice uno de los métodos siguientes:
  - En el menú Inicio de Windows NT, pulse en **Programas > MQSeries Adapter Kernel > Quitar CSD**.
  - Use el programa de utilidad Agregar/Quitar programas del Panel de control.
  - Ejecute el archivo `aqmuninstallCSD.bat` situado en el directorio raíz del kernel.
  - Abra un indicador de mandatos, cambie al directorio raíz del kernel y entre el siguiente mandato:

    ```
    java uninstallCSD
    ```
- En AIX, cambie al directorio raíz del kernel y entre uno de los mandatos siguientes:

  ```
  aqmuninstallCSD.sh
  java uninstallCSD
  ```

Paso 3. Desinstale MQSeries Adapter Kernel tal como se indica a continuación:

- En Windows NT, utilice uno de los métodos siguientes:
  - En el menú Inicio de Windows NT, pulse en **Programas > MQSeries Adapter Kernel > Desinstalar MQSeries Adapter Kernel**.
  - Use el programa de utilidad Agregar/Quitar programas del Panel de control.
  - Ejecute el archivo `aqmuninstall.bat` situado en el directorio raíz del kernel.
  - Abra un indicador de mandatos, cambie al directorio raíz del kernel y entre el siguiente mandato:

    ```
    java uninstall
    ```
- En AIX, cambie al directorio raíz del kernel y entre uno de los mandatos siguientes:

  ```
  aqmuninstall.sh
  java uninstall
  ```

Paso 4. Instale MQSeries Adapter Kernel versión 1.1. Si desea obtener más información, consulte el apartado “Instalación del kernel” en la página 41.

Paso 5. Restaure los archivos `aqmsetup` y `aqmconfig` en sus ubicaciones anteriores en el directorio de instalación de MQSeries Adapter

Kernel. Si es necesario, convierta el archivo `aqmconfig.properties` en un archivo `aqmconfig.xml` . Para obtener más información acerca del archivo `aqmconfig.xml`, consulte el apartado "El archivo de configuración" en la página 65.

Eliminación del kernel

Existen numerosos procedimientos para eliminar el kernel. Tenga en cuenta que el proceso de desinstalación no elimina ningún archivo o directorio creado después de la instalación del kernel, lo que incluye todos los archivos de anotaciones y los archivos de datos que ha copiado el usuario.

- En los sistemas Windows, utilice uno de los métodos siguientes:
  - En el menú Inicio, pulse en **Programas > IBM MQSeries Adapter Kernel > Desinstalar MQSeries Adapter Kernel**.
  - Use el programa de utilidad Agregar/Quitar programas del Panel de control.
  - Ejecute el archivo `aqmuninstall.bat` situado en el directorio raíz del kernel.
  - Para desinstalar el kernel de forma silenciosa (es decir, sin que el programa de desinstalación le solicite detalles ni confirmación), abra un indicador de mandatos, cambie al directorio de instalación del kernel y entre el siguiente mandato:

    ```
    java -cp uninstall.jar run -silent
    ```
- En UNIX, cambie al directorio raíz del kernel y entre el siguiente mandato:

  ```
  aqmuninstall.sh
  ```

  Para desinstalar el kernel de forma silenciosa (es decir, sin que el programa de desinstalación le solicite detalles ni confirmación), cambie al directorio raíz del kernel y entre el siguiente mandato:

  ```
  java -cp uninstall.jar run -silent
  ```
- En OS/400, utilice uno de los métodos siguientes para desinstalar el kernel:
  - Si utiliza Remote AWT para desinstalar el kernel, lleve a cabo los pasos que se indican a continuación:

    Paso 1. Asegúrese de que Remote AWT está configurado y en ejecución. Si desea obtener más información, consulte el apartado "Utilización de Remote AWT" en la página 34.

    Paso 2. Para mejorar el rendimiento del proceso de desinstalación, ejecute el mandato de creación de programa Java (**CRTJVAPGM**) para el archivo `/QIBM/ProdData/mqak/uninstall/uninstall.jar`.

    Paso 3. Utilice el mandato de ejecución de Java (**RUNJVA**) tal como se indica a continuación, donde *n.n.n.n* representa la dirección TCP/IP de la estación de trabajo que ejecuta Remote AWT:

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

– Si utiliza una estación de trabajo cliente conectada para desinstalar el kernel, lleve a cabo los pasos que se indican a continuación:

  Paso 1. Asegúrese de que se cumplen los requisitos especificados en el apartado "Utilización de un cliente conectado" en la página 35.

  Paso 2. Asegúrese de que la opción de servidores de sistema principal está instalada y en ejecución en la máquina AS/400. Puede iniciar la opción de servidores de sistema principal utilizando el mandato de inicio de servidores de sistema principal (**STRHOSTSVR**) en un indicador de CL (Control Language).

  Paso 3. Asegúrese de que TCP/IP está instalado y en ejecución en la máquina AS/400. Puede iniciar TCP/IP utilizando el mandato de inicio de TCP/IP (**STRTCP**) desde un indicador de CL.

  Paso 4. Copie los archivos `uninstall.jar` y `uninstall.dat` del directorio `/QIBM/ProdData/mqak/uninstall` del sistema AS/400 en un directorio de la estación de trabajo cliente.

  Paso 5. Entre el siguiente mandato:

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

  Para desinstalar el kernel de forma silenciosa (es decir, sin que el programa de desinstalación le solicite detalles ni confirmación), entre el siguiente mandato:

  ```
  java -cp uninstall.jar run -silent -os400 nombre_máquina ID_usuario
  contraseña
  ```

  donde *nombre_máquina* es la dirección TCP/IP del sistema AS/400, *ID_usuario* es su ID de usuario y *contraseña* es su contraseña.

– Si trabaja directamente en el sistema AS/400 en un indicador CL o sesión **qsh** y desea desinstalar el kernel de forma silenciosa (es decir, sin que el programa de desinstalación le solicite detalles ni confirmación), entre uno de los siguientes mandatos:

  - En un indicador CL:

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

  - En una sesión **qsh**:

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

Capítulo 4. Utilización del kernel

Este capítulo contiene la información siguiente acerca de la utilización del kernel:

Preparación para el trabajo real

Antes de utilizar el kernel para el trabajo real, lleve a cabo las tareas siguientes:

1. Diseñe la arquitectura global del sistema, que debe incluir la Oferta MQSeries Adapter, MQSeries u otro software de envío de mensajes y, si lo desea, MQSeries Integrator, dependiendo de las condiciones y los requisitos de su sitio. Por lo general, la arquitectura es exclusiva de cada sitio.
2. Cree los adaptadores origen y destino necesarios utilizando MQSeries Adapter Builder y, a continuación, pruébelos y difúndalos.
3. Desarrolle interfaces específicas de la aplicación fuera de la Oferta MQSeries Adapter con las finalidades siguientes:
   - Para permitir que el adaptador origen obtenga los datos de aplicación de la aplicación de origen
   - Para permitir que la aplicación de destino obtenga los datos de mensaje del adaptador destino

   La naturaleza exacta de la interfaz específica de la aplicación depende de las características de la aplicación de origen y de la aplicación de destino. Algunos ejemplos de interfaces específicas de aplicación incluyen:
   - Llamadas API y salidas de usuario
   - Lecturas y escrituras de archivo
   - Activaciones de base de datos
   - Colas de mensajes
4. Configure el kernel para que ofrezca soporte para el flujo de ejecución: envío, direccionamiento, rastreo y entrega de mensajes. Para obtener más

© Copyright IBM Corp. 2000, 2001

información acerca de la configuración del kernel, consulte el apartado "Configuración del kernel".

5. Configure MQSeries u otro software de envío de mensajes y, si lo desea, MQSeries Integrator para ofrecer soporte para la arquitectura global de su sistema. Consulte el apartado "Configuración de MQSeries y MQSeries Integrator" en la página 92.
6. Si fuera necesario, desarrolle clases de inicio de sesión Java para ofrecer soporte para la entrega de mensajes. Dichas clases son específicas para cada aplicación de destino. Sólo se necesitan si el adaptador destino requiere información para iniciar la sesión y conectar a la aplicación.
7. Pruebe el sistema completo, es decir, MQSeries Adapter Kernel con los adaptadores origen y destino, las interfaces específicas de la aplicación y el código personalizado, antes de utilizarlo para el trabajo real.
8. Difunda el sistema en el entorno de trabajo real.
9. Active el kernel iniciando uno o más daemons de adaptador y, si lo desea, los servidores de rastreo. Asegúrese de que se ha iniciado la aplicación de origen. Si se ejecuta el adaptador origen en el proceso de la aplicación de origen, el adaptador origen se inicia automáticamente con la aplicación de origen, es decir, no es necesario realizar pasos adicionales para iniciar el adaptador origen y, por este motivo, se deben iniciar todos los daemons o servidores que contengan el adaptador origen. Consulte el apartado "Inicio del kernel" en la página 92.

Configuración del kernel

Este apartado trata de la configuración del kernel para la utilización en el entorno del cliente. "Visión general de la configuración" ofrece una visión general conceptual de la configuración del kernel. "Archivos implicados en el arranque y la configuración" en la página 63 trata de los diferentes archivos que definen la configuración de MQSeries Adapter Kernel. "El archivo de instalación" en la página 64 trata del archivo `aqmsetup`, que define varios valores de configuración inicial del kernel. "El archivo de configuración" en la página 65 trata del archivo `aqmconfig.xml`, que ofrece al kernel información de configuración específica como, por ejemplo, los nombres de las aplicaciones de origen y de destino, adaptadores origen y destino, colas y gestores de colas, modalidades de comunicación y especificaciones de anotaciones cronológicas y seguimiento.

Visión general de la configuración

Este apartado ofrece una visión general conceptual de la configuración del kernel. Es importante comprender el flujo de ejecución del kernel antes de proceder a su configuración. Este apartado trata del flujo de ejecución en un nivel simplificado. Para obtener más información sobre el flujo de ejecución, consulte el apartado "Flujo de ejecución" en la página 14.

En el nivel más básico, la configuración de MQSeries Adapter Kernel está controlada por los datos que fluyen entre las aplicaciones. La configuración también debe tener en cuenta los siguientes factores:

- Las aplicaciones que reciben los datos.
- Los adaptadores que se necesitan en la parte origen y los adaptadores destino, daemons de adaptador y trabajos que se necesitan en la parte destino.
- Las modalidades de comunicación, los formatos de clasificación y los mecanismos de transporte que se utilizan.

Las estructuras de datos y el formato de datos son diferentes para cada aplicación de la configuración. Por ejemplo, si una configuración incluye dos aplicaciones, A y B, que envían datos de pedidos de compra a la aplicación C, es posible que los datos de la aplicación A tengan un formato diferente y diferentes significados de código de los datos de la aplicación B. Para evitar que la aplicación C tenga que reconocer y analizar los dos flujos diferentes de datos de las dos diferentes aplicaciones, los datos de cada aplicación se convierten en un *mensaje de integración* que se encuentra en un *formato neutro de integración*. Generalmente, el formato neutro de integración es un estándar de la industria que se basa en XML. El único formato de datos que la aplicación C debe reconocer y analizar es el formato neutro de integración.

La Figura 3 en la página 60 muestra el flujo de datos desde las aplicaciones A y B a las aplicaciones C y D. A continuación de la figura aparece una explicación de los diferentes flujos de datos que presenta.

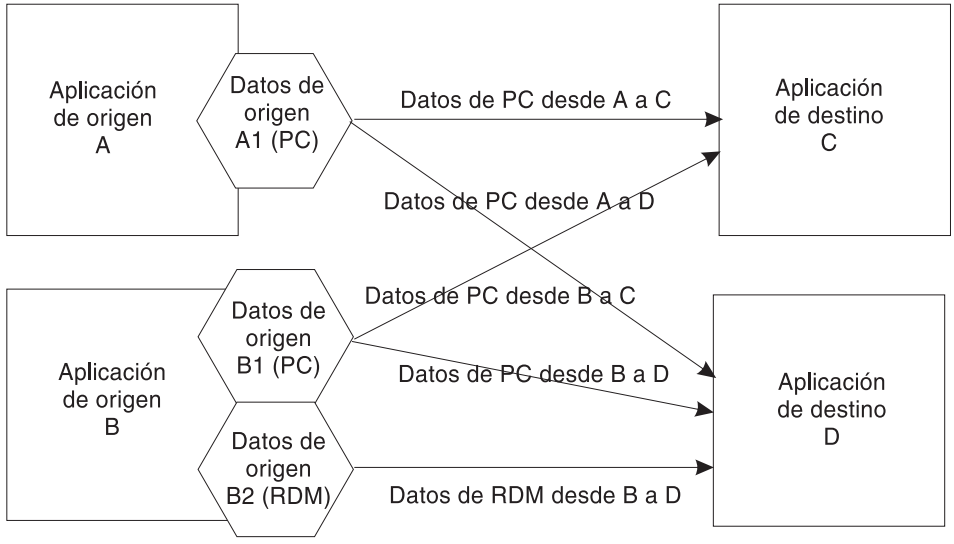


Figura 3. Aplicaciones conectadas por flujos de datos en una configuración sencilla

En la Figura 3, los datos de pedidos de compra de la aplicación A pasan a las aplicaciones C y D, los datos de pedidos de compra de la aplicación B también pasan a las aplicaciones C y D, y los datos de la relación de material de la aplicación B pasan sólo a la aplicación D. En todos los casos, los datos se convierten en formato estándar de la industria antes de ser enviados a las aplicaciones de destino. En el caso de los datos de pedido de compra que provienen de las aplicaciones A y B, los datos se convierten en un formato XML estándar que representa los datos de pedido de compra. En el caso de datos de relación de material que provienen de la aplicación B, los datos se convierten en un formato XML estándar que representa los datos de relación de material.

Se utiliza un transporte de comunicaciones como MQSeries o una implementación de Java Message Service (JMS) para enviar los datos a la aplicación o aplicaciones de destino. El mensaje de integración se convierte en el *formato de clasificación* que necesita el transporte de comunicaciones específico y, a continuación, se entrega al transporte de comunicaciones (por ejemplo, una cola MQSeries). Todas las aplicaciones de destino pueden utilizar un transporte de comunicaciones y formato de clasificación diferentes para recibir mensajes. Por ejemplo, la aplicación C puede utilizar MQSeries para recibir mensajes y la aplicación D puede utilizar JMS, tal como aparece en la Figura 4 en la página 61. En este caso, todos los mensajes de integración que van a la aplicación C (es decir, los datos de pedido de compra de las aplicaciones A y B) se convierten en un formato de clasificación MQSeries y todos los mensajes de integración que van a la aplicación D (es decir, los datos de pedido de compra de las aplicaciones A y B además de los datos de

relación de material de la aplicación B) se convierten en un formato de clasificación JMS. MQSeries Adapter Kernel utiliza los siguientes mecanismos para realizar estas conversiones:

- Se utiliza un *adaptador origen* para convertir los datos de aplicación en un mensaje de integración. Los adaptadores origen se crean en MQSeries Adapter Builder.
- Se utiliza el *adaptador nativo* para convertir el mensaje de integración en un *mensaje de comunicación*. El adaptador nativo utiliza el *servicio de mensaje lógico* (LMS) para convertir el mensaje que debe transportar el transporte de comunicaciones; el LMS es específico del transporte de comunicaciones que se utiliza. A continuación, el LMS utiliza un *formateador* para clasificar el mensaje en el transporte.

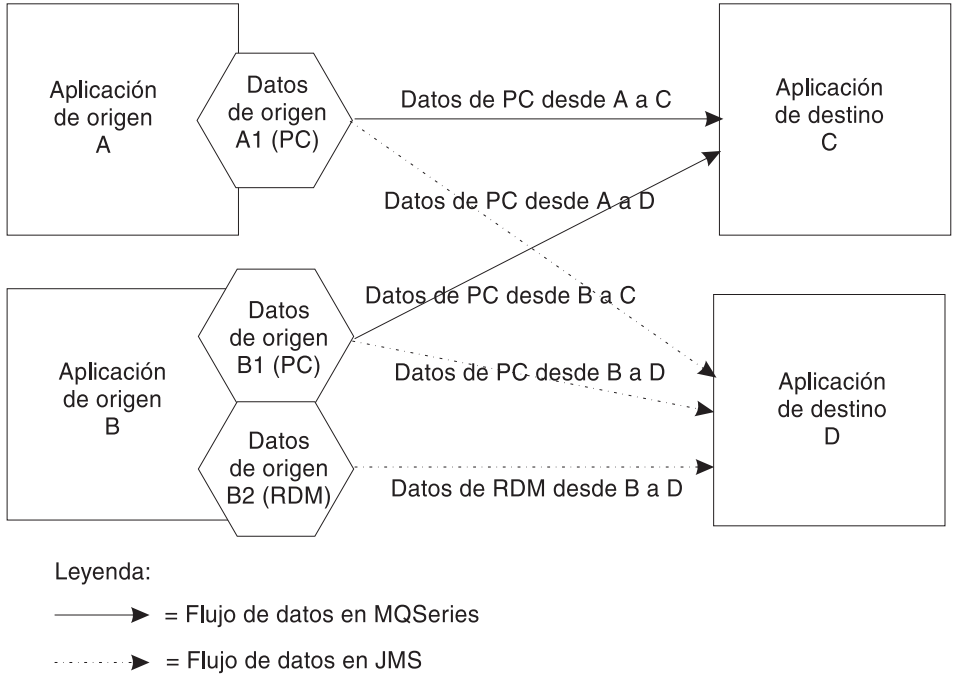


Figura 4. Aplicaciones conectadas por diferentes transportes de comunicación en una configuración sencilla

El flujo de datos desde la aplicación al mensaje de integración y al mensaje de comunicación aparece en la Figura 5 en la página 62 y la Figura 6 en la página 62. Cuando se recibe un mensaje de comunicación en el destino, las conversiones se invierten: el adaptador nativo convierte el mensaje de comunicación en el mensaje de integración y, a continuación, si es necesario, el

adaptador destino convierte el mensaje de integración en el formato de datos que necesita la aplicación de destino.

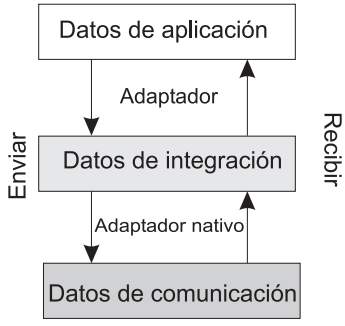


Figura 5. Conversión de datos

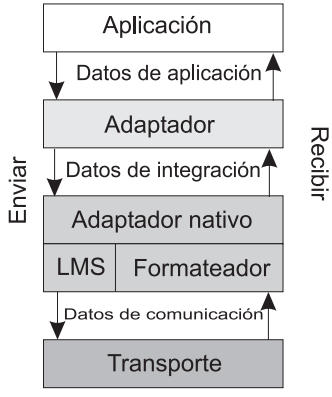


Figura 6. Flujo de datos

Todos los puntos por los que pasan los datos deben estar representados en el archivo de configuración del kernel. Existen tres requisitos de configuración lógica que se dividen por identificador de aplicación (origen o destino). Un identificador de aplicación puede ser para una aplicación de origen o de destino, según si los mensajes van a la aplicación (destino) o vienen de la aplicación (origen). El archivo de configuración debe incluir los siguientes tipos de información:

- Comunicaciones
  - Donde deben ir los datos
  - El transporte de comunicaciones que se utiliza
  - El método de clasificación de comunicación (formateador) que se utiliza
  - Los requisitos de comunicación subyacentes como, por ejemplo, un gestor de colas MQSeries y nombres de cola MQSeries, o una factoría de conexión de colas JMS y nombres de cola JMS

- Adaptadores (sólo para la parte destino)
  - Los adaptadores que se necesitan para procesar los datos
  - El tipo de adaptadores que se utilizan (EAB o EJB)
  - Información adicional sobre el tipo de adaptador específico que se utiliza
  - Para MQSeries Adapter Kernel autónomo, el daemon del adaptador y la información del trabajo
- Otros
  - Especificaciones de seguimiento
  - Especificaciones de anotaciones cronológicas

El flujo de datos a medida que se relaciona con las diferentes partes de la configuración, tal como aparece en la Figura 7.

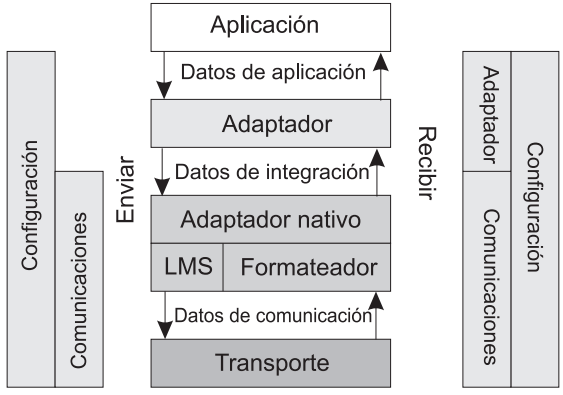


Figura 7. Flujo de datos relacionados con la configuración

Consulte el apartado “Sintaxis y organización del archivo de configuración” en la página 66 donde encontrará información detallada sobre la correlación de estos requisitos de configuración con elementos XML en el archivo `aqmconfig.xml`, que controla estos aspectos de la configuración del kernel. El apartado “Configuraciones comunes” en la página 78 enumera varias configuraciones comunes.

Archivos implicados en el arranque y la configuración

Varios archivos personalizables determinan la configuración del kernel. Utilizando un editor de texto estándar, edite los archivos para configurar el kernel para su sitio. La configuración del kernel implica los archivos siguientes:

- El archivo `aqmsetenv.bat` (sistemas Windows) o `aqmsetenv.sh` (UNIX), que establece las variables de entorno. Si lo desea, edite este archivo para cambiar las variables de entorno del sistema después de la instalación. Las variables de entorno que establece este archivo incluyen PATH,

CLASSPATH y LIBPATH. En los sistemas Windows, el programa de instalación establece estas variables automáticamente. Para establecer estas variables automáticamente al iniciar la sesión en UNIX, añada los valores especificados en el archivo `aqmsetenv.sh` en el archivo `.profile` (si utiliza el shell Bourne o Korn) o el archivo `.cshrc` (si utiliza el shell C).

Para obtener más información acerca del establecimiento de las variables de entorno adecuadas en OS/400, consulte el Paso 6 en la página 44.

- El archivo `aqmsetup`, que proporciona numerosos valores de instalación inicial para el kernel. Para obtener más información, consulte el apartado "El archivo de instalación".
- El archivo `aqmconfig.xml`, que configura el kernel. Para obtener más información, consulte el apartado "El archivo de configuración" en la página 65. Este archivo contiene la mayor parte de los valores para la configuración del kernel.
- El archivo `aqmcreateq.bat` (sistemas Windows) o `aqmcreateq.sh` (UNIX y OS/400), que es un script que crea colas de MQSeries. Consulte el apartado "Creación de colas de MQSeries" en la página 97.

Todos estos archivos incluyen comentarios que le pueden ayudar a editarlos.

Se recomienda hacer una copia de seguridad de estos archivos. Para obtener más información, consulte el apartado "Mantenimiento del kernel" en la página 95.

El archivo de instalación

El archivo de instalación, `aqmsetup`, controla muchos de los valores iniciales del kernel, entre los que se incluyen los siguientes:

- La ubicación del archivo de configuración. Consulte el apartado "El archivo de configuración" en la página 65.
- La ubicación de DTD XML, si no se encuentran en el directorio actual.
- Las variables de entorno JNI (Java Native Interface) para la interfaz C, para cambiar la cantidad de memoria utilizada. Se aplica cuando un módulo ejecutable C inicia un proceso y dicho proceso convierte en instancia una máquina virtual de Java. En este caso, se puede controlar el uso de memoria eliminando la marca de comentario y modificando las líneas siguientes del archivo `aqmsetup` :

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

Todos los tamaños están expresados en bytes.

En el "Apéndice E. Ejemplo del archivo de instalación" en la página 125 se proporciona un archivo `aqmsetup` de ejemplo, que también se incluye en el directorio `samples` de la instalación de MQSeries Adapter Kernel.

Si fuera necesario, edite el archivo de instalación al instalar MQSeries Adapter Kernel por primera vez. Después de la instalación, edite el archivo sólo si el kernel localiza un problema de falta de memoria de Java, tal como se ha explicado en la lista anterior.

El archivo de configuración

En este apartado se trata el archivo `aqmconfig.xml`, que determina la configuración del kernel. El apartado "Sintaxis y organización del archivo de configuración" en la página 66 ofrece información sobre la estructura del archivo de configuración. Además, en el apartado "Edición del archivo de configuración" en la página 86 se facilitan sugerencias prácticas para editarlo.

Un archivo XML denominado `aqmconfig.xml` determina la configuración de MQSeries Adapter Kernel. En el "Apéndice D. Ejemplo del archivo de configuración" en la página 119 se proporciona un archivo de configuración de ejemplo, que también se incluye en el directorio `samples` de la instalación de MQSeries Adapter Kernel.

Los valores especificados en el archivo de configuración controlan los elementos siguientes del kernel:

- Identificadores lógicos de origen
- Identificadores lógicos de destino
- Daemons de adaptador e información del trabajo en la parte destino
- Clientes de rastreo
- Servidores de rastreo
- Clasificación y direccionamiento de mensajes, que determinan las especificaciones siguientes:
  - Los nombres de las colas de recepción, de error y de respuestas
  - Uno o más destinos por omisión a los que enviar mensajes
  - El nombre del gestor de colas MQSeries o de la factoría de conexión de colas JMS que obtiene o envía el mensaje
  - El tiempo de espera para recibir mensajes o respuestas
  - La clase del adaptador destino de la parte destino del kernel que procesa cada mensaje
  - Información adicional específica del adaptador destino
  - El número mínimo de trabajos de la parte destino (si se ejecuta MQSeries Adapter Kernel autónomo)
  - La habilitación o inhabilitación del rastreo y el control del nivel de rastreo

  - La habilitación o inhabilitación de las anotaciones cronológicas de seguimiento
- Modalidad de comunicación

Sintaxis y organización del archivo de configuración

Puesto que la configuración de MQSeries Adapter Kernel se basa en LDAP (Lightweight Directory Access Protocol), la estructura del archivo de configuración duplica LDAP. El elemento XML de nivel superior, `Epic`, representa el nivel superior del directorio LDAP, y elementos XML anidados en el elemento de nivel superior representan los objetos LDAP. Algunos elementos XML tienen atributos obligatorios que representan información de LDAP. Los valores se añaden a la configuración como contenido de los elementos o como atributos de los elementos. Un ejemplo de un valor de configuración asignado como contenido de un elemento puede ser `<epictracelevel>-1</epictracelevel>`, que asigna el valor `-1` (todos los mensajes posibles) al elemento `epictracelevel`. Un ejemplo de un valor de configuración asignado como un atributo de un elemento puede ser `<ePICTraceHandler epictracehandler="com.ibm.logging.ConsoleHandler">`, que asigna la clase `com.ibm.logging.ConsoleHandler` que se debe utilizar como manejador de rastreo.

A continuación se incluye una lista con la descripción de los elementos de alto nivel que se utilizan en el archivo de configuración. En el apartado "Elementos XML que se utilizan en el archivo de configuración" en la página 69 se enumeran y describen todos los elementos que se utilizan en el archivo de configuración. En el archivo de configuración de ejemplo se incluyen ejemplos del modo en que se utilizan los diferentes elementos en el contexto.

- `Epic`—Elemento de nivel superior que se necesita para el archivo `aqmconfig.xml`.
- `ePICApplications`—Elemento dependiente obligatorio de `Epic`.
- `ePICApplication`—Elemento dependiente obligatorio de `ePICApplications`. Enumera y define las aplicaciones a las que debe atender el kernel. Se necesita un elemento `ePICApplication` completamente definido (incluidos los elementos dependientes) para cada aplicación.
- `AdapterRouting`—Elemento dependiente opcional de `ePICApplication`. Define el gestor de colas y la información relacionada.
- `ePICBodyCategory`—Elemento dependiente obligatorio de `AdapterRouting`. Establece la categoría de cuerpo para los mensajes que direcciona el kernel.
- `ePICBodyType`—Elemento dependiente obligatorio de `ePICBodyCategory`. Establece el tipo de cuerpo de los mensajes que direcciona el kernel. Contiene definiciones para los elementos como, por ejemplo, definiciones de mensajes, modalidades de comunicación para recibir mensajes y formateadores de mensajes.

- `ePICAdapterDaemonExtensions`—Elemento dependiente opcional de `ePICApplication` que representa un daemon del adaptador. Contiene información relacionada con los daemons del adaptador, que incluye los identificadores de aplicación y los trabajos del adaptador.
- `ePICTraceExtensions`—Elemento dependiente opcional de `ePICApplication` que representa una aplicación cliente de rastreo o un elemento del servidor de rastreo. Define información relacionada con el rastreo.

La Figura 8 en la página 68 muestra la estructura de alto nivel del archivo de configuración. No es un ejemplo del funcionamiento de un archivo de configuración. Su finalidad es, simplemente, demostrar las relaciones y las dependencias entre los elementos de alto nivel. En el "Apéndice D. Ejemplo del archivo de configuración" en la página 119 se proporciona un ejemplo completo.

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

Figura 8. Estructura de alto nivel del archivo de configuración

A continuación, se incluye una lista con la descripción de todos los elementos que se utilizan en el archivo de configuración. Si se indica que un elemento tiene un valor por omisión, el kernel utiliza dicho valor si un elemento de la configuración necesita un valor que no se ha especificado de modo explícito.

Elementos XML que se utilizan en el archivo de configuración

Epic Elemento de nivel superior para el archivo de configuración.

Elementos dependientes:

- `context`
- `ePICApplications` (obligatorio)

Atributos: `o="ePIC"` (obligatorio)

context
Especifica la raíz de FSContext (contexto del sistema de archivos) de JNDI (Java Naming and Directory Interface) cuando se utilizan objetos JMS (Java Message Service). El valor por omisión es el directorio actual. Es obligatorio si se utiliza JMS. Consulte el apartado "Utilización del almacenamiento de objetos JMS" en la página 106 para obtener más información acerca de la utilización de objetos JMS con MQSeries Adapter Kernel.

Elementos dependientes: Ninguno

Atributos: Ninguno

ePICApplications
Contiene información sobre las aplicaciones a las que atiende el kernel.

Elementos dependientes: `ePICApplication` (obligatorio)

Atributos: `o="ePICApplications"` (obligatorio)

ePICApplication
Especifica información acerca de una aplicación a la que atiende el kernel.

Elementos dependientes:

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

Atributos: epicappid="*ID_aplicación*", donde *ID_aplicación* es un identificador de aplicación válido (obligatorio)

epiclogging
Determina si se debe llevar a cabo la anotación cronológica de seguimiento. La anotación cronológica de seguimiento requiere el producto WebSphere Business Integrator. El valor por omisión es false.

Elementos dependientes: Ninguno

Atributos: Ninguno

epictrace
Determina si se debe utilizar el rastreo. El valor por omisión es false.

Elementos dependientes: Ninguno

Atributos: Ninguno

epictracelevel
Establece el nivel de rastreo, utilizando las constantes que especifica la clase com.ibm.logging.IRecordType. El valor por omisión es 0 (ningún mensaje). En la publicación *Problem Determination Guide* encontrará información detallada acerca del rastreo y una lista completa de los niveles de rastreo válidos.

Elementos dependientes: Ninguno

Atributos: Ninguno

epictraceclientid
Especifica el nombre de la aplicación cliente de rastreo. El valor por omisión es TraceClient.

Elementos dependientes: Ninguno

Atributos: Ninguno

epiclogoninfoclassname
Especifica el nombre de la clase de inicio de sesión que se utiliza para conectar a una aplicación cuando se utiliza un adaptador EAB. El valor por omisión es com.ibm.epic.adapters.eak.adapterdaemon.EpicLogonDefault.

Elementos dependientes: Ninguno

Atributos: Ninguno

AdapterRouting
Contiene información acerca de los tipos de mensaje y su direccionamiento.

Elementos dependientes:

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

Atributos: cn="epicadapterrouting" (obligatorio)

epicmqppqueuemgr
Si se utiliza MQSeries como mecanismo de transporte, especifica el nombre del gestor de colas que se va a utilizar. Si no se especifica, o si se especifica como DEFAULT, se utiliza el gestor de colas por omisión.

Elementos dependientes: Ninguno

Atributos: Ninguno

epicuseremotequeuemanagertosend
Si se utiliza MQSeries como mecanismo de transporte, especifica si se debe utilizar un gestor de colas remoto para enviar mensajes. El valor por omisión es false.

Elementos dependientes: Ninguno

Atributos: Ninguno

epicmqppqueuemgrhostname
Si se utiliza MQSeries como mecanismo de transporte, especifica el nombre de sistema principal TCP/IP de la máquina en la que reside el gestor de colas. Obligatorio sólo si se utiliza MQSeries Cliente.

Elementos dependientes: Ninguno

Atributos: Ninguno

epicmqppqueuemgrportnumber
Si se utiliza MQSeries como mecanismo de transporte, especifica el número de puerta del proceso de servidor del gestor de colas. El valor por omisión es 1414 (el valor por omisión de MQSeries). Obligatorio sólo si se utiliza MQSeries Cliente.

Elementos dependientes: Ninguno

Atributos: Ninguno

epicmqppqueuemgrchannelname
Si se utiliza MQSeries como mecanismo de transporte, especifica el nombre de canal del servidor del gestor de colas. Obligatorio sólo si se utiliza MQSeries Cliente.

Elementos dependientes: Ninguno

Atributos: Ninguno

epicjmsconnectionfactoryname
Si se utiliza JMS como mecanismo de transporte, especifica el nombre de fábrica de la Conexión JMS. El valor se debe especificar como *atributo=objeto*, donde *atributo* es el atributo LDAP y *objeto* es el objeto de Conexión JMS. El objeto se debe almacenar en el elemento `AdapterRouting`. Por ejemplo, para un objeto de conexión JMS denominado `QCFTEST1` con un atributo LDAP de `cn`, el valor que especifica este elemento es `cn=QCFTEST1`.

Elementos dependientes: Ninguno

Atributos: Ninguno

ePICBodyCategory
Especifica la categoría de cuerpo de los mensajes que se envían.

Elementos dependientes: `ePICBodyType` (obligatorio)

Atributo: `epicbodycategory=`*`categoría_cuerpo`*, donde *categoría_cuerpo* especifica la categoría de cuerpo de los mensajes que se envían (obligatorio)

ePICBodyType
Especifica el tipo de cuerpo de los mensajes que se envían.

Elementos dependientes:

- `epiccommandclassname`
- `epiccommandtype`
- `epiccommandejbmapper`
- `epiccommandejbmethod`
- `epiccommandejbmethodparmtype`
- `epiccommandejburl`
- `epiccommandejbinitialcontext`
- `epicdestids`
- `epicreceivemode`
- `epicmessageformatter`
- `epicreceivetimeout`
- `epicreceivemqppqueue`
- `epicerrormqppqueue`
- `epicreplymqppqueue`
- `epicjmsreceivequeuename`
- `epicjmserrorqueuename`

- epicjmsreplyqueuename
- epicreceivefiledir
- epiccommitfiledir
- epicerrorfiledir

Atributos: epicbodytype=*tipo_cuerpo*, donde *tipo_cuerpo* especifica el tipo de cuerpo de los mensajes que se envían (obligatorio)

epiccommandclassname
Especifica el nombre de un adaptador destino EAB o mandato EJB que se invoca para procesar mensajes. Es obligatorio si se utiliza un daemon del adaptador o Websphere Application Server para recibir mensajes.

Elementos dependientes: Ninguno

Atributos: Ninguno

epiccommandtype
Especifica el tipo de adaptador destino. Los valores posibles incluyen MQAKEAB y MQAKEJB. MQAKEAB especifica un adaptador destino EAB estándar de MQSeries Adapter Kernel; MQAKEJB especifica que los Enterprise Beans se utilizan en la parte destino del kernel de WebSphere Application Server. El valor por omisión es MQAKEAB. Es obligatorio un valor de MQAKEJB cuando el adaptador destino es un Enterprise Bean.

Elementos dependientes: Ninguno

Atributos: Ninguno

epiccommandejbmapper
Especifica el nombre de la clase TDCMapper que se utiliza para correlacionar datos de entrada. El valor por omisión es TDCGenericMapper. Es obligatorio cuando el adaptador destino es un Enterprise Bean.

Elementos dependientes: Ninguno

Atributos: Ninguno

epiccommandejbmethod
Especifica el nombre del método que se debe invocar en un Enterprise Bean. El método debe aceptar un objeto TerminalDataContainer como entrada y devolver un objeto TerminalDataContainer. El valor por omisión es execute. Es obligatorio cuando el adaptador destino es un Enterprise Bean.

Elementos dependientes: Ninguno

Atributos: Ninguno

epiccommandejbmethodparmtype
Especifica el nombre de clase del objeto que se utiliza como parámetro del método que se invoca en el Enterprise Bean. El valor por omisión es el nombre de clase del objeto devuelto por `TDCMapper`. Es obligatorio cuando el adaptador destino es un Enterprise Bean.

Elementos dependientes: Ninguno

Atributos: Ninguno

epiccommandejburl
Especifica el URL (uniform resource locator) de un Enterprise Bean difundido, en la forma `IIOP://`*`nombre_sistema principal`*`:`*`puerta`*, donde *nombre_sistema principal* es el nombre del sistema principal del servidor EJB y *puerta* es la puerta a la que el servidor de nombre escucha (por omisión, `900`). El valor por omisión es `IIOP:///`. Es obligatorio cuando el adaptador destino es un Enterprise Bean.

Elementos dependientes: Ninguno

Atributos: Ninguno

epiccommandejbinitialcontext
Especifica el nombre de la factoría de contexto inicial que se utiliza para buscar el nombre inicial del Enterprise Bean. El valor por omisión es `com.ibm.ejs.ns.jndi.CNInitialContextFactory`. Es obligatorio cuando el adaptador destino es un Enterprise Bean.

Elementos dependientes: Ninguno

Atributos: Ninguno

epicdestids
Especifica los identificadores de una o más aplicaciones que se van a utilizar como destinos de mensaje. Obligatorio si la aplicación envía mensajes y el ID lógico de destino se establece en `NONE`.

Elementos dependientes: Ninguno

Atributos: Ninguno

epicreceivemode
Especifica la modalidad de comunicación que se va a utilizar. En el "Apéndice A. Modalidades de comunicación" en la página 103 se enumeran y explican las modalidades de comunicación válidas. Obligatorio si la aplicación recibe mensajes.

Elementos dependientes: Ninguno

Atributos: Ninguno

epicmessageformatter
Especifica el formateador de mensajes que se va a utilizar,

dependiendo del valor de `epicreceivemode` y del método de transporte. En la Tabla 10 en la página 105 y en la Tabla 11 en la página 105 encontrará información detallada acerca de los formateadores de mensajes y los métodos de transporte.

Elementos dependientes: Ninguno

Atributos: Ninguno

epicreceivetimeout
Especifica el tiempo, en milisegundos, que espera el receptor los mensajes antes de exceder el tiempo de espera. El valor por omisión es `0`. Un valor de `-1` especifica que no hay tiempo de espera (espera indefinida).

Elementos dependientes: Ninguno

Atributos: Ninguno

epicreceivemqppqueue
Especifica el nombre de la cola de la que se van a recibir mensajes. Es obligatorio cuando el elemento `epicreceivemode` especifica una modalidad de comunicación de MQSeries. En el "Apéndice A. Modalidades de comunicación" en la página 103 se enumeran las modalidades de comunicación de MQSeries.

Elementos dependientes: Ninguno

Atributos: Ninguno

epicerrormqppqueue
Especifica el nombre de la cola en la que se van a colocar los mensajes de error. Es obligatorio si se utiliza la colocación de mensajes de error en cola y el elemento `epicreceivemode` especifica una modalidad de comunicación de MQSeries. En el "Apéndice A. Modalidades de comunicación" en la página 103 se enumeran las modalidades de comunicación de MQSeries.

Elementos dependientes: Ninguno

Atributos: Ninguno

epicreplymqppqueue
Especifica el nombre de la cola de la que se van a recibir los mensajes de respuesta. Es obligatorio si se utilizan peticiones de respuesta y el elemento `epicreceivemode` especifica una modalidad de comunicación de MQSeries. En el "Apéndice A. Modalidades de comunicación" en la página 103 se enumeran las modalidades de comunicación de MQSeries.

Elementos dependientes: Ninguno

Atributos: Ninguno

epicjmsreceivequeuename
Especifica el nombre de la cola de la que se van a recibir mensajes. Es obligatorio para la modalidad de comunicación JMS. El objeto se debe almacenar en el elemento ePICBodyType. El valor se debe especificar como *atributo=objeto*, donde *atributo* es el atributo LDAP y *objeto* es el nombre del objeto de cola de JMS. Por ejemplo, para un objeto JMS denominado TEST1AIQ con un atributo LDAP de cn, el valor que especifica este elemento es cn=TEST1AIQ.

Elementos dependientes: Ninguno

Atributos: Ninguno

epicjmserrorqueuename
Especifica el nombre de la cola en la que se van a colocar los mensajes de error. Es obligatorio si se utilizan las colas de mensajes de error con la modalidad de comunicación JMS. El objeto se debe almacenar en el elemento ePICBodyType. El valor se debe especificar como *atributo=objeto*, donde *atributo* es el atributo LDAP y *objeto* es el nombre del objeto de cola de JMS. Por ejemplo, para un objeto JMS denominado TEST1AEQ con un atributo LDAP de cn, el valor que especifica este elemento es cn=TEST1AEQ.

Elementos dependientes: Ninguno

Atributos: Ninguno

epicjmsreplyqueuename
Especifica el nombre de la cola de la que se van a recibir los mensajes de respuesta. Es obligatorio si se utilizan peticiones de respuesta con la modalidad de comunicación JMS. El objeto se debe almacenar en el elemento ePICBodyType. El valor se debe especificar como *atributo=objeto*, donde *atributo* es el atributo LDAP y *objeto* es el nombre del objeto de cola de JMS. Por ejemplo, para un objeto JMS denominado TEST1RPL con un atributo LDAP de cn, el valor que especifica este elemento es cn=TEST1RPL.

Elementos dependientes: Ninguno

Atributos: Ninguno

epicreceivefiledir
Especifica el nombre del directorio del que se van a recibir mensajes. Es obligatorio para la modalidad de comunicación FILE.

Elementos dependientes: Ninguno

Atributos: Ninguno

epiccommitfiledir
Especifica el nombre del directorio que va a contener los mensajes

recibidos hasta que se confirmen. Es obligatorio para la modalidad de comunicación FILE cuando se reciben mensajes.

Elementos dependientes: Ninguno

Atributos: Ninguno

epicerrorfiledir
Especifica el nombre del directorio en el que se van a colocar los mensajes de error. Es obligatorio si se utilizan las colas de mensajes de error con la modalidad de comunicación FILE.

Elementos dependientes: Ninguno

Atributos: Ninguno

ePICAdapterDaemonExtensions
Contiene información acerca de las extensiones de daemon del adaptador.

Elementos dependientes:

- `epicdepappid`
- `epicminworkers`

Atributos: `cn="epicappextensions"` (obligatorio)

ePICTraceExtensions
Contiene información acerca de las extensiones de rastreo. En la publicación *Problem Determination Guide* se proporciona más información acerca de este elemento y sus elementos dependientes.

Elementos dependientes:

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

Atributos: `cn="epicappextensions"` (obligatorio)

epicdepappid
Especifica el identificador de la aplicación a la que atiende el daemon del adaptador. Toma por omisión el ID de la aplicación con el que se ha iniciado el daemon del adaptador.

Elementos dependientes: Ninguno

Atributos: Ninguno

epicminworkers

Especifica el número de trabajos del adaptador que ha iniciado el daemon del adaptador. El valor por omisión es 1.

Elementos dependientes: Ninguno

Atributos: Ninguno

Configuraciones comunes

En este apartado se enumeran los valores de configuración de los diferentes entornos de configuración comunes, incluidos los valores para enviar y recibir mensajes utilizando varios transportes de comunicaciones. Cuando se envía un mensaje, los valores de configuración se obtienen de la parte origen y destino; cuando se recibe un mensaje, los valores de configuración se obtienen sólo de la parte destino. El origen y el destino están representados por sus respectivos identificadores lógicos. Estos ejemplos dan por sentado que el origen y el destino están en dos máquinas diferentes. Si el identificador de la aplicación de destino aún no está establecido, se determina desde el valor del elemento `epicdestids` en la configuración del origen.

Nota: Los entornos de configuración enumeran los valores de configuración de elementos que se aplican y se pueden establecer. Consulte la lista del elemento en el apartado "Elementos XML que se utilizan en el archivo de configuración" en la página 69 para obtener los valores por omisión que se aplican a dicho elemento.

Configuraciones comunes de MQSeries: En este apartado se ofrecen las configuraciones comunes cuando se utiliza MQSeries como transporte de comunicaciones. El elemento `epicreceivemode` especifica una modalidad de comunicación de MQSeries (por ejemplo, MQPP o MQRFH2). Se enumeran los siguientes casos:

- La Tabla 2 en la página 79 muestra los elementos de configuración que se deben establecer cuando se envía un mensaje desde un servidor MQSeries a otro servidor MQSeries.
- La Tabla 3 en la página 79 muestra los elementos de configuración que se deben establecer cuando se envía un mensaje desde un servidor MQSeries que utiliza un gestor de colas remoto a un servidor MQSeries.
- La Tabla 4 en la página 80 muestra los elementos de configuración que se deben establecer cuando se envía un mensaje desde un cliente MQSeries que utiliza un servidor principal a un servidor MQSeries.
- La Tabla 5 en la página 81 muestra los elementos de configuración que se deben establecer cuando se recibe un mensaje en un servidor MQSeries.
- La Tabla 6 en la página 82 muestra los elementos de configuración que se deben establecer cuando se recibe un mensaje en un cliente MQSeries que utiliza un servidor principal.

Tabla 2. Configuraciones comunes: enviar un mensaje desde un servidor MQSeries a otro servidor MQSeries

Configuración de origen	Configuración de destino
	El elemento `epicreceivemode` especifica una modalidad de comunicación de MQSeries.
El elemento `epicmqppqueuemgr` especifica el nombre del gestor de colas. Este gestor de colas debe salir en la máquina de la aplicación de origen.	
	El elemento `epicreceivemqppqueue` especifica el nombre de la cola de recepción. Esta cola debe ser una cola remota MQSeries de la máquina de la aplicación de destino o parte de un clúster MQSeries.
El elemento `epicreplymqppqueue` especifica el nombre de la cola de respuestas. Esta cola debe ser una cola local MQSeries de la máquina del emisor o parte de un clúster MQSeries. Sólo se utiliza para peticiones y respuestas síncronas.	
	El elemento `epicmessageformatter` especifica el nombre del formateador que se debe utilizar.
El elemento `epicreceivetimeout` especifica el tiempo que el receptor espera una respuesta antes del tiempo de espera.	

Tabla 3. Configuraciones comunes: enviar un mensaje desde un servidor MQSeries a otro servidor MQSeries a través de un gestor de colas remoto

Configuración de origen	Configuración de destino
	El elemento `epicreceivemode` especifica una modalidad de comunicación de MQSeries.
El elemento `epicmqppqueuemgr` especifica el nombre del gestor de colas. Este gestor de colas debe salir en la máquina de la aplicación de origen.	El elemento `epicmqppqueuemgr` especifica el nombre del gestor de colas. Este gestor de colas debe salir en la máquina de la aplicación de origen. Se debe especificar el nombre; no se puede utilizar un valor por omisión.

Tabla 3. Configuraciones comunes: enviar un mensaje desde un servidor MQSeries a otro servidor MQSeries a través de un gestor de colas remoto (continuación)

Configuración de origen	Configuración de destino
El elemento `epicremotequeuemanagertosend` especifica que se utiliza un gestor de colas remoto para enviar mensajes.	
	El elemento `epicreceivemqppqueue` especifica el nombre de la cola de recepción. Esta cola debe ser una cola local MQSeries de la máquina de la aplicación de destino o parte de un clúster MQSeries.
El elemento `epicreplymqppqueue` especifica el nombre de la cola de respuestas. Esta cola debe ser una cola local MQSeries de la máquina del emisor o parte de un clúster MQSeries. Sólo se utiliza para peticiones y respuestas síncronas.	
	El elemento `epicmessageformatter` especifica el nombre del formateador que se debe utilizar.
El elemento `epicreceivetimeout` especifica el tiempo que el receptor espera una respuesta antes de concluir.	

Tabla 4. Configuración común: enviar un mensaje desde un cliente MQSeries que utiliza un servidor principal a un servidor MQSeries

Configuración de origen	Configuración de destino
	El elemento `epicreceivemode` especifica una modalidad de comunicación de MQSeries.
El elemento `epicmqppqueuemgr` especifica el nombre del gestor de colas. Este gestor de colas debe salir en la máquina de sistema principal del cliente emisor.	
El elemento `epicmqppqueuemgrhostname` especifica el nombre del sistema principal de la máquina servidor MQSeries.	
El elemento `epicmqppqueuemgrportnumber` especifica el número de puerta del proceso de servidor del gestor de colas del servidor.	

Tabla 4. Configuración común: enviar un mensaje desde un cliente MQSeries que utiliza un servidor principal a un servidor MQSeries (continuación)

Configuración de origen	Configuración de destino
El elemento `epicmqppqueuemgrchannelnumber` especifica el número de canal del servidor del gestor de colas.	
	El elemento `epicreceivemqppqueue` especifica el nombre de la cola de recepción. Esta cola debe ser una cola remota MQSeries de la máquina de la aplicación de destino o parte de un clúster MQSeries.
El elemento `epicreplymqppqueue` especifica el nombre de la cola de respuestas. Esta cola debe ser una cola local MQSeries de la máquina de sistema principal del cliente emisor o parte de un clúster MQSeries. Sólo se utiliza para peticiones y respuestas síncronas.	
	El elemento `epicmessageformatter` especifica el nombre del formateador que se debe utilizar.
El elemento `epicreceivetimeout` especifica el tiempo que el receptor espera una respuesta antes de concluir.	

Tabla 5. Configuración común: el servidor MQSeries recibe un mensaje

Configuración de origen	Configuración de destino
No aplicable.	El elemento `epicreceivemode` especifica una modalidad de comunicación de MQSeries.
	El elemento `epicmqppqueuemgr` especifica el nombre del gestor de colas. Este gestor de colas debe salir en la máquina de la aplicación de origen.
	El elemento `epicreceivemqppqueue` especifica el nombre de la cola de recepción. Esta cola debe ser una cola local MQSeries de la máquina de destino.

Tabla 5. Configuración común: el servidor MQSeries recibe un mensaje (continuación)

Configuración de origen	Configuración de destino
	El elemento `epicerrormqppqueue` especifica el nombre de la cola de errores. Esta cola debe ser una cola local MQSeries de la máquina de destino o parte de un clúster. Obligatorio sólo si se utiliza un trabajo adaptador.
	El elemento `epicmessageformatter` especifica el nombre del formateador que se debe utilizar.
	El elemento `epicreceivetimeout` especifica el tiempo que el receptor espera una respuesta antes de concluir.

Tabla 6. Configuración común: cliente MQSeries que utiliza un servidor principal que recibe mensajes

Configuración de origen	Configuración de destino
No aplicable.	El elemento `epicreceivemode` especifica una modalidad de comunicación de MQSeries.
	El elemento `epicmqppqueuemgr` especifica el nombre del gestor de colas. Este gestor de colas debe salir en la máquina cliente de sistema principal del receptor.
	El elemento `epicmqppqueuemgrhostname` especifica el nombre del sistema principal de la máquina servidor MQSeries.
	El elemento `epicmqppqueuemgrportnumber` especifica el número de puerta del proceso de servidor del proceso de colas de la máquina servidor.
	El elemento `epicmqppqueuemgrchannelnumber` especifica el número de canal del servidor del gestor de colas.
	El elemento `epicreceivemqppqueue` especifica el nombre de la cola de recepción. Esta cola debe ser una cola local MQSeries de la máquina cliente de sistema principal del receptor.

Tabla 6. Configuración común: cliente MQSeries que utiliza un servidor principal que recibe mensajes (continuación)

Configuración de origen	Configuración de destino
	El elemento `epicerrormqppqueue` especifica el nombre de la cola de errores. Esta cola debe ser una cola local MQSeries de la máquina cliente de sistema principal o parte de un clúster. Obligatorio sólo si se utiliza un trabajo adaptador.
	El elemento `epicmessageformatter` especifica el nombre del formateador que se debe utilizar.
	El elemento `epicreceivetimeout` especifica el tiempo que el receptor espera una respuesta antes de concluir.

Configuraciones comunes de JMS: En este apartado se ofrecen las configuraciones comunes cuando se utiliza JMS como transporte de comunicaciones. El elemento `epicreceivemode` especifica JMS.

Si se utiliza la implementación MQSeries JMS, deben existir los objetos MQSeries adecuados. Por ejemplo, una factoría de conexión de colas JMS debe estar relacionada con un gestor de colas de un servidor MQSeries y una cola JMS debe estar relacionada con una cola MQSeries. No es necesario enumerar los objetos MQSeries en la configuración, pero los objetos MQSeries de soporte deben existir.

Se enumeran los siguientes casos:

- La Tabla 7 muestra los elementos de configuración que se deben establecer cuando se envía un mensaje a través de JMS.
- La Tabla 8 en la página 84 muestra los elementos de configuración que se deben establecer cuando se recibe un mensaje a través de JMS.

Tabla 7. Configuración común: enviar un mensaje a través de JMS

Configuración de origen	Configuración de destino
	El elemento `epicreceivemode` especifica la modalidad de comunicación de JMS.
El elemento `epicjmsconnectionfactoryname` especifica el nombre de la factoría de conexión de colas JMS. El objeto al que se hace referencia debe existir en la configuración.	

Tabla 7. Configuración común: enviar un mensaje a través de JMS (continuación)

Configuración de origen	Configuración de destino
	El elemento `epicjmsreceivequeuename` especifica el nombre de la cola de recepción JMS. El objeto al que se hace referencia debe existir en la configuración.
El elemento `epicjmsreplyqueuename` especifica el nombre de la cola de respuestas JMS. El objeto al que se hace referencia debe existir en la configuración. Sólo se utiliza para peticiones y respuestas síncronas.	
	El elemento `epicmessageformatter` especifica el nombre del formateador que se debe utilizar.
El elemento `epicreceivetimeout` especifica el tiempo que el receptor espera una respuesta antes de concluir.	

Tabla 8. Configuración común: recibir un mensaje a través de JMS

Configuración de origen	Configuración de destino
No aplicable.	El elemento `epicreceivemode` especifica la modalidad de comunicación de JMS.
	El elemento `epicjmsconnectionfactoryname` especifica el nombre de la factoría de conexión de colas JMS. El objeto al que se hace referencia debe existir en la configuración.
	El elemento `epicjmsreceivequeuename` especifica el nombre de la cola de recepción JMS. El objeto al que se hace referencia debe existir en la configuración.
	El elemento `epicjmserrorqueuename` especifica el nombre de la cola de errores JMS. El objeto al que se hace referencia debe existir en la configuración.
	El elemento `epicmessageformatter` especifica el nombre del formateador que se debe utilizar.
	El elemento `epicreceivetimeout` especifica el tiempo que el receptor espera una respuesta antes de concluir.

Configuraciones comunes del adaptador: En este apartado se ofrecen las configuraciones comunes cuando se invoca un adaptador de la parte destino. Se utilizan diferentes valores de configuración según si el destino es un adaptador destino EAB (Enterprise Access Builder), especificado por un valor `epiccommandtype` de `MQAKEAB`, o un adaptador destino de bean de sesión de servicio EJB, especificado por un valor `epiccommandtype` de `MQAKEJB`.

Nota: Los adaptadores destino de bean de sesión de servicio EJB sólo reciben soporte de WebSphere Application Server.

Para un valor `epiccommandtype` de `MQAKEAB`, especifique los valores de los siguientes elementos adicionales:

- `epiclogoninfoclassname`
- `epiccommandclassname`

Para un valor `epiccommandtype` de `MQAKEJB`, especifique los valores de los siguientes elementos adicionales:

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

Adición de información del adaptador a la configuración

Cuando se añade un nuevo adaptador a la configuración del kernel, se deben añadir varias especificaciones al archivo de configuración para la configuración mínima. Para ver un ejemplo de un archivo de configuración mínima, consulte el archivo `aqmconfig.minimum.xml`. Este archivo se muestra en el "Apéndice D. Ejemplo del archivo de configuración" en la página 119 y también se incluye en el directorio `samples` de la instalación de MQSeries Adapter Kernel.

Las especificaciones siguientes representan la información mínima que se debe añadir a la configuración cuando se añade un nuevo adaptador:

- **Adaptador origen** (envío de mensajes):
  - El identificador lógico de la aplicación en la que se ejecuta el adaptador origen.
  - El gestor de colas por omisión. Si se utiliza MQSeries como mecanismo de transporte, y se ha instalado y está en ejecución en la misma máquina que el adaptador origen, no es necesario configurar el gestor de colas específicamente.

- Identificadores lógicos de destino para los mensajes. Si todos los mensajes van al mismo destino, se debe utilizar una categoría de cuerpo DEFAULT y un tipo de cuerpo DEFAULT.
- Una cola de recepción para cada identificador lógico de destinación al que el adaptador origen envía mensajes.

- **Adaptador destino** (recepción de mensajes):
  - El identificador de la aplicación en la que se ejecuta el adaptador destino.
  - El gestor de colas por omisión. Si se utiliza MQSeries como mecanismo de transporte, y se ha instalado y está en ejecución en la misma máquina que el adaptador origen, no es necesario configurar el gestor de colas específicamente.
  - La modalidad de recepción para MQSeries. Por lo general, es la misma para todos los mensajes. Si es así, utilice una categoría de cuerpo DEFAULT y un tipo de cuerpo DEFAULT.
  - La cola de recepción. Si es la misma para todos los mensajes, utilice una categoría de cuerpo DEFAULT y un tipo de cuerpo DEFAULT.
  - La cola de errores, en caso de que se produzca algún error cuando el adaptador destino procesa el mensaje. Por lo general, es la misma para todos los mensajes. Si es así, utilice una categoría de cuerpo DEFAULT y un tipo de cuerpo DEFAULT.
  - El nombre de la clase de adaptador destino al que invocar cuando se recibe un mensaje. Es específico de la categoría y el tipo de cuerpo.
  - Valor de tiempo de espera de recepción. Se recomienda establecer un valor apropiado para evitar usar mucho la CPU. Por lo general, es el mismo para todos los mensajes. Si es así, utilice una categoría de cuerpo DEFAULT y un tipo de cuerpo DEFAULT.

  Para adaptadores destino adicionales, la misma información puede ser suficiente si se utiliza la misma cola de recepción. En este caso, la única información distinta que se debe especificar es el nombre de la clase de adaptador destino para invocar la categoría y el tipo de cuerpo específicos.
- **Especificaciones de rastreo**:
  - Si el rastreo está activado o desactivado.
  - El nivel de rastreo.
  - Otras especificaciones de rastreo, que incluyen el destino del rastreo, para adaptadores origen y destino. Por omisión, el rastreo se muestra en el terminal o en la ventana del indicador de mandatos en que se ha iniciado el kernel.

Edición del archivo de configuración

Utilice un editor de texto o un editor XML dedicado para editar el archivo de configuración. En el directorio `samples` de la instalación del kernel se

proporciona un archivo DTD denominado `aqmconfig.dtd` para los usuarios de editores XML. Se puede bajar un editor XML denominado Xeena del sitio web IBM alphaWorks, situado en la dirección siguiente: www.alphaworks.ibm.com. Para editar el archivo de configuración, se aplican las recomendaciones siguientes:

- Antes de editar el archivo de configuración, reúna toda la información pertinente acerca de la configuración que desea, lo que incluye los nombres de las aplicaciones y las colas implicadas en la configuración, los tipos de mensajes que se van a intercambiar, la modalidad o las modalidades de comunicación que se van a utilizar e información acerca de los programas de rastreo y otras extensiones.
- Copie el archivo `aqmconfig.xml` de ejemplo del directorio `samples` en la ubicación que desee. No redenomine la copia del archivo de configuración. Edite la copia.
- Utilice elementos para identificar secciones diferentes del archivo de configuración y para documentar los valores específicos que se utilizan en la configuración (por ejemplo, identificadores de aplicación, nombres de colas de mensajes y valores de tiempo de espera). En XML, los comentarios empiezan por los caracteres `<!--` y finalizan con los caracteres `-->`. Los comentarios pueden ocupar varias líneas, como en el ejemplo siguiente:

  ```
  <!--
       Texto de comentario
  -->
  ```

  Debe tener en cuenta que XML no permite comentarios dentro de otros comentarios.
- Organice el archivo de configuración según los identificadores de la aplicación. Mantenga unidas las entradas de cada identificador de aplicación.
- Si no utiliza un editor XML dedicado, utilice un editor de texto que mantenga los finales de línea y no las divida al guardar el archivo. Algunos ejemplos de este tipo de editor de texto son el Bloc de notas de los sistemas Windows y vi o Emacs de UNIX.
- Recuerde que XML es sensible a las mayúsculas y minúsculas, por lo que debe utilizar las mayúsculas y minúsculas de modo correcto para todos los atributos y los nombres de códigos (elementos). Si usa de modo incorrecto las mayúsculas y minúsculas al codificar, puede invalidar el archivo de configuración. Utilizar un editor XML dedicado puede ayudar a evitar errores de mayúsculas y minúsculas.
- Si desea utilizar valores por omisión para la categoría y el tipo de cuerpo, pero aún no ha establecido el valor por omisión, debe configurar el valor DEFAULT para cada valor en el archivo de configuración. En caso contrario, el kernel no utiliza ningún valor por omisión.

- Valide el archivo de configuración antes de utilizarlo para el trabajo real. Consulte el apartado “Validación del archivo de configuración”.
- Los cambios que realiza en el archivo de configuración se activan la próxima vez que se inicia un proceso kernel. Si hay un proceso en ejecución al cambiar el archivo de configuración, se debe detener y reiniciar el proceso para que se activen los cambios. Preste especial atención si edita el archivo de configuración que se está utilizando actualmente para el trabajo real.
- Haga una copia de seguridad del archivo de configuración cada vez que lo edite.

Validación del archivo de configuración

Después de editar el archivo de configuración y antes de utilizarlo para el trabajo real, se aconseja validarlo. Para validar el archivo de configuración, lleve a cabo los pasos generales siguientes:

1. Cree un directorio de validación del archivo de configuración en el que validar y configurar la prueba.
2. Cree un mensaje XML de validación.
3. Configure colas de mensajes para ofrecer soporte para la prueba de validación.
4. Configure y, a continuación, ejecute una prueba de validación del archivo de configuración en la que se envíe y reciba un mensaje.
5. Examine los resultados de la prueba para determinar si el archivo de configuración es correcto.

Tanto el programa de utilidad que ayuda a crear un mensaje XML de validación como la prueba de validación del archivo de configuración se proporcionan como parte del kernel.

La prueba de validación del archivo de configuración invoca al método `sendMsg` y envía un mensaje XML de validación desde un adaptador nativo de la parte origen del kernel a un daemon del adaptador de la parte destino del kernel. No se necesita un adaptador origen y un adaptador destino. Sin embargo, si se dispone de un adaptador destino, también se puede probar el envío del mensaje a la aplicación de destino.

El procedimiento es el siguiente:

Nota: para mayor comodidad, se proporcionan varios scripts que se pueden utilizar en el procedimiento. Si lo desea, puede copiar los scripts y, a continuación, editar las copias para crear sus versiones personalizadas. Si utiliza OS/400, las versiones UNIX de los scripts se pueden ejecutar

en una sesión **qsh**. Puede iniciar una sesión **qsh** entrando el mandato de inicio de QSH (**STRQSH**) en un indicador de CL (Control Language).

Paso 1. Abra una ventana de indicador de mandatos.

Paso 2. Cree un directorio de validación del archivo de configuración y copie en él el archivo de configuración y el archivo de instalación.

Paso 3. Cambie al directorio de validación.

Paso 4. Entre el mandato siguiente para crear el mensaje XML de validación:

- `aqmcrtmsg.bat` (sistemas Windows)
- `aqmcrtmsg.sh` (UNIX y OS/400)

Paso 5. Se muestra una lista de opciones. Seleccione una opción y pulse Intro. Entre un valor para cada uno. Puede entrar los valores en el orden que desee. Algunos ejemplos de opciones son: `set sourcelogicalid`, `set msgtype` y `set bodycategory`. Debe entrar valores para las opciones 20, 21, 22 y 23. Puede utilizar las opciones 24 ó 241 para proporcionar datos del cuerpo del mensaje. Los demás valores no son obligatorios.

Paso 6. Entre la opción 1 para crear el archivo XML de validación. El archivo XML de validación se crea en el directorio actual y se denomina `EpicMessage`*nn*`.xml`, donde *nn* es el número del archivo XML.

Paso 7. Entre la opción 0 para salir del programa de utilidad de validación.

Paso 8. Configure las colas de mensajes adecuadas para ofrecer soporte para la validación.

Paso 9. Establezca la variable de entorno AQMSETUPFILE para que señale temporalmente al archivo de instalación del directorio de validación.

- En los sistemas Windows, entre lo siguiente en un indicador de mandatos:

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

  donde `E:\` representa la unidad correcta y *archivos_ejecución* es el directorio de validación.

- En UNIX y OS/400, entre el mandato siguiente: En el ejemplo de mandato se da por supuesto que se utiliza el shell Korn. Si utiliza otro shell, cambie el mandato según proceda.

  ```
  export AQMSETUPFILE=directorio_raíz/archivos_ejecución/aqmsetup
  ```

  donde *directorio_raíz* es el directorio de instalación del kernel y *archivos_ejecución* es el directorio de validación. En OS/400, el

archivo aqmsetup siempre debe estar ubicado en el directorio home de IFS (/home/*nombre_usuario*).

Si fuera necesario, edite el archivo de instalación del directorio de validación para que señale al archivo de configuración que se está validando.

Paso 10. Elija lo que desea probar entre lo siguiente:

- Sólo la parte origen del kernel.
- Si se puede direccionar el mensaje hasta la aplicación de destino. Para esta prueba debe haber un adaptador destino instalado previamente.
- Rastreo.

Pruebe primero la parte origen y después la parte destino. Desactive el daemon del adaptador para probar sólo la parte origen. Active el daemon del adaptador para probar también la parte destino. Si el adaptador destino aún no está instalado, puede probar si el daemon del adaptador procesa el mensaje hasta el punto en que intenta invocar al mandato para el adaptador destino adecuado. Se aconseja habilitar el rastreo, especialmente si aún no se ha instalado un adaptador destino.

Paso 11. Ejecute la prueba de validación. Desde cualquier directorio, entre el mandato siguiente:

- En sistemas Windows:

```
aqmsndmsg.bat -a
identificador_lógico_origen -f archivo_mensajes_XML
```

- En UNIX y OS/400:

```
aqmsndmsg.sh -a
identificador_lógico_origen -f archivo_mensajes_XML
```

donde:

identificador_lógico_origen
indica el identificador lógico de origen. Este valor debe coincidir con el valor del identificador lógico de origen que se ha entrado para la opción 20 en el Paso 5 en la página 89.

archivo_mensajes_XML
indica el archivo de mensajes XML.

Nota: al entrar el mandato siguiente se puede visualizar una lista de todas las opciones disponibles para esta prueba:

En sistemas Windows:

```
aqmsndmsg.bat -?
```

En UNIX y OS/400:

```
aqmsndmsg.sh -?
```

Se debe tener en cuenta que el símbolo `-?` sólo funciona en el shell Korn. Si utiliza otro shell de UNIX (como, por ejemplo, el shell Bourne o C), utilice el carácter de la barra inclinada invertida antes del símbolo de interrogación (es decir, `-\?`).

Paso 12. Examine los resultados. El mensaje de validación contiene los datos y la categoría y el tipo de cuerpo correctos.

- Si sólo prueba la parte origen del kernel (es decir, si no ha iniciado el daemon del adaptador), examine la cola a la que se ha direccionado el mensaje.
  - Si ve el mensaje de validación en la cola, significa que las entradas del archivo de configuración se han validado.
  - Si no ve el mensaje de validación en la cola, compruebe el archivo de excepciones. Si ha habilitado el rastreo, compruebe los mensajes de rastreo.
- Si prueba la parte destino del kernel y dispone de un adaptador destino, compruebe la aplicación de destino.
  - Si la aplicación de destino ha recibido el mensaje de validación, significa que las entradas del archivo de configuración se han validado.
  - Si la aplicación de destino no ha recibido el mensaje de validación, compruebe el archivo de excepciones. Si ha habilitado el rastreo, compruebe los mensajes de rastreo.
- Si prueba la parte destino del kernel y no dispone de un adaptador destino, compruebe si el mensaje de validación está en la cola de errores y el mensaje de excepción en el archivo de excepciones. Si ha habilitado el rastreo, compruebe los mensajes de rastreo.
  - Si ve el mensaje de validación en la cola de errores y un mensaje de excepción, significa que las entradas del archivo de configuración se han validado.
  - Si no ve el mensaje de validación en la cola de errores, compruebe el archivo de excepciones. Si ha habilitado el rastreo, compruebe los mensajes de rastreo.

Paso 13. Si fuera necesario, modifique el archivo de configuración y vuélvalo a validar.

Configuración de MQSeries y MQSeries Integrator

Configure MQSeries y el software opcional como, por ejemplo, MQSeries Integrator, para ofrecer soporte para el kernel, tal como se indica a continuación:

En MQSeries:

- Para verificar la instalación, se utilizan varias colas. Si las utiliza para sus entornos de trabajo real o de prueba, bórrelas para verificar la instalación. Consulte en el apartado "Procedimiento de verificación" en la página 48 las colas que se utilizan para verificar la instalación.
- Configure colas para ofrecer soporte para el transporte de mensajes según el esquema de direccionamiento que ha diseñado.
- Cuando cree colas, establezca la variable de entorno MAX_QUEUE_DEPTH para la profundidad máxima de cola permitida.

En MQSeries Integrator, configure colas de entrada y de salida en normas (versión 1.1) o en flujos de mensajes (versión 2) que correspondan a las colas que se han configurado en el archivo de configuración.

Recomendaciones de rendimiento

Las recomendaciones de rendimiento siguientes se aplican específicamente a MQSeries Adapter Kernel:

- Al analizar DTD de XML, asegúrese de que los archivos DTD residen en el mismo directorio que el proceso que los analiza. De este modo, se reduce el esfuerzo que debe realizar el proceso para localizar los DTD.
- Cuando se envían o reciben mensajes de gran tamaño, utilizar el tipo de mensaje RFH2 ofrece mejor rendimiento que utilizar el tipo de mensaje XML.

En la documentación de MQSeries se proporcionan recomendaciones generales para mejorar el rendimiento.

Inicio del kernel

Para iniciar el kernel, inicie los elementos siguientes:

- El daemon del adaptador para cada aplicación de destino
- El servidor de rastreo (opcional)

Debe tener en cuenta que si ejecuta el adaptador origen en el proceso de la aplicación de origen, el adaptador origen se inicia automáticamente con la aplicación de origen, es decir, no es necesario realizar pasos adicionales para iniciar el adaptador origen y, por este motivo, se deben iniciar todos los

daemons o servidores que contengan adaptadores origen. Los adaptadores origen no se deben iniciar directamente.

Inicie cada daemon del adaptador y servidor de rastreo llevando a cabo los pasos siguientes:

Nota: para mayor comodidad, se proporcionan varios scripts que se pueden utilizar en el procedimiento. Si lo desea, puede copiar los scripts y, a continuación, editar las copias para crear sus versiones personalizadas. Si utiliza OS/400, las versiones UNIX de los scripts se pueden ejecutar en una sesión **qsh**. Puede iniciar una sesión **qsh** entrando el mandato de inicio de QSH (**STRQSH**) en un indicador de CL (Control Language).

Paso 1. Inicie MQSeries u otro software de mensajería y el software opcional como, por ejemplo, MQSeries Integrator.

Paso 2. Inicie el software asociado que necesita para su sitio, por ejemplo, aplicaciones (fuera del kernel) para leer los mensajes de rastreo de las colas.

Paso 3. Abra un indicador de mandatos. Para cada daemon del adaptador, entre el mandato siguiente:

- En sistemas Windows:

  ```
  aqmstrad.bat -a identificador_aplicación [-bc categoría_cuerpo
  -bt tipo_cuerpo] [-noretry]
  ```

- En UNIX y OS/400:

  ```
  aqmstrad.sh -a identificador_aplicación [-bc categoría_cuerpo
  -bt tipo_cuerpo] [-noretry]
  ```

  donde

  -a *identificador_aplicación*
  Identifica el identificador lógico de destinación al que atiende el daemon del adaptador.

  -bc *categoría_cuerpo*
  Especifica la categoría de cuerpo que utiliza el trabajo del daemon del adaptador para determinar la modalidad de comunicación y la información relacionada para recibir mensajes. Si no se indica ningún valor, el daemon del adaptador utiliza el valor DEFAULT.

  -bt *tipo_cuerpo*
  Especifica el tipo de cuerpo que utiliza el trabajo del daemon del adaptador para determinar la modalidad de comunicación y la información relacionada para recibir mensajes. Si no se indica ningún valor, el daemon del adaptador utiliza el valor DEFAULT.

-noretry
Especifica que el trabajo se detiene automáticamente cuando no hay más mensajes. Si no se especifica `-noretry`, el trabajo sondea continuamente si hay mensajes en la cola y el daemon del adaptador se debe detener de modo manual.

Nota: Si necesita modificar los parámetros de arranque de Java, edite el archivo `aqmstrad.bat` (sistemas Windows) o `aqmstrad.sh` (UNIX y OS/400). Consulte los comentarios que se incluyen en el archivo para obtener más detalles.

Paso 4. Para cada servidor de rastreo, entre el mandato siguiente:

- En sistemas Windows:

  `aqmstrtd.bat -`*`cómo`* `-a` *`identificador_aplicación_origen`*

- En UNIX y OS/400:

  `aqmstrtd.sh -`*`cómo`* `-a` *`identificador_aplicación_origen`*

  donde:

  -*cómo*
  Indica el modo de rastrear los mensajes que se reciben. Los valores posibles incluyen los siguientes:
  - `socket`
  - `ena`, es decir, adaptador nativo

  -a *identificador_aplicación_origen*
  Identificador de la aplicación de origen. Si no se indica ningún valor, se utiliza el valor por omisión, `TraceServer`, del archivo de configuración.

Si desea ver más información acerca de los servidores de rastreo, consulte la publicación *Problem Determination Guide*.

Paso 5. Después de iniciar un daemon del adaptador o un servidor de rastreo, una ventana de proceso permanece abierta hasta que se detiene el daemon del adaptador. La ventana del proceso puede mostrar excepciones. Consulte el apartado "Mensajes de excepción" en la página 96.

Detención del kernel

Para detener el kernel, detenga cada uno de los daemons del adaptador y los servidores de rastreo. Existen numerosos procedimientos para detenerlos:

- Cuando inicie el daemon del adaptador, establezca el parámetro `-noretry`. Consulte el apartado "Inicio del kernel" en la página 92.
- Vaya al indicador de mandatos (sistemas Windows) o al terminal (UNIX) desde el que ha iniciado el daemon del adaptador o el servidor de rastreo y pulse **Ctrl-C**. Realice este paso para cada daemon del adaptador o servidor de rastreo.

- En los sistemas Windows, puede utilizar el Administrador de tareas para finalizar el proceso.
- En UNIX, puede utilizar el mandato **ps** para determinar el número del proceso y, a continuación, utilizar el mandato **kill** para finalizarlo.

Mantenimiento del kernel

Establezca un plan de mantenimiento del kernel. Se recomienda hacer una copia de seguridad de los elementos siguientes periódicamente:

- La configuración, tal como se ha especificado en los archivos siguientes:
  - `aqmconfig.xml`
  - `aqmsetup`
- Los adaptadores que se han generado y sus archivos asociados

No es necesario hacer una copia de seguridad o suprimir periódicamente el contenido del archivo de rastreo y de otros archivos que utiliza el kernel para ofrecer soporte para su propio proceso. Si lo desea, puede hacer una copia de seguridad de estos archivos. Si los mensajes de rastreo se direccionan a un único archivo, en lugar de a varios archivos, el archivo de rastreo único puede llegar a ser muy grande. Si establece el nivel de rastreo para obtener un gran número de detalles (por ejemplo, todos los mensajes de rastreo o los mensajes informativos), tenga en cuenta que debe suprimir los archivos de rastreo periódicamente.

Diagnóstico de problemas

Puede utilizar mensajes de excepción, mensajes de rastreo y la cola de errores de MQSeries como ayuda para el diagnóstico de problemas. MQSeries Adapter Kernel produce mensajes de excepción y, si se ha habilitado el rastreo, mensajes de rastreo. Para obtener más información acerca del diagnóstico de problemas en un entorno MQSeries Adapter Kernel, consulte la publicación *Problem Determination Guide*.

Para comprender los mensajes de excepción y de rastreo, debe comprender el funcionamiento del kernel. El kernel utiliza una cola de errores para manejar determinados errores. Consulte el apartado “Cómo funciona el kernel” en la página 9.

La combinación del identificador de mensaje exclusivo y el identificador de transacción exclusivo le permite identificar los mensajes que han causado los mensajes de excepción y de rastreo.

No hay ningún identificador que le permita identificar exactamente el mismo mensaje en la cola de errores y en el kernel. Sin embargo, puede correlacionar

de modo manual un mensaje de la cola de errores con el mensaje de rastreo o de excepción correspondiente, o con ambos. Puede comparar uno o más de los siguientes:

- Indicación aproximada de la hora
- Cola del identificador lógico de origen
- Cola del identificador lógico de destinación
- Categoría de cuerpo
- Tipo de cuerpo
- Identificador de mensaje exclusivo
- Identificador de transacción exclusivo

Si coinciden, es probable que haya correlacionado el mensaje de la cola de errores con el mensaje de rastreo o de excepción correspondiente.

Número de versión

Ejecute el archivo `aqmversion.bat` (sistemas Windows) o `aqmversion.sh` (UNIX y OS/400) del directorio `bin` para visualizar el número de versión del kernel.

Mensajes de excepción

El kernel crea los tipos de mensajes de excepción siguientes:

- El adaptador nativo de la parte origen del kernel emite excepciones al adaptador origen. Consulte la documentación de MQSeries Adapter Builder para obtener información acerca del modo en que el adaptador origen maneja estas excepciones.
- El adaptador nativo de la parte destino del kernel emite excepciones al trabajo que gestiona el adaptador nativo.
- El trabajo graba excepciones en el archivo `EpicSystemExceptionFile`*`nnnnnnnn`*`.log`, que reside en el mismo directorio que el trabajo.
- El daemon del adaptador graba mensajes de excepción en un archivo de excepciones denominado `EpicSystemExceptionFile`*`nnnnnnnn`*`.log`, que reside en el mismo directorio que el daemon del adaptador. Puesto que el daemon del adaptador y sus trabajos residen en el mismo directorio, todos graban en el mismo archivo de excepciones. El daemon del adaptador también graba mensajes de excepción en la consola (es decir, el indicador de mandatos o el terminal que se ha utilizado para iniciarlo, si se ha iniciado desde una ventana).

Los mensajes de excepción de rastreo del kernel son diferentes de los mensajes de excepción de MQSeries. A continuación, se incluye un ejemplo de un mensaje de excepción del kernel:

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

Los valores del mensaje de excepción dependen de la naturaleza del mensaje y pueden incluir los elementos siguientes:

- Indicación de la hora
- Identificador lógico de origen
- Identificador lógico de destinación
- Categoría de cuerpo
- Tipo de cuerpo
- Identificador de mensaje exclusivo
- Identificador de transacción exclusivo
- Información de excepción

Consulte el apartado “Problemas habituales de la verificación” en la página 49, donde se explican los problemas comunes que se pueden producir durante la verificación de la instalación y se proporcionan las posibles respuestas.

Mensajes de rastreo

El kernel se puede configurar para crear mensajes de rastreo. Para obtener más información acerca del rastreo, consulte la publicación *Problem Determination Guide*.

Programas de utilidad

Creación de colas de MQSeries

Para automatizar la creación de colas de MQSeries, puede utilizar archivos de proceso por lotes o scripts de shell. Ejecute el archivo `aqmcreateq.bat` (sistemas Windows) o `aqmcreateq.sh` (UNIX y OS/400), utilizando el nombre de la aplicación como parámetro. Estos archivos crean las colas siguientes para cada aplicación:

- Cola de recepción, denominada *nombre_aplicación*AIQ.
- Cola de errores, denominada *nombre_aplicación*AEQ.
- Cola de respuestas, denominada *nombre_aplicación*RPL.

Capítulo 5. Utilización de API de MQSeries Adapter Kernel

El kernel incluye API que se utilizan para funciones como, por ejemplo, enviar y recibir mensajes, crear y analizar XML y gestionar la configuración del kernel. Las utilizan los adaptadores que se crean utilizando MQSeries Adapter Builder. El Centro de información de MQSeries Adapter Kernel incluye documentación de API en línea asociada, en formato HTML Javadoc.

La finalidad del kernel es utilizarlo con los adaptadores que genera el usuario usando MQSeries Adapter Builder. El kernel no tiene como finalidad que lo utilicen únicamente las invocaciones a las API del kernel desde el código personalizado. La documentación de API en línea sólo se proporciona como ayuda para comprender el funcionamiento del kernel y para los diagnósticos.

La documentación de API en línea del kernel está situada en el directorio `documentation`.

© Copyright IBM Corp. 2000, 2001

Capítulo 6. Obtención de información adicional

Hay varias fuentes de información que pueden resultar útiles cuando se utiliza la Oferta MQSeries Adapter. Para obtener información adicional acerca de MQSeries Adapter Kernel, consulte el documento *Problem Determination Guide*, disponible en el Centro de información de MQSeries Adapter Kernel que se instala con el producto. La publicación *Problem Determination Guide* proporciona información para resolver problemas específicos que pueden surgir al utilizar el kernel. Para obtener información acerca de MQSeries Adapter Builder, consulte el Centro de información del producto y el sistema de ayuda en línea.

Disponible en Internet

El sitio web de la familia de productos MQSeries se encuentra en la dirección siguiente: www.ibm.com/software/ts/mqseries/. Siguiendo los enlaces de este sitio web, puede:

- Obtener la información más actualizada acerca de la familia de productos MQSeries, que incluye la Oferta MQSeries Adapter.
- Acceder a las publicaciones de MQSeries en formatos HTML y PDF, que pueden incluir una edición más reciente de esta publicación. El enlace directo a la página de la biblioteca MQSeries es el siguiente: www.ibm.com/software/ts/mqseries/library/manualsa/.
- Bajar SupportPacs de MQSeries.

Para obtener información acerca de la utilización de MQSeries en OS/400, consulte la biblioteca de OS/400, en la dirección siguiente: www.ibm.com/servers/eserver/iseries/library/. También puede consultar las publicaciones específicas de OS/400, disponibles en el sitio web de la biblioteca de MQSeries, en la dirección siguiente: www.ibm.com/software/ts/mqseries/library/manualsa/.

Consultas

En el material de consulta siguiente se tratan temas contenidos en este documento:

- El sitio web de Open Applications Group está en la dirección siguiente: www.openapplications.org/
- La recomendación acerca de XML (Extensible Markup Language) 1.0 W3C está en la dirección siguiente: www.w3.org/TR/1998/Rec-xml-19980210

© Copyright IBM Corp. 2000, 2001

No son sitios web de IBM.

Apéndice A. Modalidades de comunicación

En este apéndice se proporciona información acerca de las modalidades de comunicación para las que ofrece soporte MQSeries Adapter Kernel y las clases Java que se utilizan para dicho soporte. Algunas modalidades de comunicación se facilitan como modalidades de ayuda con formateadores por omisión. En la Tabla 10 en la página 105 se indican los formateadores por omisión que se utilizan con las modalidades de ayuda.

Se ofrece soporte para las modalidades de comunicación siguientes:

MQPP El kernel transporta mensajes utilizando servicios base de MQSeries. Es una modalidad de ayuda.

MQRFH1 El kernel transporta mensajes utilizando MQSeries e intermedia en los mensajes utilizando MQSeries Integrator, versión 1.1. Es una modalidad de ayuda.

MQRFH2 El kernel transporta mensajes utilizando MQSeries e intermedia en los mensajes utilizando MQSeries Integrator, versión 2. Es una modalidad de ayuda.

MQBD El kernel transporta mensajes utilizando servicios base de MQSeries, pero sólo envía y recibe datos del cuerpo. Es una modalidad de ayuda. Esta modalidad tiene las características exclusivas siguientes:

- Sólo puede enviar datos del cuerpo, no valores de cabecera de mensaje.
- Puede recibir mensajes que sólo contengan datos del cuerpo. Utiliza los valores de cabecera de mensaje por omisión siguientes para los mensajes recibidos:
  - `SourceLogicalApplicationID`—Valor del objeto `ENAService` que se utiliza en la llamada del método de recepción.
  - `BodyCategory`—Valor del objeto `ENAService` que se utiliza en la llamada del método de recepción.
  - `BodyType`—Valor del objeto `ENAService` que se utiliza en la invocación del método de recepción.
  - `Acknowledgment`—Si el MQMessage recibido es una petición (REQUEST) de MQSeries, `Acknowledgment` (acuse de recibo) se establece en 1.
  - `BodyData`—Datos de mensaje recibidos de MQSeries.

© Copyright IBM Corp. 2000, 2001

Todos los demás valores de cabecera utilizan los valores por omisión normales.

MQ El kernel transporta mensajes utilizando servicios base de MQSeries.

JMS El kernel transporta mensajes utilizando JMS (Java Message Service). Para obtener más información acerca de la utilización de objetos JMS con MQSeries Adapter Kernel, consulte el apartado "Utilización del almacenamiento de objetos JMS" en la página 106.

FILE El kernel coloca y obtiene los mensajes de un archivo. Esta modalidad sólo se proporciona para diagnósticos.

En la Tabla 9 se enumeran las modalidades de comunicación y las clases Java que les dan soporte. Todas las clases Java pertenecen al paquete Java `com.ibm.epic.adapters.eak.nativeadapter`. Cualquier clase Java que ofrezca soporte para LMS (servicio lógico de mensajes) se puede especificar como modalidad de comunicación. En este caso, se utiliza la clase en sí misma para ofrecer soporte para la comunicación.

Tabla 9. Modalidades de comunicación y clases Java de soporte

Modalidad de comunicación	**Clase Java**	**Notas**
MQPP	`LMSMQBindingMQPP`	Requiere la instalación de MQSeries
MQRFH1	`LMSMQBindingMQRFH1`	Requiere la instalación de MQSeries
MQRFH2	`LMSMQBindingMQRFH2`	Requiere la instalación de MQSeries
MQBD	`LMSMQMQBD`	Requiere la instalación de MQSeries
MQ	`LMSMQBinding`	Requiere la instalación de MQSeries
JMS	`LMSJMS`	Requiere la instalación de JMS
FILE	`LMSFile`	Ninguna

En la Tabla 10 en la página 105 se enumeran las modalidades de comunicación y sus interfaces de formateador asociadas. En la Tabla 11 en la página 105 se establecen referencias cruzadas de las interfaces de formateador, los nombres de clase de formateador y sus utilizaciones. Todos los formateadores pertenecen al paquete Java `com.ibm.epic.adapters.eak.nativeadapter`. Se puede especificar cualquier clase de formateador para la modalidad de

comunicación. En este caso, se utiliza como formateador la clase de formateador especificada.

Tabla 10. Modalidades de comunicación e interfaces de formateador

Modalidad de comunicación	Interfaz de formateador	Formateador por omisión
MQPP	MQFormatterInterface	MQNMXMLFormatter
MQRFH1	MQFormatterInterface	MQNMRFH1Formatter
MQRFH2	MQFormatterInterface	MQNMRFH2Formatter
MQBD	MQFormatterInterface	MQNMBDFormatter
MQ	MQFormatterInterface	MQNMXMLFormatter
JMS	JMSFormatterInterface	JMSNMRFH2Formatter
FILE	StringFormatterInterface	NMXMLFormatter

Tabla 11. Interfaces de formateador, nombres de clase de formateador y finalidades

Interfaz de formateador	Nombre de clase de formateador	Finalidad
MQFormatterInterface	MQNMXMLFormatter	EpicMessage como XML
	MQNMRFH1Formatter	EpicMessage como RFH1
	MQNMRFH2Formatter	EpicMessage como RFH2
	MQNMDBFormatter	Sólo datos del cuerpo
JMSFormatterInterface	JMSNMXMLFormatter	EpicMessage como XML
	JMSNMRFH2Formatter	EpicMessage como RFH2
	JMSNMBDFormatter	Sólo datos del cuerpo
StringFormatterInterface	NMXMLFormatter	EpicMessage como XML

En la Tabla 12 se enumeran las clases LMS para las que se ofrece soporte y su grado de soporte de transacciones. Para obtener más información acerca de la utilización de transacciones con MQSeries Adapter Kernel, consulte el apartado “Posibilidades transaccionales” en la página 28.

Tabla 12. Clases LMS y soporte de transacciones

Clase LMS	Soporte de transacciones
LMSMQBindingMQPP	Fase única
LMSMQBindingMQRFH1	Fase única
LMSMQBindingMQRFH2	Fase única
LMSMQMQBD	Fase única
LMSMQBinding	Fase única

Tabla 12. Clases LMS y soporte de transacciones (continuación)

Clase LMS	Soporte de transacciones
LMSJMS	Fase única
LMSFILE	Sin soporte

Utilización del almacenamiento de objetos JMS

Los nombres de los objetos JMS se almacenan utilizando la implementación de archivo FSContext de JNDI, que se entrega como parte de MQSeries JMS SupportPac. El contexto (estructura de directorios) que utiliza el kernel para FSContext sigue la jerarquía de LDAP utilizando el atributo distintivo con el valor asociado para el nombre del directorio. Por ejemplo, para la jerarquía de LDAP `o=ePIC, o=ePICApplications, epicappid=TEST1`, la estructura de directorios es `o-ePIC/o-ePICApplications/epicappid-TEST1`.

Para crear el contexto y objetos, utilice la herramienta de administración de JMS que se proporciona con la instalación de JMS. Los pasos básicos consisten en definir un contexto y, a continuación, modificarlo. Al modificar el contexto, se entra en el contexto. Cree los objetos JMS en los lugares adecuados. A continuación, se incluyen mandatos de ejemplo para crear la estructura de contexto y los objetos JMS. En este ejemplo, el ID de la aplicación es `TEST1`.

```
#
# aqmjmscreatesample.scp 1.00 09Mar01
# Se utiliza para MQSeries Adapter Kernel
# Muestra de la configuración de AQM JMS
#
# Copyright (c) 2001 International Business Machines. Reservados todos los derechos.
#
# Este archivo de configuración es sólo un ejemplo.
#
# IBM NO OFRECE NINGUNA GARANTÍA O REPRESENTACIÓN, EXPLÍCITA O IMPLÍCITA, SOBRE LA
# DISPONIBILIDAD DE ESTA MUESTRA, INCLUIDAS PERO SIN LIMITARSE A ELLAS,
# LAS GARANTÍAS IMPLÍCITAS DE COMERCIALIZACIÓN, DE ADECUACIÓN A UN FIN
# DETERMINADO Y DE NO ABUSO.
#
# CopyrightVersion 1.0
#
#
# Éste es un script para utilizar con la herramienta de administración de JMS
# (JMSAdmin)
# que viene con MQSeries Support pac MA88.
# Esta herramienta requiere que se haya establecido JMSAdmin.config para utilizar
# FSCONTEXT (archivo) o LDAP. Este script se ha establecido para que funcione con
# FSCONTEXT, pero funcionará con LDAP con los siguientes cambios:
# - Cambiar el símbolo "-" a "=". Por ejemplo: define ctx(o-ePIC)
#   se convierte en define ctx(o=ePIC)
# - En LDAP los contextos ya tienen que estar definidos utilizando la
#   herramienta de administración de LDAP. Por ejemplo, no es necesario
```

```
#   "define ctx(o=ePIC) sino sólo convertirlo con el
#   mandato "change ctx(o-ePIC)"
# - En el siguiente script se encuentran algunas notas que resaltan
#   las diferencias cuando se utiliza LDAP.
#
#
# Uso de ejemplo: MQSeries root\java\bin\jmsadmin.bat < aqmjmscreatesample.scp
#
# Algunos mandatos de ayuda son:
# "display ctx" mostrará el contexto del contexto en el que se encuentra
# actualmente.
# "=UP" significa volver al contexto del elemento superior. Ejemplo: change ctx(=UP)
# "=INIT" significa volver al contexto raíz. En este ejemplo,un nivel de directorio
# superior a o-ePIC. Ejemplo: change ctx(=INIT)
# "define xxx" se utiliza para crear un contexto o un objeto.
# "change xxx" se utiliza para cambiar de contexto o moverse en él.
#
# Siempre es obligatorio.
define ctx(o-ePIC)
change ctx(o-ePIC)
# Siempre es obligatorio.
define ctx(o-ePICApplications)
change ctx(o-ePICApplications)
# El ID de la aplicación es TEST1. Necesita un contexto.
define ctx(epicappid-TEST1)
change ctx(epicappid-TEST1)
# Siempre es obligatorio.
define ctx(cn-epicadapterrouting)
change ctx(cn-epicadapterrouting)
# No contendrá el objeto JMS QueueConnectionFactory.
# Nota: Estos dos pasos no son necesarios para LDAP.
define ctx(cn-QCFTEST1)
change ctx(cn-QCFTEST1)
# Crear el objeto JMS QueueConnectionFactory cuyo nombre es QCFTEST1
# Utilizando MQSeries en modalidad de servidor (enlaces).
define qcf(QCFTEST1) qmgr(yourQManagerName) tran(BIND)
change ctx(=UP)
# BodyCategory es el valor por omisión (DEFAULT)
define ctx(epicbodycategory-DEFAULT)
change ctx(epicbodycategory-DEFAULT)
# BodyType es el valor por omisión (DEFAULT)
define ctx(epicbodytype-DEFAULT)
change ctx(epicbodytype-DEFAULT)
# No contendrá el objeto de cola JMS denominado TEST1AIQ.
# Nota: Estos dos pasos no son necesarios para LDAP.
define ctx(cn-TEST1AIQ)
change ctx(cn-TEST1AIQ)
# Crear el objeto de cola JMS cuyo nombre es TEST1AIQ
# q(Nombre de objeto de cola JMS) queue(Nombre de cola de MQSeries)
define q(TEST1AIQ) queue(TEST1AIQ)
# Se puede desplazar hacia arriba y definir otros contextos y objetos JMS.
# Salir de la herramienta de administración.
end
```

Apéndice B. Configuraciones validadas

Hay muchas configuraciones y combinaciones posibles de MQSeries, la Oferta MQSeries Adapter y MQSeries Integrator. Cada uno de estos miembros de la familia de productos MQSeries ofrece numerosas características y configuraciones y, además, se pueden combinar las funciones de MQSeries, la Oferta MQSeries Adapter y MQSeries Integrator. Algunas funciones de uno de los miembros de la familia de productos MQSeries se pueden solapar parcialmente con las funciones que proporcionan otros miembros de la familia. Debe determinar el modo en que desea utilizar y combinar las distintas funciones de direccionamiento y entrega de mensajes de MQSeries, la Oferta MQSeries Adapter y MQSeries Integrator.

En el momento de publicar este documento, ya se han validado algunas de las configuraciones siguientes de MQSeries, la Oferta MQSeries Adapter y MQSeries Integrator. En el sitio web de MQSeries puede encontrar las últimas configuraciones validadas.

MQSeries Adapter Kernel:

- Envío de un mensaje con acuse de recibo solicitado y sin acuse de recibo solicitado.
- Utilización de las modalidades de comunicación MQSeries o JMS. Si desea obtener más información acerca de las modalidades de comunicación válidas , consulte el "Apéndice A. Modalidades de comunicación" en la página 103.
- Direccionamiento y entrega de mensajes:
  - Enviar un mensaje de un adaptador origen a un adaptador destino
  - Enviar un mensaje de un adaptador origen a varios adaptadores destino
  - Entrega de mensajes de múltiples hebras, es decir, varios trabajos
  - Con el identificador lógico de destinación establecido en NONE en el mensaje, de modo que se utilice el archivo de configuración del kernel para determinar el identificador lógico de destinación según la categoría y el tipo de cuerpo y el identificador lógico de origen
  - Modelo de entrega de inserción
  - Con la función de rastreo habilitada

© Copyright IBM Corp. 2000, 2001

Nota: consulte el "Apéndice C. Cabeceras de mensaje" en la página 111 . Contiene los campos de cabecera de mensaje de MQSeries Adapter Kernel que llena y procesa el kernel.

- Con los requisitos previos que se muestran en los apartados "Hardware" en la página 31 y "Software" en la página 32.
- Utilizar el archivo de configuración, no LDAP, para definir la configuración.

MQSeries:

- No utilizar clústers de MQSeries.

  Nota: Consulte el "Apéndice C. Cabeceras de mensaje" en la página 111. Contiene los campos de cabecera de mensaje de MQSeries Adapter Kernel que llena y procesa el kernel.

MQSeries Integrator:

- MQSeries Adapter Kernel y MQSeries pueden direccionar y entregar el mensaje a MQSeries Integrator. Consulte la información de MQSeries Integrator para determinar sus posibilidades para intermediar en estos mensajes.
- Enviar mensajes de la parte origen del kernel, a través de MQSeries y MQSeries Integrator versión 2, y direccionarlos directamente a la parte destino del kernel. En MQSeries Integrator, el flujo de mensajes se configura para el direccionamiento estático. Todos los mensajes que llegan al nodo MQInput del flujo se direccionan directamente a una cola MQOutput específica.

  Nota: consulte el "Apéndice C. Cabeceras de mensaje" en la página 111. Contiene los campos de cabecera de mensaje de MQSeries Adapter Kernel que llena y procesa el kernel.

Apéndice C. Cabeceras de mensaje

La Oferta MQSeries Adapter utiliza numerosas cabeceras de mensaje. En el apartado “Mensaje y formato del mensaje” en la página 12 se proporciona información acerca de las cabeceras que se utilizan y en qué circunstancias.

En este apéndice se enumeran y describen los campos de las cabeceras de mensaje.

Cabecera del descriptor de mensaje de MQSeries Adapter Kernel

Los valores de cabecera utilizados por MQSeries Adapter Kernel. Estos valores se colocan en los objetos de soporte de mensajes. La columna **¿Propagado en las respuestas?** presenta una lista dependiendo de si el valor se ha vuelto a propagar a la aplicación de origen en un mensaje de respuesta cuando la aplicación de origen solicita una respuesta. Algunos valores sólo se utilizan con WebSphere Business Integrator.

Tabla 13. cabecera de MQSeries Adapter Kernel

Nombre de cabecera	¿Propagado en las respuestas?	Significado o utilización
UniqueID	No	Identificador exclusivo para cada mensaje.
TransactionID	Sí	Identificador transaccional que comparten todos los mensajes y sus respuestas, si las tienen. Equivalente a un Extricity PublicProcessID o a un DataInterchange (DI) ApplicationID.
MessageType	No	Utilizado para los mensajes de pasarela y anotación/seguimiento/ excepción.
SourceLogicalID	No	Identificador lógico de la aplicación de origen. Equivalente a los nombres reservados en DI, Partner Agreement Manager (PAM) y Business Flow Manager (BFM).

© Copyright IBM Corp. 2000, 2001

Tabla 13. cabecera de MQSeries Adapter Kernel (continuación)

Nombre de cabecera	¿Propagado en las respuestas?	Significado o utilización
DestinationLogicalID	No	Identificador lógico de la aplicación de destino. El valor por omisión para DI y PAM es `none`, pero se puede modificar.
RespondToLogicalID	Sí	Identificador lógico al que se va a enviar un mensaje de respuesta. Copiado en DestinationLogicalID para DI y en SourceLogicalID para PAM.
CorrelationID	No	Uso reservado.
GroupStatus	No	Uso reservado.
ProcessingCategory	No	Equivalente a un PAM Public Process Identifier o un DI Command Process Identifier.
QosPolicy	No	Uso reservado.
DeliveryCategory	No	Equivalente a un DI RequestorProfileID.
AckRequested	No	Determina si la aplicación de origen solicita un mensaje de respuesta.
PublicationTopic	No	Uso reservado.
SessionID	No	Equivalente a un DI BatchID.
EncryptionStatus	No	Determina el tipo de firma y cifrado del cuerpo.
TimeStampCreated	No	Fecha y hora en la que se ha creado el mensaje.
TimeStampExpired	No	Fecha y hora a partir de la que el mensaje deja de ser significativo. Un valor de `-1` significa que no caduca.
Size	No	Uso reservado.
BodyType	No	Representa la finalidad específica del mensaje.
BodyCategory	No	Representa el tipo de aplicación del mensaje.

Tabla 13. cabecera de MQSeries Adapter Kernel (continuación)

Nombre de cabecera	¿Propagado en las respuestas?	Significado o utilización
BodySecondaryType	No	Uso reservado.
UserArea	No	Área general para datos de usuario.
RelatedSubjectID	No	Utilizado para la correlación de interprocesos.
ExternalID	No	Identificador del propietario actual (por ejemplo, un usuario o socio comercial) fuera del entorno de aplicación.
InternalID	No	Identificador del propietario actual (por ejemplo, un usuario o socio comercial) dentro del entorno de aplicación.
BodySignature	No	Uso reservado.
TransportCorrelationID	Sí	Uso reservado.

Cabecera de descriptor de mensaje de MQSeries

MQSeries determina el contenido de los campos. La Oferta MQSeries Adapter coloca los mensajes en las colas, tal como determinan los valores de control de mensaje. Si desea obtener más información, consulte el apartado "Valores de control de mensaje" en la página 16.

Tabla 14. Cabecera de MQSeries

Sección o campo	Significado o utilización
Revision	Fijada.
UniqueID	Cada mensaje tiene un identificador exclusivo.
TransactionID	Un mensaje y su respuesta comparten el mismo identificador de transacción.
MessageType	Uso reservado.
SourceLogicalID	Identificador lógico de la aplicación de origen.
DestinationLogicalID	Identificador lógico de la aplicación de destino.
RespondToLogicalID	Identificador lógico al que se va a enviar el mensaje de respuesta.
CorrelationID	Uso reservado.

Tabla 14. Cabecera de MQSeries (continuación)

GroupStatus	Uso reservado.
ProcessingCategory	Uso reservado.
QosPolicy	Uso reservado.
DeliveryCategory	Uso reservado.
AckRequested	Determina si la aplicación de origen solicita una respuesta.
PublicationTopic	Uso reservado.
SessionID	Uso reservado.
EncryptionStatus	Uso reservado.
TimeStampCreated	Fecha y hora en la que se ha creado el mensaje.
TimeStampExpired	Fecha y hora a partir de la que el mensaje deja de ser significativo.
Size	Uso reservado.
BodyCategory	Representa el tipo de aplicación del mensaje, por ejemplo, OAG o RosettaNet.
BodyType	Representa la finalidad específica del mensaje, por ejemplo, añadir pedido de venta o sincronizar inventario.
BodySecondaryType	Reservado.
UserArea	Área general para datos de usuario.
BodyData	Datos del cuerpo del mensaje.

MQSeries sin MQSeries Integrator

Los valores de cabecera del kernel y los datos del cuerpo se colocan en un documento XML. A continuación, se incluye un ejemplo de DTD que describe el documento XML:

```
<!ELEMENT EPICHEADER (HEADER, EPICBODY,USERAREA*)>
<!ELEMENT HEADER (#PCDATA)>
<!ATTLIST HEADER Revision CDATA #FIXED "001">
<!ATTLIST HEADER UniqueID CDATA #REQUIRED>
<!ATTLIST HEADER TransactionID CDATA #REQUIRED>
<!ATTLIST HEADER MessageType CDATA #REQUIRED>
<!ATTLIST HEADER SourceLogicalID CDATA #REQUIRED>
<!ATTLIST HEADER DestinationLogicalID CDATA #REQUIRED>
<!ATTLIST HEADER RespondToLogicalID CDATA #IMPLIED>
<!ATTLIST HEADER CorrelationID CDATA #IMPLIED>
<!ATTLIST HEADER GroupStatus CDATA #IMPLIED>
<!ATTLIST HEADER ProcessingCategory CDATA #IMPLIED>
<!ATTLIST HEADER QosPolicy CDATA #IMPLIED>
```

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

Cabecera de la versión 1 de MQSeries Integrator

La cabecera de la versión 1 de MQSeries Integrator, RFH1, consta de los elementos siguientes:

1. Sección fija
2. Cabecera de Neon
3. Sección de datos, que contiene la cabecera del kernel y los datos del cuerpo del mensaje

Tabla 15. Cabecera de la versión 1 de MQSeries Integrator — RFH1

Sección o campo	**Significado o utilización**
Sección fija	Se utiliza tal como se especifica en MQSeries Integrator versión 1.1.
Cabecera de Neon	Sigue el formato de cabecera de Neon.
OPT_APP_GRP	Valor de SourceLogicalId. Se toma de la cabecera del kernel.
OPT_MSG_TYPE	BodyCategory+BodyType. Derivado de la cabecera del kernel. Ejemplo: Si BodyCategory es OAG y BodyType es SyncItem, el valor es OAG+SyncItem.
Sección de datos	Consta de los valores de cabecera del kernel seguidos de los datos del cuerpo del mensaje.
Cabecera del kernel	La cabecera del kernel se incluye dentro de los códigos <EPICHEADER>*cabecera*</EPICHEADER>. Los valores de cabecera del kernel están en la sintaxis XML. Sólo están presentes los atributos con valores. Los datos reales no están en líneas separadas. Ejemplo del formato de un valor: <MessageType>*valor*</MessageType>.

Tabla 15. Cabecera de la versión 1 de MQSeries Integrator — RFH1 (continuación)

MessageType	Uso reservado.
SourceLogicalID	Identificador lógico de la aplicación de origen.
DestinationLogicalID	Identificador lógico de la aplicación de destino.
RespondToLogicalID	Identificador lógico al que se va a enviar el mensaje de respuesta.
TimeStampCreated	Fecha y hora en la que se ha creado el mensaje.
TimeStampExpired	Fecha y hora a partir de la que el mensaje deja de ser significativo.
TransactionID	Un mensaje y su respuesta comparten el mismo identificador de transacción.
UniqueID	Cada mensaje tiene un identificador exclusivo.
AckRequested	Determina si la aplicación de origen solicita una respuesta.
ProcessingCategory	Reservado.
BodyCategory	Representa el tipo de aplicación del mensaje, por ejemplo, OAG o RosettaNet.
BodyType	Representa la finalidad específica del mensaje, por ejemplo, añadir pedido de venta o sincronizar inventario.
BodySecondaryType	Reservado.
UserArea	Datos de aplicación específicos de la integración de usuario.
MsgHeaderVersion	Versión de la cabecera del kernel (reservado).
CorrelationID	Específico de la integración de usuario.
GroupStatus	Específico de la integración de usuario.
QosPolicy	Reservado.
DeliveryCategory	Reservado.
PublicationTopic	Reservado.
SessionID	Reservado.
EncryptionStatus	Reservado.
Message body data	Datos del cuerpo del mensaje.

Cabecera de la versión 2 de MQSeries Integrator

La cabecera de la versión 2 de MQSeries Integrator, RFH2, consta de los elementos siguientes:

1. Sección fija
2. Carpeta <mcd> — descriptor del contenido del mensaje
3. Carpeta <usr> — propiedades definidas de la aplicación (usuario)
4. Sección de datos, que contiene la cabecera del kernel y los datos del cuerpo del mensaje

Tabla 16. Cabecera de la versión 2 de MQSeries Integrator — RFH2

Sección o campo	**Significado o utilización**
Sección fija	Se utiliza tal como se especifica en MQSeries Integrator versión 2.
<mcd>	XML si es un mensaje XML. Siga las normas de MQSeries Integrator versión 2.
set	El kernel no lo utiliza.
type	El kernel no lo utiliza.
format	XML si es un mensaje XML. Siga las normas de MQSeries Integrator versión 2.
Carpeta <usr> — propiedades definidas de la aplicación (usuario)	Consta de los valores de cabecera del kernel.
Cabecera del kernel	Sólo están presentes los atributos con valores. Los datos reales no están en líneas separadas.
SourceLogicalID	Identificador lógico de la aplicación de origen.
DestinationLogicalID	Identificador lógico de la aplicación de destino.
MessageType	Uso reservado.
RespondToLogicalID	Identificador lógico al que se va a enviar el mensaje de respuesta.
TimeStampCreated	Fecha y hora en la que se ha creado el mensaje.
TimeStampExpired	Fecha y hora a partir de la que el mensaje deja de ser significativo.
TransactionID	Un mensaje y su respuesta comparten el mismo identificador de transacción.
UniqueID	Cada mensaje tiene un identificador exclusivo.
ProcessingCategory	Reservado.
BodyCategory	Representa el tipo de aplicación del mensaje, por ejemplo, OAG o RosettaNet.

Tabla 16. Cabecera de la versión 2 de MQSeries Integrator — RFH2 (continuación)

BodyType	Representa la finalidad específica del mensaje, por ejemplo, añadir pedido de venta o sincronizar inventario.
BodySecondaryType	Reservado.
AckRequested	Determina si la aplicación de origen solicita una respuesta.
UserArea	Datos de aplicación específicos de la integración de usuario.
MsgHeaderVersion	Versión de la cabecera del kernel (reservado).
CorrelationID	Específico de la integración de usuario.
GroupStatus	Específico de la integración de usuario.
QosPolicy	Reservado.
DeliveryCategory	Reservado.
PublicationTopic	Reservado.
SessionID	Reservado.
EncryptionStatus	Reservado.
Sección de datos	Datos del cuerpo del mensaje.

Apéndice D. Ejemplo del archivo de configuración

Este apartado contiene la lista de la versión del archivo `aqmconfig.xml` actual en el momento de la publicación de este documento. El apartado "Ejemplo de un archivo de configuración mínima" en la página 123 contiene la lista de la versión del archivo `aqmconfig.minimum.xml` actual en el momento de la publicación de este documento. Para obtener la versión más reciente, consulte los archivos `aqmconfig.xml` y `aqmconfig.minimum.xml` del directorio `samples` de la instalación del kernel. Es posible que los ejemplos que se incluyen aquí hayan quedado obsoletos.

Consulte el apartado "El archivo de configuración" en la página 65 para obtener más información acerca de la interpretación y la edición del archivo de configuración.

En este archivo de configuración de ejemplo se han incluido varios identificadores de aplicación. Bajo cada identificador de aplicación se enumeran un conjunto de entradas. El archivo de configuración de ejemplo contiene los identificadores de aplicación siguientes:

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

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

   <Epic o="ePIC">
      <!-- If getObject is called this indicates the top level directory -->
      <!-- where the JNDI file system context will retrieve objects from. -->
      <!-- This defaults to the current directory if this key is not present. -->
      <!-- All applications share this context root. -->
      <context>file:///epic/configContext</context>
      <!-- Example using a drive letter 'c' -->
      <!--
      <context>file://c:/E/runtimefiles</context>
      -->
      <ePICApplications o="ePICApplications">
         <!-- The following is for sample Test Application ID: TEST1 with a  -->
         <!-- sample AdapterDaemon named TEST1Daemon  -->
```

© Copyright IBM Corp. 2000, 2001

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

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

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

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

Ejemplo de un archivo de configuración mínima

En este apartado se proporciona un ejemplo de un archivo de configuración mínima para su utilización con MQSeries Adapter Kernel. Para obtener más información acerca del archivo de configuración mínima, consulte el apartado "Adición de información del adaptador a la configuración" en la página 85.

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

```
<!--    and using default format.                                              -->
<!-- 5) No specific body category and body type being used.                   -->
<!-- 6) Using default tracing to the console.   -->
<!--  -->
<!--  -->
<!--  -->
<!-- Copyright (c) 2000 International Business Machines. All Rights Reserved. -->
<!--  -->
<!-- This configuration file is as an example only. -->
<!--  -->
<!-- IBM MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF THIS -->
<!-- SAMPLE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE -->
<!-- IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR -->
<!-- PURPOSE, OR NON-INFRINGEMENT. -->
<!--  -->
<!-- CopyrightVersion 1.0 -->
<!--     -->
   <Epic o="ePIC">
      <ePICApplications o="ePICApplications">
         <!-- The following is for sample Test Application ID: TEST1  -->
         <ePICApplication epicappid="TEST1">
     <!--  Tracing on/off.  If no entry defaults to false. -->
           <epictrace>false</epictrace>
     <!-- Trace levels - 512=TYPE_ERROR_EXC (Exceptions),-1=TYPE_ALL (All possible messages). -->
           <epictracelevel>0</epictracelevel>
           <AdapterRouting cn="epicadapterrouting">
              <epicmqpppqueuemgr>DEFAULT</epicmqpppqueuemgr>
              <ePICBodyCategory epicbodycategory="DEFAULT">
                 <ePICBodyType epicbodytype="DEFAULT">
     <!-- Default destinations to send messages to. -->
                     <epicdestids>TEST2</epicdestids>
                 </ePICBodyType>
              </ePICBodyCategory>
           </AdapterRouting>
         </ePICApplication>
         <!-- The following is for Test Application ID: TEST2 -->
         <ePICApplication epicappid="TEST2">
            <epictrace>false</epictrace>
            <epictracelevel>512</epictracelevel>
            <AdapterRouting cn="epicadapterrouting">
               <epicmqpppqueuemgr>DEFAULT</epicmqpppqueuemgr>
               <ePICBodyCategory epicbodycategory="DEFAULT">
                  <ePICBodyType epicbodytype="DEFAULT">
      <!-- AdapterDaemon - Command to invoke. -->
                     <epiccommandclassname>com.ibm.epic.adapters.eak.samples.SampleCAdapterWrapper
                     </epiccommandclassname>
                     <epicreceivemode>MQ</epicreceivemode>
      <!-- Receive Time out in milliseconds ie. 1000 = 1 second, -->
      <!-- -1 means never ending.  No entry defaults to 0. -->
      <!-- milliseconds.  Used when receiving messages. -->
                     <epicreceivetimeout>30000</epicreceivetimeout>
                     <epicreceivemqpppqueue>TEST2AIQ</epicreceivemqpppqueue>
                     <epicerrormqpppqueue>TEST2AEQ</epicerrormqpppqueue>
                     <epicreplymqpppqueue>TEST2RPL</epicreplymqpppqueue>
                  </ePICBodyType>
               </ePICBodyCategory>
            </AdapterRouting>
         </ePICApplication>
      </ePICApplications>
   </Epic>
```

Apéndice E. Ejemplo del archivo de instalación

Lo siguiente es un ejemplo del archivo aqmsetup, que define varios valores de configuración inicial del kernel, entre los que se incluyen numerosas variables de entorno. Para obtener más información acerca de este archivo, consulte el apartado "El archivo de instalación" en la página 64. El archivo aqmsetup está situado en el directorio samples del directorio de instalación raíz del kernel.

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

© Copyright IBM Corp. 2000, 2001

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

Avisos

Esta información se ha desarrollado para productos y servicios que se ofrecen en los Estados Unidos. Es posible que en otros países IBM no ofrezca los productos, los servicios o las características que se describen en este documento. Póngase en contacto con el representante local de IBM que le informará sobre los productos y servicios disponibles actualmente en su localidad. Las referencias a programas, productos o servicios de IBM no pretenden indicar ni implicar que sólo puedan utilizarse los productos, programas o servicios de IBM. En su lugar, se puede utilizar cualquier producto, programa o servicio funcionalmente equivalente que no infrinja ninguno de los derechos de propiedad intelectual de IBM. No obstante, es responsabilidad del usuario evaluar y comprobar el funcionamiento de cualquier producto, programa o servicio que no sea de IBM.

IBM puede tener patentes o solicitudes de patentes en trámite que afecten a los temas tratados en esta información. La posesión de esta información no confiere ninguna licencia sobre dichas patentes. Puede enviar consultas sobre licencias, por escrito, a:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
Estados Unidos

Para consultas sobre licencias en las que se solicite información sobre el juego de caracteres de doble byte (DBCS), póngase en contacto con el departamento de propiedad intelectual de IBM de su país o envíe directamente las consultas por escrito a:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokio 106, Japón

El párrafo siguiente no se aplica al Reino Unido ni a ningún otro país donde estas disposiciones sean incompatibles con la legislación vigente:
INTERNATIONAL BUSINESS MACHINES CORPORATION FACILITA ESTA PUBLICACIÓN "TAL CUAL" SIN GARANTÍAS DE NINGÚN TIPO, NI EXPLÍCITAS NI IMPLÍCITAS, INCLUYENDO, PERO SIN QUE ELLO CONSTITUYA UN LÍMITE, LAS GARANTÍAS IMPLÍCITAS DE NO INFRACCIÓN, COMERCIALIZACIÓN O ADECUACIÓN A UN FIN CONCRETO. Algunos estados o países no permiten la renuncia a las garantías

© Copyright IBM Corp. 2000, 2001

explícitas o implícitas en ciertas transacciones, por tanto, es posible que esta declaración no resulte aplicable a su caso.

Esta información puede contener imprecisiones técnicas o errores tipográficos. Periódicamente se efectúan cambios en la información aquí contenida; dichos cambios se incorporarán en nuevas ediciones de la información. IBM se reserva el derecho a realizar, si lo considera oportuno, cualquier modificación en los productos o programas que se describen en esta información y sin notificarlo previamente.

Las referencias de esta información a sitios web que no sean de IBM se proporcionan únicamente como ayuda y no se consideran en modo alguno documentos o sitios web aprobados por IBM. Los materiales de dichos sitios web no forman parte de este producto de IBM y la utilización de los mismos será por cuenta y riesgo del usuario.

IBM puede utilizar o distribuir la información que se le suministre de cualquier modo que considere adecuado sin incurrir por ello en ninguna obligación con el remitente.

Los titulares de licencias de este programa que deseen información sobre el mismo con el fin de permitir: (i) el intercambio de información entre programas creados independientemente y otros programas (incluido éste) y (ii) la utilización mutua de la información intercambiada, deben ponerse en contacto con:

IBM United Kingdom Laboratories,
Mail Point 151,
Hursley Park,
Winchester,
Hampshire,
Inglaterra
SO21 2JN.

Dicha información puede estar disponible, sujeta a los términos y condiciones adecuados, incluido, en algunos casos, el pago de una tasa.

El programa bajo licencia que se describe en esta información, y todos los materiales bajo licencia disponibles para el mismo, los proporciona IBM bajo los términos del Contrato con el cliente IBM, del Contrato Internacional de Licencias para Programas IBM o de cualquier contrato equivalente entre el cliente e IBM.

Todos los datos de rendimiento contenidos en el presente documento se han determinado en un entorno controlado. Por consiguiente, los resultados obtenidos en otros entornos operativos pueden variar considerablemente. Algunas mediciones pueden haberse realizado en sistemas de nivel de

desarrollo, por lo que no existe ninguna garantía de que dichas mediciones sean iguales en los sistemas disponibles en el mercado. Además, alguna medición se puede haber estimado mediante extrapolación. Los resultados reales pueden variar. Los usuarios de este documento deben verificar los datos aplicables para su entorno específico.

La información relacionada con productos que no son de IBM se ha obtenido de los proveedores de dichos productos, de sus anuncios publicados o de otras fuentes de disponibilidad pública. IBM no ha comprobado estos productos y no puede confirmar la precisión de su rendimiento, compatibilidad ni ninguna otra reclamación relacionada con los productos que no son de IBM. Las cuestiones relacionadas con las posibilidades de los productos que no son de IBM deberán dirigirse a los proveedores de estos productos.

Marcas registradas

Los términos siguientes son marcas registradas de International Business Machines Corporation en los Estados Unidos y/o en otros países:

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

Lotus y LotusScript son marcas registradas de Lotus Development Corporation en los Estados Unidos y/o en otros países.

Java y todos los logotipos y marcas registradas basados en Java son marcas registradas de Sun Microsystems, Inc. en los Estados Unidos o en otros países.

Windows, Windows NT y el logotipo de Windows son marcas registradas de Microsoft Corporation en los Estados Unidos y/o en otros países.

Otros nombres de empresas, productos y servicios pueden ser marcas registradas de terceros.

Glosario

El glosario contiene elementos *clave* y sus significados, tal como se utilizan en la documentación de MQSeries Adapter Kernel.

Si un término o concepto determinado sólo aparece en un apartado, es posible que no esté incluido en el glosario, pero puede encontrarlo en el "Índice" en la página 137.

El glosario no incluye términos de otros productos IBM como, por ejemplo, MQSeries.

adaptador. Salida de MQSeries Adapter Builder. Normalmente, el usuario genera cada adaptador para que sea específico para *un tipo de mensaje* que se envía a o desde una aplicación. Por este motivo, los adaptadores en sí mismos no forman parte de la Oferta MQSeries Adapter. Un adaptador consta de código fuente C o Java que se compila en una biblioteca compartida. Cuando se ejecutan al mismo tiempo los adaptadores y MQSeries Adapter Kernel, realizan la función de ejecución de la Oferta MQSeries Adapter. Dependiendo del modo en que lo ha modelado el usuario en MQSeries Adapter Builder, el adaptador puede contener una gran variedad de funciones como, por ejemplo, control de flujo; flujo de datos; navegación secuencial; ramificación condicional, que incluye decisión e iteración; escritura de datos; almacenamiento de contexto de datos; transformación de elementos de datos; control transaccional; operaciones lógicas y código personalizado. Los adaptadores creados se pueden volver a utilizar.

Consulte "tipo de mensaje" en la página 135, "aplicación de origen" en la página 132 y "aplicación de destino" en la página 132.

adaptador de servicio Java. Un tipo de adaptador de lenguaje Java que, en un entorno JMS Listener ofrece las funciones de un daemon del adaptador, trabajo y adaptador destino.

adaptador destino. Adaptador que realiza las tareas siguientes:

- Recibe un mensaje (del kernel y MQSeries u otro software de mensajería) que ha enviado un adaptador origen.
- Procesa el mensaje de integración según el modo en que se ha modelado el adaptador.
- Transforma el mensaje de integración en un mensaje con formato específico de la aplicación que puede recibir la aplicación de destino.
- Envía el mensaje a la aplicación de destino utilizando una interfaz específica de la aplicación.
- Informa al trabajo acerca del momento en que ha finalizado el envío del mensaje a la aplicación de destino, con el fin de que el trabajo pueda enviar un acuse de recibo.

Si la aplicación de destino puede recibir el mensaje de integración, es posible que no se necesite un adaptador destino.

Hay un adaptador destino para cada tipo de mensaje. Generalmente, una aplicación de destino puede aceptar varios tipos de mensajes y, por consiguiente, en muchos casos, varios adaptadores destino ofrecen soporte para una aplicación de destino. Consulte "adaptador".

adaptador nativo. Software que se utiliza para enviar y recibir objetos de soporte de mensajes.

adaptador nativo de MQSeries Adapter Kernel. Sinónimo de adaptador nativo.

© Copyright IBM Corp. 2000, 2001

adaptador origen. Adaptador que realiza las tareas siguientes:

- Acepta o, por el contrario, obtiene datos estructurados de una aplicación de origen (generalmente, utilizando una interfaz específica de la aplicación que se desarrolla fuera del adaptador).
- Procesa los datos estructurados según el modo en que se ha modelado el adaptador.
- Transforma los datos estructurados en un formato de mensaje de integración.
- Utilizando el kernel, coloca el mensaje en una cola de mensajes para entregarlo a uno o más adaptadores destino y, desde allí, a la aplicación de destino.

Hay un adaptador origen para cada tipo de mensaje. Generalmente, una aplicación de origen puede enviar varios tipos de mensajes y, por consiguiente, en muchos casos, varios adaptadores origen ofrecen soporte para una aplicación de origen.

Consulte “adaptador” en la página 131.

aplicación de destino. Programa que se necesita para recibir datos a través de una red de sistemas desde un programa (denominado aplicación de origen) que, por lo general, reside en otro sistema.

aplicación de origen. Programa que se necesita para enviar datos a través de una red de sistemas a un programa (denominado aplicación de destino) que, por lo general, reside en otro sistema.

archivo aqmconfig.xml. Consulte “archivo de configuración”.

archivo aqmsetup. Consulte “archivo de instalación”.

archivo de configuración. Archivo `aqmconfig.xml`, que contiene la mayor parte de los valores de configuración del kernel. Si desea obtener más información, consulte el apartado “El archivo de configuración” en la página 65.

archivo de instalación. Archivo que contiene muchos de los valores iniciales del kernel. El nombre del archivo por omisión es `aqmsetup`.

bean de mensajes de trabajo. Un Enterprise Bean que realiza la función de un trabajo cuando se utiliza WebSphere Application Server en la parte destino del kernel.

BOD. Documento de objeto comercial. Es una representación de un proceso de negocio estándar que fluye en una organización o entre organizaciones. Algunos ejemplos incluyen: añadir pedido de compra, mostrar la disponibilidad del producto y añadir pedido de venta. OAG define los documentos BOD utilizando XML. Consulte “OAG” en la página 134 y “XML” en la página 135.

La Oferta MQSeries Adapter utiliza BOD para definir cuerpos de mensaje en sus mensajes de integración.

categoría del cuerpo. Datos contenidos en un mensaje que representan el tipo de aplicación del mensaje, por ejemplo, OAG o RosettaNet. Pertenece al conjunto de valores de control de mensaje. Consulte “valores de control de mensaje” en la página 135.

La categoría del cuerpo también ayuda a especificar el tipo de mensaje. Consulte “tipo de mensaje” en la página 135.

clase de inicio de sesión. Clase Java específica de cada aplicación de destino que se puede utilizar como ayuda para la entrega del mensaje a la aplicación de destino. La clase de inicio de sesión sólo es obligatoria cuando el adaptador destino debe iniciar la sesión en la aplicación de destino antes de entregar el mensaje. El usuario crea todas las clases de inicio de sesión y el trabajo las convierte en instancia. La clase de inicio de sesión busca en el archivo de configuración los valores que necesita el adaptador destino para ofrecer soporte para la interfaz específica de la aplicación para la aplicación de destino. Por lo general, estos valores son parámetros de inicio de sesión. De este modo, el adaptador destino puede disponer de dichos valores.

Junto con el kernel se proporciona una clase de inicio de sesión ficticia que no realiza ninguna función.

cliente de rastreo. Componente del kernel que escribe mensajes de rastreo.

cola de errores. En la terminología de la Oferta MQSeries Adapter, una cola de mensajes que se utiliza cuando no se puede procesar un mensaje que se ha obtenido de una cola de recepción.

cola de recepción. En la terminología de la Oferta MQSeries Adapter, una cola de mensajes que se utiliza como cola principal de entrada, para recibir mensajes. Puede haber varias colas de recepción para cada aplicación de destino, pero sólo puede haber una cola de recepción para cada combinación de identificador de aplicación, categoría y tipo de cuerpo.

cola de respuestas. Cola de mensajes que se utiliza para recibir respuestas. Se utiliza con el método `sendRequestResponse` del kernel.

daemon del adaptador. Software ejecutable que forma parte del kernel. El daemon del adaptador sólo se utiliza en el modelo de entrega de inserción. Su finalidad es convertir los trabajos en instancia. Después de iniciarlo, el daemon del adaptador permanece activo. Para cada aplicación de destino, puede haber uno o más daemons de adaptador.

En algunos casos, el daemon del adaptador realiza la función de una aplicación de destino. Lleva a cabo las funciones necesarias como, por ejemplo, utilizar un adaptador destino para enviar un mensaje de correo electrónico o para grabar un registro en un archivo.

DTD. Definición de tipo de documento. En XML, generalmente, es un archivo (o varios archivos que se utilizan al mismo tiempo) que contiene una definición formal de un tipo de documento determinado. Especifica los nombres que se pueden utilizar para los elementos de DTD, donde se pueden producir los elementos dentro de DTD y el modo en que se ajustan los elementos a la vez. En la Oferta MQSeries Adapter, puede utilizar DTD para definir cuerpos de mensaje. Consulte "XML" en la página 135 y "mensaje de integración" en la página 134.

formato neutro de la aplicación. Consulte "mensaje de integración" en la página 134.

identificador de aplicación de dependencia. Nombre de la aplicación a la que atiende el trabajo. El trabajo obtiene el identificador de aplicación de dependencia del archivo de configuración según el nombre del daemon del adaptador.

identificador lógico de aplicación. Identificador que representa la aplicación a la que está asociado un adaptador (tanto de origen como de destino). Consulte "identificador lógico de origen" e "identificador lógico de destino".

identificador lógico de destinación. Valor que representa la aplicación de destino. El kernel lo utiliza, junto con otros valores de control de mensaje, para direccionar y clasificar mensajes. Consulte "valores de control de mensaje" en la página 135 .

identificador lógico de destino. Valor que representa la aplicación de destino asociada a un adaptador destino. Consulte "identificador lógico de destino" e "identificador lógico de aplicación".

identificador lógico de origen. Valor que representa la aplicación de origen. El kernel lo utiliza, junto con otros valores de control de mensaje, para direccionar y clasificar mensajes. Consulte "valores de control de mensaje" en la página 135, "identificador lógico de aplicación" e "identificador lógico de destino".

identificador lógico de respuestas. Identificador lógico de la aplicación al que se envían las respuestas cuando se solicita una respuesta. Toma el valor por omisión del identificador lógico de origen del mensaje.

interfaz específica de la aplicación. Interfaz que se ha desarrollado fuera de la Oferta MQSeries Adapter para una de las finalidades siguientes:

- Permitir que el adaptador origen obtenga un mensaje de la aplicación de origen
- Permitir que la aplicación de destino obtenga un mensaje del adaptador destino.

JMS Listener. Un componente proporcionado por el producto WebSphere Business Integrator que permite la integración entre MQSeries Adapter Kernel y WebSphere Application Server Advanced Edition.

kernel. Sinónimo de MQSeries Adapter Kernel.

mensaje. En MQSeries, que incluye la Oferta MQSeries Adapter, una colección de datos que envía un programa a otro programa.

mensaje de comunicaciones. Cualquier información específica del transporte de comunicaciones, además del objeto de soporte de mensajes, convertida en un formato de mensajería específico del transporte de comunicaciones que se está utilizando.

mensaje de integración. Mensaje que consta de datos de aplicación en un formato neutro de la aplicación para la integración. Por ejemplo, un documento XML que transforma el adaptador origen del formato de la aplicación de origen a XML.

mensajes de rastreo. Mensajes que contienen el estado del proceso de un mensaje en un punto determinado del kernel. Puede utilizar mensajes de rastreo como ayuda para el diagnóstico de problemas con el kernel o con los adaptadores.

Consulte “rastreo” en la página 135.

modalidad de comunicación. Modalidad que utiliza el kernel para transportar el mensaje y para realizar servicios de intermediario.

modelo de entrega de extracción. Consulte “modelos de entrega”.

modelo de entrega de inserción. Consulte “modelos de entrega”.

modelos de entrega. El kernel utiliza dos modelos para tener interfaz con la aplicación de destino. Son los siguientes:

inserción
El kernel se encarga de iniciar y gestionar la entrega del mensaje a la aplicación de destino. Generalmente, este modelo no requiere que se realice ninguna modificación en la aplicación de destino para ofrecer soporte para la Oferta MQSeries Adapter.

extracción
La aplicación de destino se encarga de gestionar la entrega del mensaje. Este modelo requiere llevar a cabo modificaciones en la aplicación de destino para ofrecer soporte para la Oferta MQSeries Adapter. La aplicación de destino debe gestionar la interfaz del kernel para la aplicación de destino.

MQSeries Adapter Builder. Software que permite que un usuario genere un adaptador prácticamente para cualquier aplicación utilizando una interfaz gráfica de usuario (GUI).

MQSeries Adapter Kernel. Conjunto de API y varios programas ejecutables, en C y Java, y archivos de configuración. El kernel ofrece soporte y trabaja con adaptadores. Consulte “adaptador” en la página 131. Además de ofrecer directamente soporte para los adaptadores, el kernel realiza funciones relacionadas y, entre las más importantes, se incluyen las siguientes: direccionamiento de mensajes y servicios de infraestructura como, por ejemplo, construcción de mensajes, rastreo e interfaz con MQSeries u otro software de mensajería.

OAG. Open Applications Group. Consorcio del sector sin ánimo de lucro que comprende muchos de los partícipes más destacados del campo de interoperatividad de componentes de software comercial. OAG define BOD (documentos de objetos de negocio).

objeto de soporte de mensajes. Contenedor para los metadatos que utiliza el kernel para encapsular un mensaje de integración y otros datos de control.

Oferta MQSeries Adapter. Conjunto de productos de integración de aplicaciones que consta de MQSeries Adapter Builder y MQSeries Adapter Kernel.

parte destino del kernel. Parte de la función del kernel que se inicia cuando se obtiene el mensaje de una cola de mensajes y finaliza cuando se envía el mensaje al adaptador destino.

parte origen del kernel. Parte de la función del kernel que se inicia cuando se recibe el mensaje desde el adaptador origen y finaliza cuando se coloca el mensaje en una cola de mensajes.

rastreo. Colección de procesos que utiliza el kernel para escribir mensajes de rastreo. Consulte "mensajes de rastreo" en la página 134.

servicio lógico de mensajes. Un componente que utiliza el adaptador nativo para convertir los mensajes que debe transportar el transporte de comunicaciones.

tipo de cuerpo. Datos contenidos en un mensaje que representan la finalidad específica del mensaje, por ejemplo, añadir pedido de venta o sincronizar inventario. Pertenece al conjunto de valores de control de mensaje. Consulte "valores de control de mensaje".

El tipo de cuerpo también ayuda a especificar el tipo de mensaje. Consulte "tipo de mensaje".

tipo de mensaje. Mensaje que especifica una combinación exclusiva de categoría y tipo de cuerpo. Consulte "categoría del cuerpo" en la página 132 y "tipo de cuerpo".

trabajo. Software que forma parte del kernel. El trabajo sólo se utiliza en el modelo de entrega de inserción. El daemon del adaptador inicia y crea los trabajos. Cada trabajo gestiona un adaptador nativo. El trabajo entrega cada mensaje al adaptador destino adecuado.

transacción. Conjunto de operaciones que se deben ejecutar como una unidad de trabajo indivisible. Si todas las operaciones que constituyen la transacción obtienen un resultado satisfactorio, la transacción se confirma, lo que significa que se han llevado a cabo todas las operaciones. Si una o más de las operaciones que constituyen la transacción no se ejecutan correctamente, la transacción se restituye, lo que significa que no se ha realizado ninguna operación.

valores de control de mensaje. Término colectivo para un conjunto de valores de los mensajes (cuerpo y cabeceras) y del archivo de configuración que utiliza el kernel para controlar la clasificación y el direccionamiento de mensajes, y que utiliza cada adaptador para controlar, en parte, el modo en que lleva a cabo su función.

WebSphere Application Server Advanced Edition. Un producto de software de IBM que permite utilizar la especificación Sun Microsystems Enterprise JavaBeans (EJB). WebSphere Application Server Advanced Edition incluye un servidor EJB, en el que se pueden ejecutar Enterprise Beans. Los Enterprise Beans encapsulan los datos y la lógica empresarial que utilizan y comparten los clientes EJB. Existen dos tipos de Enterprise Beans: beans de sesión, que encapsulan objetos y tareas de poca duración específicos del cliente y beans de entidad, que encapsulan los datos permanentes. Se puede utilizar un tipo de bean de sesión denominado bean de mensajes de trabajo en la parte destino de MQSeries Adapter Kernel.

XML. Extensible Markup Language. Estándar W3C para la representación de datos.

Índice

© Copyright IBM Corp. 2000, 2001

W

X

Printed in Denmatrk by IBM Danmark A/S

GC10-3637-01